GNU Smalltalk Library Reference

Version 3.2.5
24 November 2017

by Paolo Bonzini

1 Base classes

1.1 Tree

Classes documented in this manual are **boldfaced**.

```
Autoload
Object
  Behavior
    ClassDescription
      Class
      Metaclass
  BlockClosure
  Boolean
    False
    True
  CObject
    CAggregate
      CArray
      CPtr
        CString
    CCallable
      CCallbackDescriptor
      CFunctionDescriptor
    CCompound
      CStruct
      CUnion
    CScalar
      CChar
      CDouble
      CFloat
      CInt
      CLong
      CLongDouble
      CLongLong
      CShort
      CSmalltalk
      CUChar
        CByte
          CBoolean
      CUInt
      CULong
      CULongLong
      CUShort
  ContextPart
    BlockContext
    MethodContext
  Continuation
  CType
    CPtrCType
      CArrayCType
    CScalarCType
      CStringCType
  Delay
  Directory
  DLD
  DumperProxy
    AlternativeObjectProxy
      NullProxy
        VersionableObjectProxy
      PluggableProxy
      SingletonProxy
  DynamicVariable
  Exception
    Error
      ArithmeticError
        ZeroDivide
      MessageNotUnderstood
      SystemExceptions.InvalidValue
        SystemExceptions.EmptyCollection
        SystemExceptions.InvalidArgument
          SystemExceptions.AlreadyDefined
          SystemExceptions.ArgumentOutOfRange
            SystemExceptions.IndexOutOfRange
          SystemExceptions.InvalidSize
          SystemExceptions.NotFound
            SystemExceptions.PackageNotAvailable
        SystemExceptions.InvalidProcessState
        SystemExceptions.InvalidState
        SystemExceptions.NotIndexable
        SystemExceptions.ProcessTerminated
        SystemExceptions.ReadOnlyObject
        SystemExceptions.WrongClass
          SystemExceptions.MustBeBoolean
      SystemExceptions.MutationError
      SystemExceptions.NotEnoughElements
      SystemExceptions.NotImplemented
        SystemExceptions.NotYetImplemented
        SystemExceptions.ShouldNotImplement
          SystemExceptions.SubclassResponsibility
          SystemExceptions.WrongMessageSent
      SystemExceptions.VMError
```

```
      SystemExceptions.BadReturn
      SystemExceptions.NoRunnableProcess
      SystemExceptions.PrimitiveFailed
        SystemExceptions.CInterfaceError
        SystemExceptions.FileError
        SystemExceptions.WrongArgumentCount
      SystemExceptions.SecurityError
      SystemExceptions.UserInterrupt
      SystemExceptions.VerificationError
  Halt
  Notification
    SystemExceptions.EndOfStream
    SystemExceptions.ProcessBeingTerminated
    Warning
  SystemExceptions.UnhandledException
ExceptionSet
FilePath
  File
  VFS.ArchiveMember
    VFS.TmpFileArchiveMember
      VFS.StoredZipMember
  VFS.FileWrapper
    VFS.ArchiveFile
      VFS.ZipFile
FileSegment
Getopt
Iterable
  Collection
    Bag
    HashedCollection
      Dictionary
        BindingDictionary
          AbstractNamespace
            Namespace
            RootNamespace
              SystemDictionary
        LookupTable
          IdentityDictionary
            MethodDictionary
          WeakValueLookupTable
            WeakValueIdentityDictionary
        WeakKeyDictionary
          WeakKeyIdentityDictionary
      Set
        IdentitySet
        WeakSet
          WeakIdentitySet
    MappedCollection
    SequenceableCollection
      ArrayedCollection
        Array
          WeakArray
        ByteArray
        CharacterArray
          String
            Symbol
          UnicodeString
        CompiledCode
          CompiledBlock
          CompiledMethod
        Interval
        LargeArrayedCollection
          LargeArray
          LargeByteArray
          LargeWordArray
        WordArray
      LinkedList
        Semaphore
      OrderedCollection
        RunArray
        SortedCollection
  Stream
    FileDescriptor
      FileStream
    Generator
    ObjectDumper
    PositionableStream
      ReadStream
      WriteStream
        ReadWriteStream
    Random
    TextCollector
Kernel.PackageInfo
  Package
Link
  Process
    CallinProcess
  SymLink
Magnitude
  Character
    UnicodeCharacter
  Date
    DateTime
  LookupKey
```

```
    Association
      HomedAssociation
        VariableBinding
    DeferredVariableBinding
    ProcessVariable
  Number
    Float
      FloatD
      FloatE
      FloatQ
    Fraction
    Integer
      LargeInteger
        LargeNegativeInteger
        LargePositiveInteger
          LargeZeroInteger
      SmallInteger
    ScaledDecimal
  Time
    Duration
Memory
Message
  DirectedMessage
MethodInfo
NetClients.URIResolver
NetClients.URL
ObjectMemory
PackageLoader
Permission
Point
ProcessEnvironment
ProcessorScheduler
Rectangle
RecursionLock
Regex
RegexResults
SecurityPolicy
SharedQueue
UndefinedObject
ValueAdaptor
  NullValueHolder
  PluggableAdaptor
    DelayedAdaptor
  ValueHolder
    Promise
```

1.2 AbstractNamespace

Defined in namespace Smalltalk
Superclass: BindingDictionary
Category: Language-Implementation

I am a special form of dictionary. Classes hold on an instance of me; it is called their ‘environment’.

1.2.1 AbstractNamespace class: instance creation

new Disabled - use #new to create instances

primNew: parent name: spaceName
Private - Create a new namespace with the given name and parent, and add to the parent a key that references it.

1.2.2 AbstractNamespace: accessing

allAssociations
Answer a Dictionary with all of the associations in the receiver and each of its superspaces (duplicate keys are associated to the associations that are deeper in the namespace hierarchy)

allBehaviorsDo: aBlock
Evaluate aBlock once for each class and metaclass in the namespace.

allClassObjectsDo: aBlock
Evaluate aBlock once for each class and metaclass in the namespace.

allClassesDo: aBlock
Evaluate aBlock once for each class in the namespace.

allMetaclassesDo: aBlock
Evaluate aBlock once for each metaclass in the namespace.

classAt: aKey
Answer the value corrisponding to aKey if it is a class. Fail if either aKey is not found or it is associated to something different from a class.

classAt: aKey ifAbsent: aBlock
Answer the value corrisponding to aKey if it is a class. Evaluate aBlock and answer its result if either aKey is not found or it is associated to something different from a class.

1.2.3 AbstractNamespace: compiling

addSharedPool: aDictionary
Import the given bindings for classes compiled with me as environment.

import: aDictionary
Import the given bindings for classes compiled with me as environment.

removeSharedPool: aDictionary
Remove aDictionary from my list of direct pools.

sharedPoolDictionaries
Answer the shared pools (not names) imported for my classes.

1.2.4 AbstractNamespace: copying

copyEmpty: newSize
Answer an empty copy of the receiver whose size is newSize

whileCurrentDo: aBlock
Evaluate aBlock with the current namespace set to the receiver. Answer the result of the evaluation.

1.2.5 AbstractNamespace: namespace hierarchy

addSubspace: aSymbol
Create a namespace named aSymbol, add it to the receiver's subspaces, and answer it.

allSubassociationsDo: aBlock
Invokes aBlock once for every association in each of the receiver's subspaces.

allSubspaces
Answer the direct and indirect subspaces of the receiver in a Set

allSubspacesDo: aBlock
Invokes aBlock for all subspaces, both direct and indirect.

allSuperspacesDo: aBlock
Evaluate aBlock once for each of the receiver's superspaces

includesClassNamed: aString
Answer whether the receiver or any of its superspaces include the given class – note that this method (unlike #includesKey:) does not require aString to be interned and (unlike #includesGlobalNamed:) only returns true if the global is a class object.

includesGlobalNamed: aString
Answer whether the receiver or any of its superspaces include the given key – note that this method (unlike #includesKey:) does not require aString to be interned but (unlike #includesClassNamed:) returns true even if the global is not a class object.

removeSubspace: aSymbol
Remove my subspace named aSymbol from the hierarchy.

selectSubspaces: aBlock
Return a Set of subspaces of the receiver satisfying aBlock.

selectSuperspaces: aBlock
Return a Set of superspaces of the receiver satisfying aBlock.

siblings Answer all the other children of the same namespace as the receiver.

siblingsDo: aBlock
Evaluate aBlock once for each of the other root namespaces, passing the namespace as a parameter.

subspaces Answer the receiver's direct subspaces

subspacesDo: aBlock
Invokes aBlock for all direct subspaces.

superspace
Answer the receiver's superspace.

superspace: aNamespace
Set the superspace of the receiver to be 'aNamespace'. Also adds the receiver as a subspace of it.

withAllSubspaces
Answer a Set containing the receiver together with its direct and indirect subspaces

withAllSubspacesDo: aBlock
Invokes aBlock for the receiver and all subclasses, both direct and indirect.

1.2.6 AbstractNamespace: overrides for superspaces

inheritedKeys
Answer a Set of all the keys in the receiver and its superspaces

set: key to: newValue
Assign newValue to the variable named as specified by 'key'. This method won't define a new variable; instead if the key is not found it will search in superspaces and raising an error if the variable cannot be found in any of the superspaces. Answer newValue.

set: key to: newValue ifAbsent: aBlock
Assign newValue to the variable named as specified by 'key'. This method won't define a new variable; instead if the key is not found it will search in superspaces and evaluate aBlock if it is not found. Answer newValue.

values Answer a Bag containing the values of the receiver

1.2.7 AbstractNamespace: printing

name Answer the receiver's name

name: aSymbol
Change the receiver's name to aSymbol

nameIn: aNamespace
Answer Smalltalk code compiling to the receiver when the current namespace is aNamespace

printOn: aStream
Print a representation of the receiver

storeOn: aStream
Store Smalltalk code compiling to the receiver

1.2.8 AbstractNamespace: testing

isNamespace
Answer 'true'.

isSmalltalk
Answer 'false'.

1.3 AlternativeObjectProxy

Defined in namespace Smalltalk
Superclass: DumperProxy
Category: Streams-Files

I am a proxy that uses the same ObjectDumper to store an object which is not the object to be dumped, but from which the dumped object can be reconstructed. I am an abstract class, using me would result in infinite loops because by default I try to store the same object again and again. See the method comments for more information

1.3.1 AlternativeObjectProxy class: instance creation

acceptUsageForClass: aClass
The receiver was asked to be used as a proxy for the class aClass. Answer whether the registration is fine. By default, answer true except if AlternativeObjectProxy itself is being used.

on: anObject
Answer a proxy to be used to save anObject. IMPORTANT: this method MUST be overridden so that the overridden version sends #on: to super passing an object that is NOT the same as anObject (alternatively, you can override #dumpTo:, which is what NullProxy does), because that would result in an infinite loop! This also means that AlternativeObjectProxy must never be used directly – only as a superclass.

1.3.2 AlternativeObjectProxy: accessing

object
Reconstruct the object stored in the proxy and answer it. A subclass will usually override this

object: theObject
Set the object to be dumped to theObject. This should not be overridden.

primObject
Reconstruct the object stored in the proxy and answer it. This method must not be overridden

1.4 ArithmeticError

Defined in namespace Smalltalk
Superclass: Error
Category: Language-Exceptions

An ArithmeticError exception is raised by numeric classes when a program tries to do something wrong, such as extracting the square root of a negative number.

1.4.1 ArithmeticError: description

description

Answer a textual description of the exception.

isResumable

Answer true. Arithmetic exceptions are by default resumable.

1.5 Array

Defined in namespace Smalltalk
Superclass: ArrayedCollection
Category: Collections-Sequenceable

My instances are objects that have array-like properties: they are directly indexable by integers starting at 1, and they are fixed in size. I inherit object creation behavior messages such as #with:, as well as iteration and general access behavior from SequenceableCollection.

1.5.1 Array class: instance creation

from: anArray

Answer anArray, which is expected to be an array specified with a brace-syntax expression per my inherited protocol.

1.5.2 Array: built ins

at: anIndex ifAbsent: aBlock

Answer the index-th indexed instance variable of the receiver

replaceFrom: start to: stop with: byteArray startingAt: replaceStart

Replace the characters from start to stop with new characters whose ASCII codes are contained in byteArray, starting at the replaceStart location of byteArray

1.5.3 Array: mutating objects

multiBecome: anArray

Transform every object in the receiver in each corresponding object in anArray. anArray and the receiver must have the same size

1.5.4 Array: printing

isLiteralObject

Answer whether the receiver is expressible as a Smalltalk literal.

printOn: aStream
Print a representation for the receiver on aStream

storeLiteralOn: aStream
Store a Smalltalk literal compiling to the receiver on aStream

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.5.5 Array: testing

isArray Answer 'true'.

1.6 ArrayedCollection

Defined in namespace Smalltalk
Superclass: SequenceableCollection
Category: Collections-Sequenceable
My instances are objects that are generally fixed size, and are accessed by an integer index. The ordering of my instance's elements is determined externally; I will not rearrange the order of the elements.

1.6.1 ArrayedCollection class: instance creation

join: aCollection
Where aCollection is a collection of SequenceableCollections, answer a new instance with all the elements therein, in order.

join: aCollection separatedBy: sepCollection
Where aCollection is a collection of SequenceableCollections, answer a new instance with all the elements therein, in order, each separated by an occurrence of sepCollection.

new: size withAll: anObject
Answer a collection with the given size, whose elements are all set to anObject

streamContents: aBlock
Create a ReadWriteStream on an empty instance of the receiver; pass the stream to aBlock, then retrieve its contents and answer them.

with: element1
Answer a collection whose only element is element1

with: element1 with: element2
Answer a collection whose only elements are the parameters in the order they were passed

with: element1 with: element2 with: element3
Answer a collection whose only elements are the parameters in the order they were passed

with: element1 with: element2 with: element3 with: element4
Answer a collection whose only elements are the parameters in the order they were passed

with: element1 with: element2 with: element3 with: element4 with: element5
Answer a collection whose only elements are the parameters in the order they were passed

withAll: aCollection
Answer a collection whose elements are the same as those in aCollection

1.6.2 ArrayedCollection: basic

, aSequenceableCollection
Answer a new instance of an ArrayedCollection containing all the elements in the receiver, followed by all the elements in aSequenceableCollection

add: value This method should not be called for instances of this class.

atAll: keyCollection
Answer a collection of the same kind returned by #collect:, that only includes the values at the given indices. Fail if any of the values in keyCollection is out of bounds for the receiver.

copyFrom: start to: stop
Answer a new collection containing all the items in the receiver from the start-th and to the stop-th

copyWith: anElement
Answer a new instance of an ArrayedCollection containing all the elements in the receiver, followed by the single item anElement

copyWithout: oldElement
Answer a copy of the receiver to which all occurrences of oldElement are removed

1.6.3 ArrayedCollection: built ins

size Answer the size of the receiver

1.6.4 ArrayedCollection: compiler

literalEquals: anObject
Not commented.

literalHash
Not commented.

1.6.5 ArrayedCollection: copying Collections

copyReplaceAll: oldSubCollection with: newSubCollection
Answer a new collection in which all the sequences matching oldSubCollection are replaced with newSubCollection

copyReplaceFrom: start to: stop with: replacementCollection
Answer a new collection of the same class as the receiver that contains the same elements as the receiver, in the same order, except for elements from index ‘start’ to index ‘stop’.

If start < stop, these are replaced by the contents of the replacementCollection. Instead, If start = (stop + 1), like in 'copyReplaceFrom: 4 to: 3 with: anArray', then every element of the receiver will be present in the answered copy; the operation will be an append if stop is equal to the size of the receiver or, if it is not, an insert before index 'start'.

copyReplaceFrom: start to: stop withObject: anObject
Answer a new collection of the same class as the receiver that contains the same elements as the receiver, in the same order, except for elements from index 'start' to index 'stop'.

If start < stop, these are replaced by stop-start+1 copies of anObject. Instead, If start = (stop + 1), then every element of the receiver will be present in the answered copy; the operation will be an append if stop is equal to the size of the receiver or, if it is not, an insert before index 'start'.

reverse Answer the receivers' contents in reverse order

1.6.6 ArrayedCollection: enumerating the elements of a collection

collect: aBlock
Answer a new instance of an ArrayedCollection containing all the results of evaluating aBlock passing each of the receiver's elements

reject: aBlock
Answer a new instance of an ArrayedCollection containing all the elements in the receiver which, when passed to aBlock, answer false

select: aBlock
Answer a new instance of an ArrayedCollection containing all the elements in the receiver which, when passed to aBlock, answer true

with: aSequenceableCollection collect: aBlock
Evaluate aBlock for each pair of elements took respectively from the receiver and from aSequenceableCollection; answer a collection of the same kind of the receiver, made with the block's return values. Fail if the receiver has not the same size as aSequenceableCollection.

1.6.7 ArrayedCollection: sorting

sorted Return a copy of the receiver sorted according to the default sort block, which uses #<= to compare items.

sorted: sortBlock
Return a copy of the receiver sorted according to the given sort block, which accepts pair of items and returns true if the first item is less than the second one.

1.6.8 ArrayedCollection: storing

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.6.9 ArrayedCollection: streams

writeStream
Answer a WriteStream streaming on the receiver

1.7 Association

Defined in namespace Smalltalk
Superclass: LookupKey
Category: Language-Data types
My instances represent a mapping between two objects. Typically, my "key" object is a symbol, but I don't require this. My "value" object has no conventions associated with it; it can be any object at all.

1.7.1 Association class: basic

key: aKey value: aValue
Answer a new association with the given key and value

1.7.2 Association: accessing

environment
Answer nil. This is present to achieve polymorphism with instances of VariableBinding.

environment: aNamespace
Do nothing. This is present to achieve polymorphism with instances of VariableBinding.

key: aKey value: aValue
Set the association's key to aKey, and its value to aValue

value Answer the association's value

value: aValue
Set the association's value to aValue

1.7.3 Association: finalization

mourn Finalize the receiver

1.7.4 Association: printing

printOn: aStream
Put on aStream a representation of the receiver

1.7.5 Association: storing

storeOn: aStream
Put on aStream some Smalltalk code compiling to the receiver

1.7.6 Association: testing

= anAssociation
Answer whether the association's key and value are the same as anAssociation's, or false if anAssociation is not an Association. As a special case, identical values are considered equal even if #= returns false (as is the case for NaN floating-point values).

hash Answer an hash value for the receiver

1.8 Autoload

Defined in namespace Smalltalk
Superclass: none
Category: Examples-Useful tools
I am not a part of the normal Smalltalk kernel class system. I provide the ability to do late ("on-demand") loading of class definitions. Through me, you can define any class to be loaded when any message is sent to the class itself (such as to create an instance) or to its metaclass (such as #methodsFor: to extend it with class-side methods).

1.8.1 Autoload class: instance creation

class: nameSymbol from: fileNameString
Make Smalltalk automatically load the class named nameSymbol from fileNameString when needed

class: nameSymbol in: aNamespace from: fileNameString
Make Smalltalk automatically load the class named nameSymbol and residing in aNamespace from fileNameString when needed

class: nameSymbol in: aNamespace loader: anObject
Make Smalltalk automatically load the class named nameSymbol and residing in aNamespace. When the class is needed, anObject will be sent #autoload. By default, instances of FilePath and Package can be used.

class: nameSymbol loader: anObject
Make Smalltalk automatically load the class named nameSymbol. When the class is needed, anObject will be sent #autoload. By default, instances of FilePath and Package can be used.

1.8.2 Autoload: accessing

class We need it to access the metaclass instance, because that's what will load the file.

doesNotUnderstand: aMessage
Load the class and resend the message to it

1.9 Bag

Defined in namespace Smalltalk
Superclass: Collection
Category: Collections-Unordered

My instances are unordered collections of objects. You can think of me as a set with a memory; that is, if the same object is added to me twice, then I will report that that element has been stored twice.

1.9.1 Bag class: basic

new Answer a new instance of the receiver

new: size Answer a new instance of the receiver, with space for size distinct objects

1.9.2 Bag: adding

add: newObject
Add an occurrence of newObject to the receiver. Answer newObject. Fail if newObject is nil.

add: newObject withOccurrences: anInteger
If anInteger > 0, add anInteger occurrences of newObject to the receiver. If anInteger < 0, remove them. Answer newObject. Fail if newObject is nil.

1.9.3 Bag: enumerating the elements of a collection

asSet Answer a set with the elements of the receiver

do: aBlock
Evaluate the block for all members in the collection.

1.9.4 Bag: extracting items

sortedByCount
Answer a collection of counts with elements, sorted by decreasing count.

1.9.5 Bag: printing

printOn: aStream
Put on aStream a representation of the receiver

1.9.6 Bag: removing

remove: oldObject ifAbsent: anExceptionBlock
Remove oldObject from the collection and return it. If can't be found, answer instead the result of evaluationg anExceptionBlock

1.9.7 Bag: storing

storeOn: aStream
Put on aStream some Smalltalk code compiling to the receiver

1.9.8 Bag: testing collections

= aBag Answer whether the receiver and aBag contain the same objects

hash Answer an hash value for the receiver

includes: anObject
Answer whether we include anObject

occurrencesOf: anObject
Answer the number of occurrences of anObject found in the receiver

size Answer the total number of objects found in the receiver

1.10 Behavior

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation
I am the parent class of all "class" type methods. My instances know about the subclass/superclass relationships between classes, contain the description that instances are created from, and hold the method dictionary that's associated with each class. I provide methods for compiling methods, modifying the class inheritance hierarchy, examining the method dictionary, and iterating over the class hierarchy.

1.10.1 Behavior: accessing class hierarchy

allSubclasses
Answer the direct and indirect subclasses of the receiver in a Set

allSuperclasses
Answer all the receiver's superclasses in a collection

subclasses Answer the direct subclasses of the receiver in a Set

superclass Answer the receiver's superclass (if any, otherwise answer nil)

withAllSubclasses
Answer a Set containing the receiver together with its direct and indirect subclasses

withAllSuperclasses
Answer the receiver and all of its superclasses in a collection

1.10.2 Behavior: accessing instances and variables

allClassVarNames
Return all the class variables understood by the receiver

allInstVarNames
Answer the names of every instance variables the receiver contained in the receiver's instances

allInstances
Returns a set of all instances of the receiver

allSharedPoolDictionaries
Return the shared pools defined by the class and any of its superclasses, in the correct search order.

allSharedPools
Return the names of the shared pools defined by the class and any of its superclasses, in the correct search order.

classPool
Answer the class pool dictionary. Since Behavior does not support classes with class variables, we answer an empty one; adding variables to it results in an error.

classVarNames
Answer all the class variables for instances of the receiver

indexOfInstVar: aString
Answer the index of aString in the fixed instance variables of the instances of the receiver, or 0 if the variable is missing.

indexOfInstVar: aString ifAbsent: aBlock
Answer the index of aString in the fixed instance variables of the instances of the receiver, or 0 if the variable is missing.

instVarNames
Answer an Array containing the instance variables defined by the receiver

instanceCount
Return a count of all the instances of the receiver

sharedPools
Return the names of the shared pools defined by the class

subclassInstVarNames
Answer the names of the instance variables the receiver inherited from its superclass

1.10.3 Behavior: accessing the method dictionary

>> selector
Return the compiled method associated with selector, from the local method dictionary. Error if not found.

allSelectors
Answer a Set of all the selectors understood by the receiver

compiledMethodAt: selector
Return the compiled method associated with selector, from the local method dictionary. Error if not found.

compiledMethodAt: selector ifAbsent: aBlock
Return the compiled method associated with selector, from the local method dictionary. Evaluate aBlock if not found.

formattedSourceStringAt: selector
Answer the method source code as a formatted string (if available) for the given selector. Requires package Parser.

lookupAllSelectors: aSelector
: Answer a Set of all the compiled method associated with selector. from the local method dictionary and all of the superclasses.

lookupSelector: aSelector
: Return the compiled method associated with selector, from the local method dictionary or one of a superclass; return nil if not found.

parseTreeFor: selector
: Answer the parse tree for the given selector, or nil if there was an error. Requires the Parser package to be loaded.

selectorAt: method
: Return selector for the given CompiledMethod

selectors
: Answer a Set of the receiver's selectors

sourceCodeAt: selector
: Answer source code (if available) for the given selector.

sourceCodeAt: selector ifAbsent: aBlock
: Answer source code (if available) for the given selector.

sourceMethodAt: selector
: This is too dependent on the original implementation

1.10.4 Behavior: built ins

basicNewInFixedSpace
: Create a new instance of a class with no indexed instance variables. The instance is guaranteed not to move across garbage collections. Like #basicNew, this method should not be overridden.

basicNewInFixedSpace: numInstanceVariables
: Create a new instance of a class with indexed instance variables. The instance has numInstanceVariables indexed instance variables. The instance is guaranteed not to move across garbage collections. Like #basicNew:, this method should not be overridden.

flushCache
: Invalidate the method cache kept by the virtual machine. This message should not need to be called by user programs.

methodsFor: category ifTrue: condition
: Compile the following code inside the receiver, with the given category, if condition is true; else ignore it

primCompile: code
: Compile the code, a string or readable stream, with no category. Fail if the code does not obey Smalltalk syntax. Answer the generated CompiledMethod if it does.

: Do not send this in user code; use #compile: or related methods instead.

primCompile: code ifError: aBlock
As with #primCompile:, but evaluate aBlock (passing the file name, line number and description of the error) if the code does not obey Smalltalk syntax.

Do not send this in user code; use #compile:ifError: or related methods instead.

someInstance
Private - Answer the first instance of the receiver in the object table

1.10.5 Behavior: builtin

basicNew Create a new instance of a class with no indexed instance variables; this method must not be overridden.

basicNew: numInstanceVariables
Create a new instance of a class with indexed instance variables. The instance has numInstanceVariables indexed instance variables; this method must not be overridden.

new Create a new instance of a class with no indexed instance variables

new: numInstanceVariables
Create a new instance of a class with indexed instance variables. The instance has numInstanceVariables indexed instance variables.

1.10.6 Behavior: compilation

scopeDictionary
Answer the dictionary that is used when the receiver is before a period in Smalltalk source code.

1.10.7 Behavior: compilation (alternative)

methods Don't use this, it's only present to file in from Smalltalk/V

methodsFor
Don't use this, it's only present to file in from Dolphin Smalltalk

methodsFor: category ifFeatures: features
Start compiling methods in the receiver if this implementation of Smalltalk has the given features, else skip the section

methodsFor: category stamp: notUsed
Don't use this, it's only present to file in from Squeak

privateMethods
Don't use this, it's only present to file in from IBM Smalltalk

publicMethods
Don't use this, it's only present to file in from IBM Smalltalk

1.10.8 Behavior: compiling

compilerClass
Return the class that will be used to compile the parse nodes into bytecodes.

1.10.9 Behavior: compiling methods

methodsFor: aCategoryString
: Calling this method prepares the parser to receive methods to be compiled and installed in the receiver's method dictionary. The methods are put in the category identified by the parameter.

poolResolution
: Answer a PoolResolution class to be used for resolving pool variables while compiling methods on this class.

1.10.10 Behavior: creating a class hierarchy

addSubclass: aClass
: Add aClass asone of the receiver's subclasses.

removeSubclass: aClass
: Remove aClass from the list of the receiver's subclasses

superclass: aClass
: Set the receiver's superclass.

1.10.11 Behavior: enumerating

allInstancesDo: aBlock
: Invokes aBlock for all instances of the receiver

allSubclassesDo: aBlock
: Invokes aBlock for all subclasses, both direct and indirect.

allSubinstancesDo: aBlock
: Invokes aBlock for all instances of each of the receiver's subclasses.

allSuperclassesDo: aBlock
: Invokes aBlock for all superclasses, both direct and indirect.

selectSubclasses: aBlock
: Return a Set of subclasses of the receiver satisfying aBlock.

selectSuperclasses: aBlock
: Return a Set of superclasses of the receiver satisfying aBlock.

subclassesDo: aBlock
: Invokes aBlock for all direct subclasses.

withAllSubclassesDo: aBlock
: Invokes aBlock for the receiver and all subclasses, both direct and indirect.

withAllSuperclassesDo: aBlock
: Invokes aBlock for the receiver and all superclasses, both direct and indirect.

1.10.12 Behavior: evaluating

evalString: aString to: anObject
: Answer the stack top at the end of the evaluation of the code in aString. The code is executed as part of anObject

evalString: aString to: anObject ifError: aBlock
Answer the stack top at the end of the evaluation of the code in aString. If aString cannot be parsed, evaluate aBlock (see compile:ifError:). The code is executed as part of anObject

evaluate: code
Evaluate Smalltalk expression in 'code' and return result.

evaluate: code ifError: block
Evaluate 'code'. If a parsing error is detected, invoke 'block'

evaluate: code notifying: requestor
Evaluate Smalltalk expression in 'code'. If a parsing error is encountered, send #error: to requestor

evaluate: code to: anObject
Evaluate Smalltalk expression as part of anObject's method definition

evaluate: code to: anObject ifError: block
Evaluate Smalltalk expression as part of anObject's method definition. This method is used to support Inspector expression evaluation. If a parsing error is encountered, invoke error block, 'block'

1.10.13 Behavior: instance creation

newInFixedSpace
Create a new instance of a class without indexed instance variables. The instance is guaranteed not to move across garbage collections. If a subclass overrides #new, the changes will apply to this method too.

newInFixedSpace: numInstanceVariables
Create a new instance of a class with indexed instance variables. The instance has numInstanceVariables indexed instance variables. The instance is guaranteed not to move across garbage collections. If a subclass overrides #new:, the changes will apply to this method too.

1.10.14 Behavior: instance variables

addInstVarName: aString
Add the given instance variable to instance of the receiver

instanceVariableNames: instVarNames
Set the instance variables for the receiver to be those in instVarNames

removeInstVarName: aString
Remove the given instance variable from the receiver and recompile all of the receiver's subclasses

1.10.15 Behavior: method dictionary

addSelector: selector withMethod: compiledMethod
Add the given compiledMethod to the method dictionary, giving it the passed selector. Answer compiledMethod

compile: code
: Compile method source. If there are parsing errors, answer nil. Else, return a CompiledMethod result of compilation

compile: code ifError: block
: Compile method source. If there are parsing errors, invoke exception block, 'block' passing file name, line number and error. Return a CompiledMethod result of compilation

compile: code notifying: requestor
: Compile method source. If there are parsing errors, send #error: to the requestor object, else return a CompiledMethod result of compilation

compileAll
: Recompile all selectors in the receiver. Ignore errors.

compileAll: aNotifier
: Recompile all selectors in the receiver. Notify aNotifier by sending #error: messages if something goes wrong.

compileAllSubclasses
: Recompile all selector of all subclasses. Notify aNotifier by sending #error: messages if something goes wrong.

compileAllSubclasses: aNotifier
: Recompile all selector of all subclasses. Notify aNotifier by sending #error: messages if something goes wrong.

createGetMethod: what
: Create a method accessing the variable ‘what’.

createGetMethod: what default: value
: Create a method accessing the variable ‘what’, with a default value of ‘value’, using lazy initialization

createSetMethod: what
: Create a method which sets the variable ‘what’.

decompile: selector
: Decompile the bytecodes for the given selector.

defineAsyncCFunc: cFuncNameString withSelectorArgs: selectorAndArgs args: argsArray
: Please lookup the part on the C interface in the manual. This method is deprecated, you should use the asyncCCall:args: attribute.

defineCFunc: cFuncNameString withSelectorArgs: selectorAndArgs returning: returnTypeSymbol args: argsArray
: Please lookup the part on the C interface in the manual. This method is deprecated, you should use the cCall:returning:args: attribute.

edit: selector
: Open Emacs to edit the method with the passed selector, then compile it

methodDictionary
: Answer the receiver's method dictionary. Don't modify the method dictionary unless you exactly know what you're doing

methodDictionary: aDictionary
: Set the receiver's method dictionary to aDictionary

recompile: selector
: Recompile the given selector, answer nil if something goes wrong or the new CompiledMethod if everything's ok.

recompile: selector notifying: aNotifier
: Recompile the given selector. If there are parsing errors, send #error: to the aNotifier object, else return a CompiledMethod result of compilation

removeSelector: selector
: Remove the given selector from the method dictionary, answer the CompiledMethod attached to that selector

removeSelector: selector ifAbsent: aBlock
: Remove the given selector from the method dictionary, answer the CompiledMethod attached to that selector. If the selector cannot be found, answer the result of evaluating aBlock.

selectorsAndMethodsDo: aBlock
: Evaluate aBlock, passing for each evaluation a selector that's defined in the receiver and the corresponding method.

1.10.16 Behavior: parsing class declarations

parseInstanceVariableString: variableString
: As with #parseVariableString:, but answer symbols that name the variables instead of strings.

parseVariableString: aString
: Answer an array of instance variable names. aString should specify these in traditional file-in 'instanceVariableNames' format. Signal an error if aString contains something other than valid Smalltalk variables.

1.10.17 Behavior: pluggable behavior (not yet implemented)

debuggerClass
: Answer which class is to be used to debug a chain of contexts which includes the receiver. nil means 'do not debug'; other classes are sent #debuggingPriority and the one with the highest priority is picked.

decompilerClass
: Answer the class that can be used to decompile methods, or nil if there is none (as is the case now).

evaluatorClass
: Answer the class that can be used to evaluate doits, or nil if there is none (as is the case now).

parserClass
: Answer the class that can be used to parse methods, or nil if there is none (as is the case now).

1.10.18 Behavior: printing hierarchy

hierarchyIndent
: Answer the indent to be used by #printHierarchy - 4 by default

printFullHierarchy
: Print my full hierarchy (i.e. all my superclasses and subclasses) on the terminal.

printHierarchy
: Print my entire subclass hierarchy on the terminal.

1.10.19 Behavior: source code

formattedSourceStringAt: aSelector ifAbsent: aBlock
: Answer the method source code as a formatted string. Requires package Parser.

1.10.20 Behavior: still unclassified

allSharedPoolDictionariesDo: aBlock
: Answer the shared pools visible from methods in the metaclass, in the correct search order.

parseNodeAt: selector
: Available only when the Parser package is loaded–Answer an RBMethodNode that compiles to my method named by selector.

updateInstanceVars: variableArray shape: shape
: Update instance variables and instance spec of the class and all its subclasses. variableArray lists the new variables, including inherited ones.

1.10.21 Behavior: support for lightweight classes

article Answer an article (‘a’ or ‘an’) which is ok for the receiver’s name

asClass Answer the first superclass that is a full-fledged Class object

environment
: Answer the namespace that this class belongs to - the same as the superclass, since Behavior does not support namespaces yet.

name Answer the class name; this prints to the name of the superclass enclosed in braces. This class name is used, for example, to print the receiver.

nameIn: aNamespace
: Answer the class name when the class is referenced from aNamespace - a dummy one, since Behavior does not support names.

printOn: aStream in: aNamespace
: Answer the class name when the class is referenced from aNamespace - a dummy one, since Behavior does not support names.

securityPolicy
Not commented.

securityPolicy: aSecurityPolicy
This method should not be called for instances of this class.

1.10.22 Behavior: testing functionality

isBehavior Answer 'true'.

1.10.23 Behavior: testing the class hierarchy

includesBehavior: aClass
Returns true if aClass is the receiver or a superclass of the receiver.

inheritsFrom: aClass
Returns true if aClass is a superclass of the receiver

kindOfSubclass
Return a string indicating the type of class the receiver is

shape Answer the symbolic shape of my instances.

shape: shape
Give the provided shape to the receiver's instances. The shape can be nil, or one of #byte #int8 #character #short #word #ushort #int #uint #int64 #uint64 #utf32 #float #double or #pointer. In addition, the special value #inherit means to use the shape of the superclass; note however that this is a static setting, and subclasses that used #inherit are not mutated when the superclass adopts a different shape.

1.10.24 Behavior: testing the form of the instances

instSize Answer how many fixed instance variables are reserved to each of the receiver's instances

isBits Answer whether my instances' variables are immediate, non-OOP values.

isFixed Answer whether the receiver's instances have no indexed instance variables

isIdentity Answer whether x = y implies x == y for instances of the receiver

isImmediate
Answer whether, if x is an instance of the receiver, x copy == x

isPointers Answer whether the instance variables of the receiver's instances are objects

isVariable Answer whether the receiver's instances have indexed instance variables

1.10.25 Behavior: testing the method dictionary

canUnderstand: selector
Returns true if the instances of the receiver understand the given selector

hasMethods
Return whether the receiver has any methods defined

includesSelector: selector
Returns true if the local method dictionary contains the given selector

scopeHas: name ifTrue: aBlock
If methods understood by the receiver's instances have access to a symbol named 'name', evaluate aBlock

whichClassIncludesSelector: selector
Answer which class in the receiver's hierarchy contains the implementation of selector used by instances of the class (nil if none does)

whichSelectorsAccess: instVarName
Answer a Set of selectors which access the given instance variable

whichSelectorsAssign: instVarName
Answer a Set of selectors which read the given instance variable

whichSelectorsRead: instVarName
Answer a Set of selectors which read the given instance variable

whichSelectorsReferTo: anObject
Returns a Set of selectors that refer to anObject

whichSelectorsReferToByteCode: aByteCode
Return the collection of selectors in the class which reference the byte code, aByteCode

1.11 BindingDictionary

Defined in namespace Smalltalk
Superclass: Dictionary
Category: Language-Implementation
I am a special form of dictionary that provides special ways to access my keys, which typically begin with an uppercase letter; also, my associations are actually VariableBinding instances.

My keys are (expected to be) symbols, so I use == to match searched keys to those in the dictionary – this is done expecting that it brings a bit more speed.

1.11.1 BindingDictionary: accessing

define: aSymbol
Define aSymbol as equal to nil inside the receiver. Fail if such a variable already exists (use #at:put: if you don't want to fail)

doesNotUnderstand: aMessage
Try to map unary selectors to read accesses to the Namespace, and one-argument keyword selectors to write accesses. Note that: a) this works only if the selector has an uppercase first letter; and b) 'aNamespace Variable: value' is the same as 'aNamespace set: #Variable to: value', not the same as 'aNamespace at: #Variable put: value' — the latter always refers to the current namespace, while the former won't define a new variable, instead searching in superspaces (and raising an error if the variable cannot be found).

environment
Answer the environment to which the receiver is connected. This can be the class for a dictionary that holds class variables, or the super-namespace. In general it is used to compute the receiver's name.

environment: anObject
Set the environment to which the receiver is connected. This can be the class for a dictionary that holds class variables, or the super-namespace. In general it is used to compute the receiver's name.

import: aSymbol from: aNamespace
Add to the receiver the symbol aSymbol, associated to the same value as in aNamespace. Fail if aNamespace does not contain the given key.

name Answer the receiver's name, which by default is the same as the name of the receiver's environment.

nameIn: aNamespace
Answer the receiver's name when referred to from aNamespace; by default the computation is deferred to the receiver's environment.

1.11.2 BindingDictionary: basic & copying

= arg Answer whether the receiver is equal to arg. The equality test is by default the same as that for equal objects. = must not fail; answer false if the receiver cannot be compared to arg

hash Answer an hash value for the receiver. This is the same as the object's #identityHash.

1.11.3 BindingDictionary: copying

copy Answer the receiver.

copyEmpty: newSize
Answer an empty copy of the receiver whose size is newSize

copyEmptyForCollect
Answer an empty copy of the receiver which is filled in to compute the result of #collect:

copyEmptyForCollect: size
Answer an empty copy of the receiver which is filled in to compute the result of #collect:

deepCopy Answer the receiver.

shallowCopy
Answer the receiver.

1.11.4 BindingDictionary: forward declarations

at: key put: value
Store value as associated to the given key. If any, recycle Associations temporarily stored by the compiler inside the 'Undeclared' dictionary.

1.11.5 BindingDictionary: printing

printOn: aStream in: aNamespace
Print the receiver's name when referred to from aNamespace; by default the computation is deferred to the receiver's environment.

1.11.6 BindingDictionary: testing

species Answer 'IdentityDictionary'.

1.12 BlockClosure

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation
I am a factotum class. My instances represent Smalltalk blocks, portions of executeable code that have access to the environment that they were declared in, take parameters, and can be passed around as objects to be executed by methods outside the current class. Block closures are sent a message to compute their value and create a new execution context; this property can be used in the construction of control flow methods. They also provide some methods that are used in the creation of Processes from blocks.

1.12.1 BlockClosure class: instance creation

block: aCompiledBlock
Answer a BlockClosure that activates the passed CompiledBlock.

block: aCompiledBlock receiver: anObject
Answer a BlockClosure that activates the passed CompiledBlock with the given receiver.

block: aCompiledBlock receiver: anObject outerContext: aContext
Answer a BlockClosure that activates the passed CompiledBlock with the given receiver.

numArgs: args numTemps: temps bytecodes: bytecodes depth: depth literals: literalArray
Answer a BlockClosure for a new CompiledBlock that is created using the passed parameters. To make it work, you must put the BlockClosure into a CompiledMethod's literals.

1.12.2 BlockClosure class: testing

isImmediate
Answer whether, if x is an instance of the receiver, x copy == x

1.12.3 BlockClosure: accessing

argumentCount
Answer the number of arguments passed to the receiver

block Answer the CompiledBlock which contains the receiver's bytecodes

block: aCompiledBlock
Set the CompiledBlock which contains the receiver's bytecodes

finalIP Answer the last instruction that can be executed by the receiver

fixTemps This should fix the values of the temporary variables used in the block that are ordinarily shared with the method in which the block is defined. Not defined yet, but it is not harmful that it isn't. Answer the receiver.

initialIP Answer the initial instruction pointer into the receiver.

method Answer the CompiledMethod in which the receiver lies

numArgs Answer the number of arguments passed to the receiver

numTemps
Answer the number of temporary variables used by the receiver

outerContext
Answer the method/block context which is the immediate outer of the receiver

outerContext: containingContext
Set the method/block context which is the immediate outer of the receiver

receiver Answer the object that is used as 'self' when executing the receiver (if nil, it might mean that the receiver is not valid though...)

receiver: anObject
Set the object that is used as 'self' when executing the receiver

stackDepth
Answer the number of stack slots needed for the receiver

1.12.4 BlockClosure: built ins

cull: arg1 Evaluate the receiver, passing arg1 as the only parameter if the receiver has parameters.

cull: arg1 cull: arg2
Evaluate the receiver, passing arg1 and arg2 as parameters if the receiver accepts them.

cull: arg1 cull: arg2 cull: arg3
Evaluate the receiver, passing arg1, arg2 and arg3 as parameters if the receiver accepts them.

value Evaluate the receiver passing no parameters

value: arg1
Evaluate the receiver passing arg1 as the only parameter

value: arg1 value: arg2
Evaluate the receiver passing arg1 and arg2 as the parameters

value: arg1 value: arg2 value: arg3
Evaluate the receiver passing arg1, arg2 and arg3 as the parameters

valueWithArguments: argumentsArray
Evaluate the receiver passing argArray's elements as the parameters

1.12.5 BlockClosure: control structures

repeat Evaluate the receiver 'forever' (actually until a return is executed or the process is terminated).

whileFalse Evaluate the receiver until it returns true

whileFalse: aBlock
Evaluate the receiver. If it returns false, evaluate aBlock and restart

whileTrue Evaluate the receiver until it returns false

whileTrue: aBlock
Evaluate the receiver. If it returns true, evaluate aBlock and restart

1.12.6 BlockClosure: exception handling

ifError: aBlock
Evaluate the receiver; when #error: is called, pass to aBlock the receiver and the parameter, and answer the result of evaluating aBlock. If another exception is raised, it is passed to an outer handler; if no exception is raised, the result of evaluating the receiver is returned.

on: anException do: aBlock
Evaluate the receiver; when anException is signaled, evaluate aBlock passing a Signal describing the exception. Answer either the result of evaluating the receiver or the parameter of a Signal>>#return:

on: e1 do: b1 on: e2 do: b2
Evaluate the receiver; when e1 or e2 are signaled, evaluate respectively b1 or b2, passing a Signal describing the exception. Answer either the result of evaluating the receiver or the argument of a Signal>>#return:

on: e1 do: b1 on: e2 do: b2 on: e3 do: b3
Evaluate the receiver; when e1, e2 or e3 are signaled, evaluate respectively b1, b2 or b3, passing a Signal describing the exception. Answer either the result of evaluating the receiver or the parameter of a Signal>>#return:

on: e1 do: b1 on: e2 do: b2 on: e3 do: b3 on: e4 do: b4
Evaluate the receiver; when e1, e2, e3 or e4 are signaled, evaluate respectively b1, b2, b3 or b4, passing a Signal describing the exception. Answer either the result of evaluating the receiver or the parameter of a Signal>>#return:

on: e1 do: b1 on: e2 do: b2 on: e3 do: b3 on: e4 do: b4 on: e5 do: b5
Evaluate the receiver; when e1, e2, e3, e4 or e5 are signaled, evaluate respectively b1, b2, b3, b4 or b5, passing a Signal describing the exception. Answer either the result of evaluating the receiver or the parameter of a Signal>>-#return:

1.12.7 BlockClosure: multiple process

fork Create a new process executing the receiver and start it

forkAt: priority
Create a new process executing the receiver with given priority and start it

forkWithoutPreemption
Evaluate the receiver in a process that cannot be preempted. If the receiver expect a parameter, pass the current process.

newProcess
Create a new process executing the receiver in suspended state. The priority is the same as for the calling process. The receiver must not contain returns

newProcessWith: anArray
Create a new process executing the receiver with the passed arguments, and leave it in suspended state. The priority is the same as for the calling process. The receiver must not contain returns

valueWithoutInterrupts
Evaluate aBlock and delay all interrupts that are requested to the active process during its execution to after aBlock returns.

valueWithoutPreemption
Evaluate the receiver with external interrupts disabled. This effectively disables preemption as long as the block does not explicitly yield control, wait on semaphores, and the like.

1.12.8 BlockClosure: overriding

copy Answer the receiver.

deepCopy Answer a shallow copy.

1.12.9 BlockClosure: testing

hasMethodReturn
Answer whether the block contains a method return

1.12.10 BlockClosure: unwind protection

ensure: aBlock
Evaluate the receiver; when any exception is signaled exit returning the result of evaluating aBlock; if no exception is raised, return the result of evaluating aBlock when the receiver has ended

ifCurtailed: aBlock
Evaluate the receiver; if its execution triggers an unwind which truncates the execution of the block (‘curtails’ the block), evaluate aBlock. The three cases which can curtail the execution of the receiver are: a non-local return in the receiver, a non-local return in a block evaluated by the receiver which returns past the receiver itself, and an exception raised and not resumed during the execution of the receiver.

valueWithUnwind
: Evaluate the receiver. Any errors caused by the block will cause a backtrace, but execution will continue in the method that sent #valueWithUnwind, after that call. Example: [1 / 0] valueWithUnwind. 'unwind works!' printNl.

  Important: this method is public, but it is intended to be used in very special cases (as a rule of thumb, use it only when the corresponding C code uses the _gst_prepare_execution_environment and _gst_finish_execution_environment functions). You should usually rely on #ensure: and #on:do:.

1.13 BlockContext

Defined in namespace Smalltalk
Superclass: ContextPart
Category: Language-Implementation

My instances represent executing Smalltalk blocks, which are portions of executeable code that have access to the environment that they were declared in, take parameters, and result from BlockClosure objects created to be executed by methods outside the current class. Block contexts are created by messages sent to compute a closure's value. They contain a stack and also provide some methods that can be used in inspection or debugging.

1.13.1 BlockContext: accessing

caller Answer the context that called the receiver

home Answer the MethodContext to which the receiver refers, or nil if it has been optimized away

isBlock Answer whether the receiver is a block context

isDisabled Answers false, because contexts that are skipped when doing a return are always MethodContexts. BlockContexts are removed from the chain whenever a non-local return is done, while MethodContexts need to stay there in case there is a non-local return from the #ensure: block.

isEnvironment
: To create a valid execution environment for the interpreter even before it starts, GST creates a fake context whose selector is nil and which can be used as a marker for the current execution environment. Answer whether the receiver is that kind of context (always false, since those contexts are always MethodContexts).

isUnwind Answers whether the context must continue execution even after a non-local return (a return from the enclosing method of a block, or a call to the #continue: method of ContextPart). Such contexts are created only by #ensure: and are always MethodContexts.

nthOuterContext: n
: Answer the n-th outer block/method context for the receiver

outerContext
: Answer the outer block/method context for the receiver

1.13.2 BlockContext: debugging

isInternalExceptionHandlingContext
Answer whether the receiver is a context that should be hidden to the user when presenting a backtrace. Such contexts are never blocks, but check the rest of the chain.

1.13.3 BlockContext: printing

printOn: aStream
Print a representation for the receiver on aStream

1.14 Boolean

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Data types
I have two instances in the Smalltalk system: true and false. I provide methods that are conditional on boolean values, such as conditional execution and loops, and conditional testing, such as conditional and and conditional or. I should say that I appear to provide those operations; my subclasses True and False actually provide those operations.

1.14.1 Boolean class: testing

isIdentity Answer whether x = y implies x == y for instances of the receiver

isImmediate
Answer whether, if x is an instance of the receiver, x copy == x

1.14.2 Boolean: basic

& aBoolean
This method's functionality should be implemented by subclasses of Boolean

and: aBlock
This method's functionality should be implemented by subclasses of Boolean

eqv: aBoolean
This method's functionality should be implemented by subclasses of Boolean

ifFalse: falseBlock
This method's functionality should be implemented by subclasses of Boolean

ifFalse: falseBlock ifTrue: trueBlock
This method's functionality should be implemented by subclasses of Boolean

ifTrue: trueBlock
This method's functionality should be implemented by subclasses of Boolean

ifTrue: trueBlock ifFalse: falseBlock
This method's functionality should be implemented by subclasses of Boolean

not This method's functionality should be implemented by subclasses of Boolean

or: aBlock This method's functionality should be implemented by subclasses of Boolean

xor: aBoolean
This method's functionality should be implemented by subclasses of Boolean

| aBoolean
This method's functionality should be implemented by subclasses of Boolean

1.14.3 Boolean: C hacks

asCBooleanValue
This method's functionality should be implemented by subclasses of Boolean

1.14.4 Boolean: overriding

deepCopy Answer the receiver.

shallowCopy
Answer the receiver.

1.14.5 Boolean: storing

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

storeLiteralOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

storeOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

1.15 ByteArray

Defined in namespace Smalltalk
Superclass: ArrayedCollection
Category: Collections-Sequenceable
My instances are similar to strings in that they are both represented as a sequence of bytes, but my individual elements are integers, where as a String's elements are characters.

1.15.1 ByteArray class: instance creation

fromCData: aCObject size: anInteger
Answer a ByteArray containing anInteger bytes starting at the location pointed to by aCObject

1.15.2 ByteArray: basic

= aCollection
Answer whether the receiver's items match those in aCollection

indexOf: anElement startingAt: anIndex
Answer the first index > anIndex which contains anElement. Invoke exceptionBlock and answer its result if no item is found

indexOf: anElement startingAt: anIndex ifAbsent: exceptionBlock
Answer the first index > anIndex which contains anElement. Invoke exceptionBlock and answer its result if no item is found

1.15.3 ByteArray: built ins

asCData: aCType
Allocate memory with malloc for a copy of the receiver, and return it converted to a CObject with the given type

at: anIndex ifAbsent: aBlock
Answer the index-th indexed instance variable of the receiver

byteAt: index
Answer the index-th indexed instance variable of the receiver

byteAt: index put: value
Store the 'value' byte in the index-th indexed instance variable of the receiver

hash Answer an hash value for the receiver

replaceFrom: start to: stop with: aByteArray startingAt: replaceStart
Replace the characters from start to stop with the bytes contained in aByteArray (which, actually, can be any variable byte class), starting at the replaceStart location of aByteArray

replaceFrom: start to: stop withString: aString startingAt: replaceStart
Replace the characters from start to stop with the ASCII codes contained in aString (which, actually, can be any variable byte class), starting at the replaceStart location of aString

1.15.4 ByteArray: CObject

asCData Allocate memory with malloc for a copy of the receiver, and return a pointer to it as a CByte.

castTo: type
Give access to the receiver as a value with the given CType.

1.15.5 ByteArray: converting

asString Answer a String whose character's ASCII codes are the receiver's contents

asUnicodeString
Answer a UnicodeString whose character's codes are the receiver's contents. This is not implemented unless you load the I18N package.

1.15.6 ByteArray: more advanced accessing

charAt: index
Access the C char at the given index in the receiver. The value is returned as a Smalltalk Character. Indices are 1-based just like for other Smalltalk access.

charAt: index put: value
Store as a C char the Smalltalk Character or Integer object identified by ‘value’, at the given index in the receiver, using sizeof(char) bytes - i.e. 1 byte. Indices are 1-based just like for other Smalltalk access.

doubleAt: index
Access the C double at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

doubleAt: index put: value
Store the Smalltalk Float object identified by ‘value’, at the given index in the receiver, writing it like a C double. Indices are 1-based just like for other Smalltalk access.

floatAt: index
Access the C float at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

floatAt: index put: value
Store the Smalltalk Float object identified by ‘value’, at the given index in the receiver, writing it like a C float. Indices are 1-based just like for other Smalltalk access.

intAt: index
Access the C int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

intAt: index put: value
Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(int) bytes. Indices are 1-based just like for other Smalltalk access.

longAt: index
Access the C long int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

longAt: index put: value
Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(long) bytes. Indices are 1-based just like for other Smalltalk access.

longDoubleAt: index
Access the C long double at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

longDoubleAt: index put: value
Store the Smalltalk Float object identified by ‘value’, at the given index in the receiver, writing it like a C double. Indices are 1-based just like for other Smalltalk access.

objectAt: index
Access the Smalltalk object (OOP) at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

objectAt: index put: value

Store a pointer (OOP) to the Smalltalk object identified by ‘value’, at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

shortAt: index

Access the C short int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

shortAt: index put: value

Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(short) bytes. Indices are 1-based just like for other Smalltalk access.

stringAt: index

Access the string pointed by the C ‘char *’ at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

stringAt: index put: value

Store the Smalltalk String object identified by ‘value’, at the given index in the receiver, writing it like a *FRESHLY ALLOCATED* C string. It is the caller’s responsibility to free it if necessary. Indices are 1-based just like for other Smalltalk access.

ucharAt: index

Access the C unsigned char at the given index in the receiver. The value is returned as a Smalltalk Character. Indices are 1-based just like for other Smalltalk access.

ucharAt: index put: value

Store as a C char the Smalltalk Character or Integer object identified by ‘value’, at the given index in the receiver, using sizeof(char) bytes - i.e. 1 byte. Indices are 1-based just like for other Smalltalk access.

uintAt: index

Access the C unsigned int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

uintAt: index put: value

Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(int) bytes. Indices are 1-based just like for other Smalltalk access.

ulongAt: index

Access the C unsigned long int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

ulongAt: index put: value

Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(long) bytes. Indices are 1-based just like for other Smalltalk access.

unsignedCharAt: index
Access the C unsigned char at the given index in the receiver. The value is returned as a Smalltalk Character. Indices are 1-based just like for other Smalltalk access.

unsignedCharAt: index put: value
Store as a C char the Smalltalk Character or Integer object identified by ‘value’, at the given index in the receiver, using sizeof(char) bytes - i.e. 1 byte. Indices are 1-based just like for other Smalltalk access.

unsignedIntAt: index
Access the C unsigned int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

unsignedIntAt: index put: value
Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(int) bytes. Indices are 1-based just like for other Smalltalk access.

unsignedLongAt: index
Access the C unsigned long int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

unsignedLongAt: index put: value
Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(long) bytes. Indices are 1-based just like for other Smalltalk access.

unsignedShortAt: index
Access the C unsigned short int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

unsignedShortAt: index put: value
Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(short) bytes. Indices are 1-based just like for other Smalltalk access.

ushortAt: index
Access the C unsigned short int at the given index in the receiver. Indices are 1-based just like for other Smalltalk access.

ushortAt: index put: value
Store the Smalltalk Integer object identified by ‘value’, at the given index in the receiver, using sizeof(short) bytes. Indices are 1-based just like for other Smalltalk access.

1.15.7 ByteArray: storing

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

storeLiteralOn: aStream
Put a Smalltalk literal evaluating to the receiver on aStream.

storeOn: aStream
Put Smalltalk code evaluating to the receiver on aStream.

1.16 CAggregate

Defined in namespace Smalltalk
Superclass: CObject
Category: Language-C interface

1.16.1 CAggregate class: accessing

alignof Answer the receiver's instances required aligment

sizeof Answer the receiver's instances size

1.16.2 CAggregate: accessing

elementType
Answer the type over which the receiver is constructed.

1.17 CallinProcess

Defined in namespace Smalltalk
Superclass: Process
Category: Language-Processes
I represent a unit of computation for which external C code requested execution, so I must store the returned value once my computation terminates and I must not survive across image saves (since those who invoked me no longer exist). I am otherwise equivalent to a Process.

1.17.1 CallinProcess: debugging

detach Continue running the receiver as a normal Process, and return nil from the callin.

1.18 CArray

Defined in namespace Smalltalk
Superclass: CAggregate
Category: Language-C interface

1.18.1 CArray: accessing

alignof Answer the receiver's required aligment

sizeof Answer the receiver's size

1.19 CArrayCType

Defined in namespace Smalltalk
Superclass: CPtrCType
Category: Language-C interface

1.19.1 CArrayCType class: instance creation

elementType: aCType
This method should not be called for instances of this class.

elementType: aCType numberOfElements: anInteger
Answer a new instance of CPtrCType that maps an array whose elements are of the given CType, and whose size is exactly anInteger elements (of course, anInteger only matters for allocation, not for access, since no out-of-bounds protection is provided for C objects).

from: type
Private - Called by CType>>from: for arrays

1.19.2 CArrayCType: accessing

alignof Answer the alignment of the receiver's instances

numberOfElements
Answer the number of elements in the receiver's instances

sizeof Answer the size of the receiver's instances

1.19.3 CArrayCType: basic

= anObject
Return whether the receiver and anObject are equal.

hash Return a hash code for the receiver.

1.19.4 CArrayCType: storing

storeOn: aStream
As with super.

1.20 CBoolean

Defined in namespace Smalltalk
Superclass: CByte
Category: Language-C interface
I return true if a byte is not zero, false otherwise.

1.20.1 CBoolean class: conversion

type Answer a CType for the receiver

1.20.2 CBoolean: accessing

value Get the receiver's value - answer true if it is != 0, false if it is 0.

value: aBoolean
Set the receiver's value - it's the same as for CBytes, but we get a Boolean, not a Character

1.21 CByte

Defined in namespace Smalltalk
Superclass: CUChar
Category: Language-C interface

You know what a byte is, don't you?!?

1.21.1 CByte class: conversion

cObjStoredType
Nothing special in the default case - answer a CType for the receiver

type Answer a CType for the receiver

1.21.2 CByte: accessing

cObjStoredType
Nothing special in the default case - answer the receiver's CType

value Answer the value the receiver is pointing to. The returned value is a SmallInteger

value: aValue
Set the receiver to point to the value, aValue (a SmallInteger).

1.22 CCallable

Defined in namespace Smalltalk
Superclass: CObject
Category: Language-C interface

I am not part of the Smalltalk definition. My instances contain information about C functions that can be called from within Smalltalk, such as number and type of parameters. This information is used by the C callout mechanism to perform the actual call-out to C routines.

1.22.1 CCallable class: instance creation

for: aCObject returning: returnTypeSymbol withArgs: argsArray
Answer a CFunctionDescriptor with the given address, return type and arguments. The address will be reset to NULL upon image save (and it's the user's task to figure out a way to reinitialize it!)

1.22.2 CCallable: accessing

isValid Answer whether the object represents a valid function.

returnType
Not commented.

1.22.3 CCallable: calling

asyncCall Perform the call-out for the function represented by the receiver. The arguments (and the receiver if one of the arguments has type #self or #selfSmalltalk) are taken from the parent context. Asynchronous call-outs don't return a value, but if the function calls back into Smalltalk the process that started the call-out is not suspended.

asyncCallNoRetryFrom: aContext
Perform the call-out for the function represented by the receiver. The arguments (and the receiver if one of the arguments has type #self or #selfSmalltalk) are taken from the base of the stack of aContext. Asynchronous call-outs don't return a value, but if the function calls back into Smalltalk the process that started the call-out is not suspended. Unlike #asyncCallFrom:, this method does not attempt to find functions in shared objects.

callInto: aValueHolder
Perform the call-out for the function represented by the receiver. The arguments (and the receiver if one of the arguments has type #self or #selfSmalltalk) are taken from the parent context, and the the result is stored into aValueHolder. aValueHolder is also returned.

callNoRetryFrom: aContext into: aValueHolder
Perform the call-out for the function represented by the receiver. The arguments (and the receiver if one of the arguments has type #self or #selfSmalltalk) are taken from the base of the stack of aContext, and the result is stored into aValueHolder. aValueHolder is also returned. Unlike #callFrom:into:, this method does not attempt to find functions in shared objects.

1.22.4 CCallable: restoring

link Rebuild the object after the image is restarted.

1.23 CCallbackDescriptor

Defined in namespace Smalltalk
Superclass: CCallable
Category: Language-C interface
I am not part of the Smalltalk definition. My instances are able to convert blocks into C functions that can be passed to C.

1.23.1 CCallbackDescriptor class: instance creation

for: aBlock returning: returnTypeSymbol withArgs: argsArray
Answer a CCallbackDescriptor with the given block, return type and arguments.

1.23.2 CCallbackDescriptor: accessing

block Answer the block of the function represented by the receiver.

block: aBlock
Set the block of the function represented by the receiver.

1.23.3 CCallbackDescriptor: restoring

link Make the address of the function point to the registered address.

1.24 CChar

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.24.1 CChar class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.24.2 CChar: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.24.3 CChar: conversion

asByteArray: size
Convert size bytes pointed to by the receiver to a String

asString Convert the data pointed to by the receiver, up to the first NULL byte, to a String

asString: size
Convert size bytes pointed to by the receiver to a String

1.25 CCompound

Defined in namespace Smalltalk
Superclass: CObject
Category: Language-C interface

1.25.1 CCompound class: instance creation

gcNew Allocate a new instance of the receiver, backed by garbage-collected storage.

new Allocate a new instance of the receiver. To free the memory after GC, remember to call #addToBeFinalized.

1.25.2 CCompound class: subclass creation

alignof Answer 1, the alignment of an empty struct

classPragmas
Return the pragmas that are written in the file-out of this class.

compileSize: size align: alignment
Private - Compile sizeof and alignof methods

declaration
Return the description of the fields in the receiver class.

declaration: array
This method's functionality should be implemented by subclasses of CCompound

declaration: array inject: startOffset into: aBlock
Compile methods that implement the declaration in array. To compute the offset after each field, the value of the old offset plus the new field's size is passed to aBlock, together with the new field's alignment requirements.

emitFieldNameTo: str for: name
Private - Emit onto the given stream the code for adding the given selector to the CCompound's #examineOn: method.

newStruct: structName declaration: array
The old way to create a CStruct. Superseded by #subclass:declaration:...

sizeof Answer 0, the size of an empty struct

subclass: structName declaration: array classVariableNames: cvn poolDictionaries: pd category: category
Create a new class with the given name that contains code to implement the given C struct. All the parameters except 'array' are the same as for a standard class creation message; see documentation for more information

1.25.3 CCompound: debugging

examineOn: aStream
Print the contents of the receiver's fields on aStream

fieldSelectorList
Answer a list of selectors whose return values should be printed by #examineOn:.

1.26 CDouble

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.26.1 CDouble class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.26.2 CDouble: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.27 CFloat

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.27.1 CFloat class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.27.2 CFloat: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.28 CFunctionDescriptor

Defined in namespace Smalltalk
Superclass: CCallable
Category: Language-C interface
I am not part of the Smalltalk definition. My instances contain information about C functions that can be called from within Smalltalk, such as number and type of parameters. This information is used by the C callout mechanism to perform the actual call-out to C routines.

1.28.1 CFunctionDescriptor class: instance creation

for: funcName returning: returnTypeSymbol withArgs: argsArray
Answer a CFunctionDescriptor with the given function name, return type and arguments. funcName must be a String.

1.28.2 CFunctionDescriptor class: testing

addressOf: function
Answer whether a function is registered (on the C side) with the given name or is dynamically loadable.

isFunction: function
Answer whether a function is registered (on the C side) with the given name.

1.28.3 CFunctionDescriptor: accessing

name Answer the name of the function (on the C side) represented by the receiver

name: aString
Set the name of the function (on the C side) represented by the receiver

1.28.4 CFunctionDescriptor: printing

printOn: aStream
Print a representation of the receiver onto aStream

1.28.5 CFunctionDescriptor: restoring

link Make the address of the function point to the registered address.

1.29 Character

Defined in namespace Smalltalk
Superclass: Magnitude
Category: Language-Data types

My instances represent the 256 characters of the character set. I provide messages to translate between integers and character objects, and provide names for some of the common unprintable characters.

Character is always used (mostly for performance reasons) when referring to characters whose code point is between 0 and 127. Above 127, instead, more care is needed: Character refers to bytes that are used as part of encoding of a character, while UnicodeCharacter refers to the character itself.

1.29.1 Character class: built ins

asciiValue: anInteger
Returns the character object corresponding to anInteger. Error if anInteger is not an integer, or not in 0..127.

codePoint: anInteger
Returns the character object, possibly an UnicodeCharacter, corresponding to anInteger. Error if anInteger is not an integer, or not in 0..16r10FFFF.

value: anInteger
Returns the character object corresponding to anInteger. Error if anInteger is not an integer, or not in 0..255.

1.29.2 Character class: constants

backspace Returns the character 'backspace'

bell Returns the character 'bel'

cr Returns the character 'cr'

eof Returns the character 'eof', also known as 'sub'

eot Returns the character 'eot', also known as 'Ctrl-D'

esc Returns the character 'esc'

ff Returns the character 'ff', also known as 'newPage'

lf Returns the character 'lf', also known as 'nl'

newPage Returns the character 'newPage', also known as 'ff'

nl Returns the character 'nl', also known as 'lf'

nul Returns the character 'nul'

space Returns the character 'space'

tab Returns the character 'tab'

1.29.3 Character class: initializing lookup tables

initialize Initialize the lookup table which is used to make case and digit-to-char conversions faster. Indices in Table are ASCII values incremented by one. Indices 1-256 classify chars (0 = nothing special, 2 = separator, 48 = digit, 55 = uppercase, 3 = lowercase), indices 257-512 map to lowercase chars, indices 513-768 map to uppercase chars.

1.29.4 Character class: instance creation

digitValue: anInteger
Returns a character that corresponds to anInteger. 0-9 map to $0-$9, 10-35 map to $A-$Z

1.29.5 Character class: testing

isImmediate
Answer whether, if x is an instance of the receiver, x copy == x

1.29.6 Character: built ins

= char Boolean return value; true if the characters are equal

asInteger Returns the integer value corresponding to self. #codePoint, #asciiValue, #value, and #asInteger are synonyms.

asciiValue Returns the integer value corresponding to self. #codePoint, #asciiValue, #value, and #asInteger are synonyms.

codePoint Returns the integer value corresponding to self. #codePoint, #asciiValue, #value, and #asInteger are synonyms.

value Returns the integer value corresponding to self. #codePoint, #asciiValue, #value, and #asInteger are synonyms.

1.29.7 Character: coercion methods

*** aNumber**
Returns a String with aNumber occurrences of the receiver.

asLowercase
Returns self as a lowercase character if it's an uppercase letter, otherwise returns the character unchanged.

asString Returns the character self as a string. Only valid if the character is between 0 and 255.

asSymbol Returns the character self as a symbol.

asUnicodeString
Returns the character self as a Unicode string.

asUppercase
Returns self as a uppercase character if it's an lowercase letter, otherwise returns the character unchanged.

1.29.8 Character: comparing

< aCharacter
Compare the character's ASCII value. Answer whether the receiver's is the least.

<= aCharacter
Compare the character's ASCII value. Answer whether the receiver's is the least or their equal.

> aCharacter
Compare the character's ASCII value. Answer whether the receiver's is the greatest.

>= aCharacter
Compare the character's ASCII value. Answer whether the receiver's is the greatest or their equal.

1.29.9 Character: converting

asCharacter
Return the receiver, since it is already a character.

digitValue Returns the value of self interpreted as a digit. Here, 'digit' means either 0-9, or A-Z, which maps to 10-35.

1.29.10 Character: printing

displayOn: aStream
Print a representation of the receiver on aStream. Unlike #printOn:, this method strips the leading dollar.

printOn: aStream
Print a representation of the receiver on aStream

storeLiteralOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

1.29.11 Character: storing

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.29.12 Character: testing

isAlphaNumeric
True if self is a letter or a digit

isDigit
True if self is a 0-9 digit

isDigit: radix
Answer whether the receiver is a valid character in the given radix.

isLetter
True if self is an upper- or lowercase letter

isLowercase
True if self is a lowercase letter

isPathSeparator
Returns true if self is a path separator ($/ or $\ under Windows, $/ only under Unix systems including Mac OS X).

isPunctuation
Returns true if self is one of '.,:;!?'

isSeparator
Returns true if self is a space, cr, tab, nl, or newPage

isUppercase
True if self is uppercase

isVowel
Returns true if self is a, e, i, o, or u; case insensitive

1.29.13 Character: testing functionality

isCharacter
Answer True. We're definitely characters

1.30 CharacterArray

Defined in namespace Smalltalk
Superclass: ArrayedCollection
Category: Collections-Text
My instances represent a generic textual (string) data type. I provide accessing and manipulation methods for strings.

1.30.1 CharacterArray class: basic

fromString: aCharacterArray
Make up an instance of the receiver containing the same characters as aCharacterArray, and answer it.

lineDelimiter
Answer a CharacterArray which one can use as a line delimiter. This is meant to be used on subclasses of CharacterArray.

1.30.2 CharacterArray class: multibyte encodings

isUnicode Answer whether the receiver stores bytes (i.e. an encoded form) or characters (if true is returned).

1.30.3 CharacterArray: basic

valueAt: anIndex ifAbsent: aBlock
Answer the ascii value of the anIndex-th character of the receiver, or evaluate aBlock and answer the result if the index is out of range.

1.30.4 CharacterArray: built ins

valueAt: index
Answer the ascii value of index-th character variable of the receiver

valueAt: index put: value
Store (Character value: value) in the index-th indexed instance variable of the receiver

1.30.5 CharacterArray: comparing

< aCharacterArray
Return true if the receiver is less than aCharacterArray, ignoring case differences.

<= aCharacterArray
Returns true if the receiver is less than or equal to aCharacterArray, ignoring case differences. If is receiver is an initial substring of aCharacterArray, it is considered to be less than aCharacterArray.

= aString Answer whether the receiver's items match those in aCollection

> aCharacterArray
Return true if the receiver is greater than aCharacterArray, ignoring case differences.

>= aCharacterArray
Returns true if the receiver is greater than or equal to aCharacterArray, ignoring case differences. If is aCharacterArray is an initial substring of the receiver, it is considered to be less than the receiver.

indexOf: aCharacterArray matchCase: aBoolean startingAt: anIndex
Answer an Interval of indices in the receiver which match the aCharacterArray pattern. # in aCharacterArray means 'match any character', * in aCharacterArray means 'match any sequence of characters'. The first item of the returned interval is >= anIndex. If aBoolean is false, the search is case-insensitive, else it is case-sensitive. If no Interval matches the pattern, answer nil.

match: aCharacterArray
Answer whether aCharacterArray matches the pattern contained in the receiver. # in the receiver means 'match any character', * in receiver means 'match any sequence of characters'.

match: aCharacterArray ignoreCase: aBoolean
Answer whether aCharacterArray matches the pattern contained in the receiver. # in the receiver means 'match any character', * in receiver means 'match any sequence of characters'. The case of alphabetic characters is ignored if aBoolean is true.

sameAs: aCharacterArray
Returns true if the receiver is the same CharacterArray as aCharacterArray, ignoring case differences.

1.30.6 CharacterArray: converting

asByteArray
Return the receiver, converted to a ByteArray of ASCII values

asClassPoolKey
Return the receiver, ready to be put in a class pool dictionary

asGlobalKey
Return the receiver, ready to be put in the Smalltalk dictionary

asInteger Parse an Integer number from the receiver until the input character is invalid and answer the result at this point

asLowercase
Returns a copy of self as a lowercase CharacterArray

asNumber Parse a Number from the receiver until the input character is invalid and answer the result at this point

asPoolKey Return the receiver, ready to be put in a pool dictionary

asString But I already am a String! Really!

asSymbol Returns the symbol corresponding to the CharacterArray

asUnicodeString
Answer a UnicodeString whose character's codes are the receiver's contents This is not implemented unless you load the I18N package.

asUppercase
Returns a copy of self as an uppercase CharacterArray

fileName But I don't HAVE a file name!

filePos But I don't HAVE a file position!

isNumeric Answer whether the receiver denotes a number

trimSeparators
Return a copy of the reciever without any spaces on front or back. The implementation is protected against the 'all blanks' case.

1.30.7 CharacterArray: multibyte encodings

encoding
Answer the encoding used by the receiver.

isUnicode
Answer whether the receiver stores bytes (i.e. an encoded form) or characters (if true is returned).

numberOfCharacters
Answer the number of Unicode characters in the receiver. This is not implemented unless you load the I18N package.

1.30.8 CharacterArray: still unclassified

withUnixShellEscapes
Answer the receiver with special shell characters converted to a backslash sequence.

withWindowsShellEscapes
Answer the receiver with Windows shell characters escaped properly.

1.30.9 CharacterArray: string processing

% aCollection
Answer the receiver with special escape sequences replaced by elements of aCollection. %n (1<=n<=9, A<=n<=Z) are replaced by the n-th element of aCollection (A being the 10-th element and so on until the 35th). %(string) sequences are accessed as strings, which makes sense only if aCollection is a Dictionary or LookupTable. In addition, the special pattern %<trueString|falseString>n or %<trueString|falseString>(string) is replaced with one of the two strings depending on the element of aCollection being true or false. The replaced elements are 'displayed' (i.e. their displayString is used).

bindWith: s1
Answer the receiver with every %1 replaced by the displayString of s1

bindWith: s1 with: s2
Answer the receiver with every %1 or %2 replaced by s1 or s2, respectively. s1 and s2 are 'displayed' (i.e. their displayString is used) upon replacement.

bindWith: s1 with: s2 with: s3
Answer the receiver with every %1, %2 or %3 replaced by s1, s2 or s3, respectively. s1, s2 and s3 are 'displayed' (i.e. their displayString is used) upon replacement.

bindWith: s1 with: s2 with: s3 with: s4
Answer the receiver with every %1, %2, %3 or %4 replaced by s1, s2, s3 or s4, respectively. s1, s2, s3 and s4 are 'displayed' (i.e. their displayString is used) upon replacement.

bindWithArguments: aCollection
Answer the receiver with special escape sequences replaced by elements of aCollection. %n (1<=n<=9, A<=n<=Z) are replaced by the n-th element of aCollection (A being the 10-th element and so on until the 35th). %(string) sequences

are accessed as strings, which makes sense only if aCollection is a Dictionary or LookupTable. In addition, the special pattern %<trueString|falseString>n or %<trueString|falseString>(string) is replaced with one of the two strings depending on the element of aCollection being true or false. The replaced elements are 'displayed' (i.e. their displayString is used).

contractTo: smallSize
Either return myself, or a copy shortened to smallSize characters by inserting an ellipsis (three dots: ...)

lines Answer an Array of Strings each representing one line in the receiver.

linesDo: aBlock
Evaluate aBlock once for every newline delimited line in the receiver, passing the line to the block.

subStrings Answer an OrderedCollection of substrings of the receiver. A new substring start at the start of the receiver, or after every sequence of white space characters

subStrings: sep
Answer an OrderedCollection of substrings of the receiver. A new substring start at the start of the receiver, or after every occurrence of one of the characters in sep

substrings Answer an OrderedCollection of substrings of the receiver. A new substring start at the start of the receiver, or after every sequence of white space characters. This message is preserved for backwards compatibility; the ANSI standard mandates 'subStrings', with an uppercase s.

substrings: sep
Answer an OrderedCollection of substrings of the receiver. A new substring start at the start of the receiver, or after every occurrence of one of the characters in sep. This message is preserved for backwards compatibility; the ANSI standard mandates 'subStrings:', with an uppercase s.

withShellEscapes
Answer the receiver with special shell characters converted to a backslash sequence.

1.30.10 CharacterArray: testing functionality

isCharacterArray
Answer 'true'.

1.31 CInt

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.31.1 CInt class: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's size

1.31.2 CInt: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's instances size

1.32 Class

Defined in namespace Smalltalk
Superclass: ClassDescription
Category: Language-Implementation
I am THE class object. My instances are the classes of the system. I provide information commonly attributed to classes: namely, the class name, class comment (you wouldn't be reading this if it weren't for me), a list of the instance variables of the class, and the class category.

1.32.1 Class class: initialize

initialize Perform the special initialization of root classes.

1.32.2 Class: accessing instances and variables

addClassVarName: aString
Add a class variable with the given name to the class pool dictionary.

addClassVarName: aString value: valueBlock
Add a class variable with the given name to the class pool dictionary, and evaluate valueBlock as its initializer.

addSharedPool: aDictionary
Add the given shared pool to the list of the class' pool dictionaries

allClassVarNames
Answer the names of the variables in the receiver's class pool dictionary and in each of the superclasses' class pool dictionaries

bindingFor: aString
Answer the variable binding for the class variable with the given name

category Answer the class category

category: aString
Change the class category to aString

classPool Answer the class pool dictionary

classPragmas
Return the pragmas that are written in the file-out of this class.

classVarNames
Answer the names of the variables in the class pool dictionary

comment Answer the class comment

comment: aString
Change the class name

environment
Answer 'environment'.

environment: aNamespace
Set the receiver's environment to aNamespace and recompile everything

initialize redefined in children (?)

initializeAsRootClass
Perform special initialization reserved to root classes.

name Answer the class name

removeClassVarName: aString
Removes the class variable from the class, error if not present, or still in use.

removeSharedPool: aDictionary
Remove the given dictionary to the list of the class' pool dictionaries

sharedPools
Return the names of the shared pools defined by the class

superclass: aClass
Set the receiver's superclass.

1.32.3 Class: filing

fileOutDeclarationOn: aFileStream
File out class definition to aFileStream. Requires package Parser.

fileOutOn: aFileStream
File out complete class description: class definition, class and instance methods. Requires package Parser.

1.32.4 Class: instance creation

extend Redefine a version of the receiver in the current namespace. Note: this method can bite you in various ways when sent to system classes; read the section on namespaces in the manual for some examples of the problems you can encounter.

inheritShape
Answer whether subclasses will have by default the same shape as this class. The default is false.

subclass: classNameString

Define a subclass of the receiver with the given name. If the class is already defined, don't modify its instance or class variables but still, if necessary, recompile everything needed.

subclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString

Define a fixed subclass of the receiver with the given name, instance variables, class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed.

variable: shape subclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString

Define a variable subclass of the receiver with the given name, shape, instance variables, class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed. The shape can be one of #byte #int8 #character #short #ushort #int #uint #int64 #uint64 #utf32 #float #double or #pointer.

variableByteSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString

Define a byte variable subclass of the receiver with the given name, instance variables (must be ''), class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed.

variableSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString

Define a variable pointer subclass of the receiver with the given name, instance variables, class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed.

variableWordSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString

Define a word variable subclass of the receiver with the given name, instance variables (must be ''), class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed.

1.32.5 Class: instance creation - alternative

categoriesFor: method are: categories

Don't use this, it is only present to file in from IBM Smalltalk

subclass: classNameString classInstanceVariableNames: stringClassInstVarNames instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames

Don't use this, it is only present to file in from IBM Smalltalk

subclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableByteSubclass: classNameString classInstanceVariableNames: stringClassInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableByteSubclass: classNameString classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableLongSubclass: classNameString classInstanceVariableNames: stringClassInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableLongSubclass: classNameString classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableSubclass: classNameString classInstanceVariableNames: stringClassInstVarNames instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

1.32.6 Class: pragmas

pragmaHandlerFor: aSymbol
Answer the (possibly inherited) registered handler for pragma aSymbol, or nil if not found.

registerHandler: aBlock forPragma: pragma
While compiling methods, on every encounter of the pragma with the given name, call aBlock with the CompiledMethod and an array of pragma argument values.

1.32.7 Class: printing

article Answer an article ('a' or 'an') which is ok for the receiver's name

printOn: aStream
Print a representation of the receiver on aStream

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.32.8 Class: saving and loading

binaryRepresentationVersion
Answer a number >= 0 which represents the current version of the object's representation. The default implementation answers zero.

convertFromVersion: version withFixedVariables: fixed indexedVariables: indexed for: anObjectDumper
This method is called if a VersionableObjectProxy is attached to a class. It receives the version number that was stored for the object (or nil if the object did not use a VersionableObjectProxy), the fixed instance variables, the indexed instance variables, and the ObjectDumper that has read the object. The default implementation ignores the version and simply fills in an instance of the receiver with the given fixed and indexed instance variables (nil if the class instances are of fixed size). If instance variables were removed from the class, extras are ignored; if the class is now fixed and used to be indexed, indexed is not used.

nonVersionedInstSize
Answer the number of instance variables that the class used to have when objects were stored without using a VersionableObjectProxy. The default implementation answers the current instSize.

1.32.9 Class: security

check: aPermission
Not commented.

securityPolicy
Answer 'securityPolicy'.

securityPolicy: aSecurityPolicy
Not commented.

1.32.10 Class: still unclassified

allSharedPoolDictionariesDo: aBlock
Answer the shared pools visible from methods in the metaclass, in the correct search order.

fileOutHeaderOn: aFileStream
Not commented.

1.32.11 Class: testing

= aClass Returns true if the two class objects are to be considered equal.

1.32.12 Class: testing functionality

asClass Answer the receiver.

isClass Answer 'true'.

1.33 ClassDescription

Defined in namespace Smalltalk
Superclass: Behavior
Category: Language-Implementation

My instances provide methods that access classes by category, and allow whole categories of classes to be filed out to external disk files.

1.33.1 ClassDescription: compiling

compile: code classified: categoryName
Compile code in the receiver, assigning the method to the given category. Answer the newly created CompiledMethod, or nil if an error was found.

compile: code classified: categoryName ifError: block
Compile method source and install in method category, categoryName. If there are parsing errors, invoke exception block, 'block' (see compile:ifError:). Return the method

compile: code classified: categoryName notifying: requestor
Compile method source and install in method category, categoryName. If there are parsing errors, send an error message to requestor

1.33.2 ClassDescription: conversion

asClass
This method's functionality should be implemented by subclasses of ClassDescription

asMetaclass
Answer the metaclass associated to the receiver

binding
Answer a VariableBinding object whose value is the receiver

1.33.3 ClassDescription: copying

copy: selector from: aClass
Copy the given selector from aClass, assigning it the same category

copy: selector from: aClass classified: categoryName
Copy the given selector from aClass, assigning it the given category

copyAll: arrayOfSelectors from: class
Copy all the selectors in arrayOfSelectors from class, assigning them the same category they have in class

copyAll: arrayOfSelectors from: class classified: categoryName
Copy all the selectors in arrayOfSelectors from aClass, assigning them the given category

copyAllCategoriesFrom: aClass
Copy all the selectors in aClass, assigning them the original category

copyCategory: categoryName from: aClass
Copy all the selectors in from aClass that belong to the given category

copyCategory: categoryName from: aClass classified: newCategoryName
: Copy all the selectors in from aClass that belong to the given category, reclassifying them as belonging to the given category

1.33.4 ClassDescription: filing

fileOut: fileName
: Open the given file and to file out a complete class description to it. Requires package Parser.

fileOutCategory: categoryName to: fileName
: File out all the methods belonging to the method category, categoryName, to the fileName file. Requires package Parser.

fileOutOn: aFileStream
: File out complete class description: class definition, class and instance methods. Requires package Parser.

fileOutSelector: selector to: fileName
: File out the given selector to fileName. Requires package Parser.

1.33.5 ClassDescription: organization of messages and classes

classify: aSelector under: aString
: Put the method identified by the selector aSelector under the category given by aString.

createGetMethod: what
: Create a method accessing the variable 'what'.

createGetMethod: what default: value
: Create a method accessing the variable 'what', with a default value of 'value', using lazy initialization

createSetMethod: what
: Create a method which sets the variable 'what'.

defineAsyncCFunc: cFuncNameString withSelectorArgs: selectorAndArgs args: argsArray
: See documentation. This function is deprecated, you should use the <asyncCCall: ... > special syntax instead.

defineCFunc: cFuncNameString withSelectorArgs: selectorAndArgs returning: returnTypeSymbol args: argsArray
: See documentation. This function is deprecated, you should use the <asyncCCall: ... > special syntax instead.

removeCategory: aString
: Remove from the receiver every method belonging to the given category

whichCategoryIncludesSelector: selector
: Answer the category for the given selector, or nil if the selector is not found

1.33.6 ClassDescription: parsing class declarations

addSharedPool: aDictionary
Add the given shared pool to the list of the class' pool dictionaries

import: aDictionary
Add the given shared pool to the list of the class' pool dictionaries

1.33.7 ClassDescription: printing

classVariableString
This method's functionality should be implemented by subclasses of ClassDescription

instanceVariableString
Answer a string containing the name of the receiver's instance variables.

nameIn: aNamespace
Answer the class name when the class is referenced from aNamespace

printOn: aStream in: aNamespace
Print on aStream the class name when the class is referenced from aNamespace

sharedVariableString
This method's functionality should be implemented by subclasses of ClassDescription

1.33.8 ClassDescription: still unclassified

fileOutCategory: category toStream: aFileStream
File out all the methods belonging to the method category, category, to aFileStream. Requires package Parser.

fileOutSelector: aSymbol toStream: aFileStream
File out all the methods belonging to the method category, category, to aFileStream. Requires package Parser.

1.34 CLong

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.34.1 CLong class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.34.2 CLong: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.35 CLongDouble

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.35.1 CLongDouble class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.35.2 CLongDouble: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.36 CLongLong

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.36.1 CLongLong class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.36.2 CLongLong: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.37 CObject

Defined in namespace Smalltalk
Superclass: Object
Category: Language-C interface

I am not part of the standard Smalltalk kernel class hierarchy. My instances contain values that are not interpreted by the Smalltalk system; they frequently hold "pointers" to data outside of the Smalltalk environment. The C callout mechanism allows my instances to be transformed into their corresponding C values for use in external routines.

1.37.1 CObject class: conversion

type Nothing special in the default case - answer a CType for the receiver

1.37.2 CObject class: instance creation

address: anInteger
Answer a new object pointing to the passed address, anInteger

alloc: nBytes
Allocate nBytes bytes and return an instance of the receiver

gcAlloc: nBytes
Allocate nBytes bytes and return an instance of the receiver

gcNew: nBytes
Allocate nBytes bytes and return an instance of the receiver

new Answer a new object pointing to NULL.

new: nBytes
Allocate nBytes bytes and return an instance of the receiver

1.37.3 CObject class: primitive allocation

alloc: nBytes type: cTypeObject
Allocate nBytes bytes and return a CObject of the given type

gcAlloc: nBytes type: cTypeObject
Allocate nBytes bytes and return a CObject of the given type

1.37.4 CObject class: subclass creation

inheritShape
Answer whether subclasses will have by default the same shape as this class. The default is true for the CObject hierarchy.

1.37.5 CObject: accessing

address Answer the address the receiver is pointing to. The address can be absolute if the storage is nil, or relative to the Smalltalk object in #storage. In this case, an address of 0 corresponds to the first instance variable.

address: anInteger
Set the receiver to point to the passed address, anInteger

isAbsolute Answer whether the object points into a garbage-collected Smalltalk storage, or it is an absolute address.

printOn: aStream
Print a representation of the receiver

storage Answer the storage that the receiver is pointing into, or nil if the address is absolute.

storage: anObject
Change the receiver to point to the storage of anObject.

type: aCType
Set the receiver's type to aCType.

1.37.6 CObject: basic

= anObject
Return true if the receiver and aCObject are equal.

hash Return a hash value for anObject.

1.37.7 CObject: C data access

at: byteOffset put: aValue type: aType
Store aValue as data of the given type from byteOffset bytes after the pointer stored in the receiver

at: byteOffset type: aType
Answer some data of the given type from byteOffset bytes after the pointer stored in the receiver

free Free the receiver's pointer and set it to null. Big trouble hits you if the receiver doesn't point to the base of a malloc-ed area.

1.37.8 CObject: conversion

castTo: aType
Answer another CObject, pointing to the same address as the receiver, but belonging to the aType CType.

narrow This method is called on CObjects returned by a C call-out whose return type is specified as a CType; it mostly allows one to change the class of the returned CObject. By default it does nothing, and that's why it is not called when #cObject is used to specify the return type.

type Answer a CType for the receiver

1.37.9 CObject: finalization

finalize To make the VM call this, use #addToBeFinalized. It frees automatically any memory pointed to by the CObject. It is not automatically enabled because big trouble hits you if you use #free and the receiver doesn't point to the base of a malloc-ed area.

1.37.10 CObject: pointer-like behavior

+ anInteger
Return another instance of the receiver's class which points at &receiver[anInteger] (or, if you prefer, what 'receiver + anInteger' does in C).

- intOrPtr If intOrPtr is an integer, return another instance of the receiver's class pointing at &receiver[-anInteger] (or, if you prefer, what 'receiver - anInteger' does in C). If it is the same class as the receiver, return the difference in chars, i.e. in bytes, between the two pointed addresses (or, if you prefer, what 'receiver - anotherCharPtr' does in C)

addressAt: anIndex
Return a new CObject of the element type, corresponding to an object that is anIndex places past the receiver (remember that CObjects represent pointers and that C pointers behave like arrays). anIndex is zero-based, just like with all other C-style accessing.

at: anIndex
Dereference a pointer that is anIndex places past the receiver (remember that CObjects represent pointers and that C pointers behave like arrays). anIndex is zero-based, just like with all other C-style accessing.

at: anIndex put: aValue
Store anIndex places past the receiver the passed Smalltalk object or CObject 'aValue'; if it is a CObject is dereferenced: that is, this method is equivalent either to cobj[anIndex]=aValue or cobj[anIndex]=*aValue. anIndex is zero-based, just like with all other C-style accessing.

In both cases, aValue should be of the element type or of the corresponding Smalltalk type (that is, a String is ok for an array of CStrings) to avoid typing problems which however will not be signaled because C is untyped.

decr Adjust the pointer by sizeof(dereferencedType) bytes down (i.e. –receiver)

decrBy: anInteger
Adjust the pointer by anInteger elements down (i.e. receiver -= anInteger)

incr Adjust the pointer by sizeof(dereferencedType) bytes up (i.e. ++receiver)

incrBy: anInteger
Adjust the pointer by anInteger elements up (i.e. receiver += anInteger)

1.37.11 CObject: testing

isNull Return true if the receiver points to NULL.

1.37.12 CObject: testing functionality

isCObject Answer 'true'.

1.38 Collection

Defined in namespace Smalltalk
Superclass: Iterable
Category: Collections

I am an abstract class. My instances are collections of objects. My subclasses may place some restrictions or add some definitions to how the objects are stored or organized; I say nothing about this. I merely provide some object creation and access routines for general collections of objects.

1.38.1 Collection class: instance creation

from: anArray
Convert anArray to an instance of the receiver. anArray is structured such that the instance can be conveniently and fully specified using brace-syntax, possibly by imposing some additional structure on anArray.

join: aCollection
Answer a collection formed by treating each element in aCollection as a 'withAll:' argument collection to be added to a new instance.

with: anObject
Answer a collection whose only element is anObject

with: firstObject with: secondObject
Answer a collection whose only elements are the parameters in the order they were passed

with: firstObject with: secondObject with: thirdObject
Answer a collection whose only elements are the parameters in the order they were passed

with: firstObject with: secondObject with: thirdObject with: fourthObject
Answer a collection whose only elements are the parameters in the order they were passed

with: firstObject with: secondObject with: thirdObject with: fourthObject with: fifthObject
Answer a collection whose only elements are the parameters in the order they were passed

withAll: aCollection
Answer a collection whose elements are all those in aCollection

1.38.2 Collection class: multibyte encodings

isUnicode Answer true; the receiver is able to store arbitrary Unicode characters.

1.38.3 Collection: adding

add: newObject
Add newObject to the receiver, answer it

addAll: aCollection
Adds all the elements of 'aCollection' to the receiver, answer aCollection

1.38.4 Collection: compiler

literalEquals: anObject
Not commented.

literalHash
Not commented.

1.38.5 Collection: concatenating

join Answer a new collection like my first element, with all the elements (in order) of all my elements, which should be collections.

I use my first element instead of myself as a prototype because my elements are more likely to share the desired properties than I am, such as in:

#('hello, ' 'world') join => 'hello, world'

1.38.6 Collection: converting

asArray Answer an Array containing all the elements in the receiver

asBag Answer a Bag containing all the elements in the receiver

asByteArray
Answer a ByteArray containing all the elements in the receiver

asOrderedCollection
Answer an OrderedCollection containing all the elements in the receiver

asRunArray
Answer the receiver converted to a RunArray. If the receiver is not ordered the order of the elements in the RunArray might not be the #do: order.

asSet Answer a Set containing all the elements in the receiver with no duplicates

asSortedCollection
Answer a SortedCollection containing all the elements in the receiver with the default sort block - [:a :b | a <= b]

asSortedCollection: aBlock
Answer a SortedCollection whose elements are the elements of the receiver, sorted according to the sort block aBlock

asString Answer a String containing all the elements in the receiver

asUnicodeString
Answer a UnicodeString containing all the elements in the receiver

1.38.7 Collection: copying Collections

copyReplacing: targetObject withObject: newObject
Copy replacing each object which is = to targetObject with newObject

copyWith: newElement
Answer a copy of the receiver to which newElement is added

copyWithout: oldElement
Answer a copy of the receiver to which all occurrences of oldElement are removed

1.38.8 Collection: copying SequenceableCollections

, anIterable
Append anIterable at the end of a copy of the receiver (using #add:), and answer a new collection

1.38.9 Collection: enumeration

anyOne Answer an unspecified element of the collection.

beConsistent
This method is private, but it is quite interesting so it is documented. It ensures that a collection is in a consistent state before attempting to iterate on it; its presence reduces the number of overrides needed by collections who try to amortize their execution times. The default implementation does nothing, so it is optimized out by the virtual machine and so it loses very little on the performance side. Note that descendants of Collection have to call it explicitly since #do: is abstract in Collection.

collect: aBlock
Answer a new instance of a Collection containing all the results of evaluating aBlock passing each of the receiver's elements

gather: aBlock
Answer a new instance of a Collection containing all the results of evaluating aBlock, joined together. aBlock should return collections. The result is the same kind as the first collection, returned by aBlock (as for #join).

readStream
Answer a stream that gives elements of the receiver

reject: aBlock
Answer a new instance of a Collection containing all the elements in the receiver which, when passed to aBlock, don't answer true

select: aBlock
Answer a new instance of a Collection containing all the elements in the receiver which, when passed to aBlock, answer true

1.38.10 Collection: finalization

mourn: anObject
Private - anObject has been found to have a weak key, remove it and possibly finalize the key.

1.38.11 Collection: printing

displayLines
Print each element of the receiver to a line on standard output.

examineOn: aStream
Print all the instance variables and objects in the receiver on aStream

printOn: aStream
Print a representation of the receiver on aStream

1.38.12 Collection: removing

empty Remove everything from the receiver.

remove: oldObject
Remove oldObject from the receiver. If absent, fail, else answer oldObject.

remove: oldObject ifAbsent: anExceptionBlock
Remove oldObject from the receiver. If absent, evaluate anExceptionBlock and answer the result, else answer oldObject.

removeAll: aCollection
Remove each object in aCollection, answer aCollection, fail if some of them is absent. Warning: this could leave the collection in a semi-updated state.

removeAll: aCollection ifAbsent: aBlock
Remove each object in aCollection, answer aCollection; if some element is absent, pass it to aBlock.

removeAllSuchThat: aBlock
Remove from the receiver all objects for which aBlock returns true.

1.38.13 Collection: sorting

sorted Return a sequenceable collection with the contents of the receiver sorted according to the default sort block, which uses #<= to compare items.

sorted: sortBlock
Return a sequenceable collection with the contents of the receiver sorted according to the given sort block, which accepts pair of items and returns true if the first item is less than the second one.

1.38.14 Collection: storing

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.38.15 Collection: testing collections

capacity Answer how many elements the receiver can hold before having to grow.

identityIncludes: anObject
Answer whether we include the anObject object

includes: anObject
Answer whether we include anObject

includesAllOf: aCollection
Answer whether we include all of the objects in aCollection

includesAnyOf: aCollection
Answer whether we include any of the objects in aCollection

isEmpty Answer whether we are (still) empty

isSequenceable
Answer whether the receiver can be accessed by a numeric index with #at:/-#at:put:.

notEmpty Answer whether we include at least one object

occurrencesOf: anObject
Answer how many occurrences of anObject we include

size Answer how many objects we include

1.39 CompiledBlock

Defined in namespace Smalltalk
Superclass: CompiledCode
Category: Language-Implementation
I represent a block that has been compiled.

1.39.1 CompiledBlock class: instance creation

new: numBytecodes header: anInteger method: outerMethod
Answer a new instance of the receiver with room for the given number of bytecodes and the given header.

numArgs: args numTemps: temps bytecodes: bytecodes depth: depth literals: literalArray
Answer an (almost) full fledged CompiledBlock. To make it complete, you must either set the new object's 'method' variable, or put it into a BlockClosure and put the BlockClosure into a CompiledMethod's literals. The clean-ness of the block is automatically computed.

1.39.2 CompiledBlock: accessing

flags Answer the 'cleanness' of the block. 0 = clean; 1 = access to receiver variables and/or self; 2-30 = access to variables that are 1-29 contexts away; 31 = return from method or push thisContext

method Answer the CompiledMethod in which the receiver lies

methodClass
Answer the class in which the receiver is installed.

methodClass: methodClass
Set the receiver's class instance variable

numArgs Answer the number of arguments passed to the receiver

numLiterals
Answer the number of literals for the receiver

numTemps
Answer the number of temporary variables used by the receiver

selector Answer the selector through which the method is called

selector: aSymbol
Set the selector through which the method is called

sourceCodeLinesDelta
Answer the delta from the numbers in LINE_NUMBER bytecodes to source code line numbers.

sourceCodeMap
Answer an array which maps bytecode indices to source code line numbers. 0 values represent invalid instruction pointer indices.

stackDepth
Answer the number of stack slots needed for the receiver

1.39.3 CompiledBlock: basic

= aMethod
Answer whether the receiver and aMethod are equal

methodCategory
Answer the method category

methodCategory: aCategory
Set the method category to the given string

methodSourceCode
Answer the method source code (a FileSegment or String or nil)

methodSourceFile
Answer the file where the method source code is stored

methodSourcePos
Answer the location where the method source code is stored in the methodSourceFile

methodSourceString
Answer the method source code as a string

1.39.4 CompiledBlock: printing

printOn: aStream
Print the receiver's class and selector on aStream

1.39.5 CompiledBlock: saving and loading

binaryRepresentationObject
This method is implemented to allow for a PluggableProxy to be used with CompiledBlocks. Answer a DirectedMessage which sends #blockAt: to the CompiledMethod containing the receiver.

1.40 CompiledCode

Defined in namespace Smalltalk
Superclass: ArrayedCollection
Category: Language-Implementation

I represent code that has been compiled. I am an abstract superclass for blocks and methods

1.40.1 CompiledCode class: cache flushing

flushTranslatorCache

Answer any kind of cache mantained by a just-in-time code translator in the virtual machine (if any). Do nothing for now.

1.40.2 CompiledCode class: instance creation

new: numBytecodes header: anInteger literals: literals

Answer a new instance of the receiver with room for the given number of bytecodes and the given header

new: numBytecodes header: anInteger numLiterals: numLiterals

Answer a new instance of the receiver with room for the given number of bytecodes and the given header

1.40.3 CompiledCode class: tables

bytecodeInfoTable

Return a ByteArray which defines some properties of the bytecodes. For each bytecode, 4 bytes are reserved. The fourth byte is a flag byte: bit 7 means that the argument is a line number to be used in creating the bytecode->line number map.

The first three have a meaning only for those bytecodes that represent a combination of operations: the combination can be BC1 ARG BC2 OPERAND if the fourth byte's bit 0 = 0 or BC1 OPERAND BC2 ARG if the fourth byte's bit 0 = 1

where BC1 is the first byte, BC2 is the second, ARG is the third and OPERAND is the bytecode argument as it appears in the bytecode stream.

specialSelectors

Answer an array of message names that don't need to be in literals to be sent in a method. Their position here reflects their integer code in bytecode.

specialSelectorsNumArgs

Answer a harmoniously-indexed array of arities for the messages answered by #specialSelectors.

1.40.4 CompiledCode: accessing

at: anIndex put: aBytecode

Store aBytecode as the anIndex-th bytecode

blockAt: anIndex
Answer the CompiledBlock attached to the anIndex-th literal, assuming that the literal is a CompiledBlock or a BlockClosure.

bytecodeAt: anIndex
Answer the anIndex-th bytecode

bytecodeAt: anIndex put: aBytecode
Store aBytecode as the anIndex-th bytecode

flags Private - Answer the optimization flags for the receiver

isAnnotated
Answer 'false'.

literalAt: anIndex
Answer the anIndex-th literal

literalAt: anInteger put: aValue
Store aValue as the anIndex-th literal

literals Answer the literals referenced by my code or any CompiledCode instances I own.

method Answer the parent method for the receiver, or self if it is a method.

methodClass
Answer the class in which the receiver is installed.

methodClass: methodClass
Set the receiver's class instance variable

numArgs Answer the number of arguments for the receiver

numLiterals
Answer the number of literals for the receiver

numTemps
Answer the number of temporaries for the receiver

primitive Answer the primitive called by the receiver

selector Answer the selector through which the method is called

selector: aSymbol
Set the selector through which the method is called

sourceCodeLinesDelta
Answer the delta from the numbers in LINE_NUMBER bytecodes to source code line numbers.

stackDepth
Answer the number of stack slots needed for the receiver

1.40.5 CompiledCode: basic

= aMethod
Answer whether the receiver is the same object as arg. Testing for equality could break the browser, since it's possible to put arbitrary objects via ##(...), so this is safer.

hash Answer an hash value for the receiver

methodCategory
Answer the method category

methodCategory: aCategory
Set the method category to the given string

methodSourceCode
Answer the method source code (a FileSegment or String or nil)

methodSourceFile
Answer the file where the method source code is stored

methodSourcePos
Answer the location where the method source code is stored in the method-SourceFile

methodSourceString
Answer the method source code as a string

1.40.6 CompiledCode: copying

deepCopy Answer a deep copy of the receiver

1.40.7 CompiledCode: debugging

examineOn: aStream
Print the contents of the receiver in a verbose way.

1.40.8 CompiledCode: decoding bytecodes

dispatchTo: anObject with: param
Disassemble the bytecodes and tell anObject about them in the form of message sends. param is given as an argument to every message send.

1.40.9 CompiledCode: literals - iteration

allLiteralSymbolsDo: aBlock
As with #allLiteralsDo:, but only call aBlock with found Symbols.

allLiteralsDo: aBlock
Walk my literals, descending into Arrays and Messages, invoking aBlock with each touched object.

literalsDo: aBlock
Invoke aBlock with each object immediately in my list of literals.

1.40.10 CompiledCode: security

verify Verify the bytecodes for the receiver, and raise an exception if the verification process failed.

1.40.11 CompiledCode: testing accesses

accesses: instVarIndex
Answer whether the receiver accesses the instance variable with the given index

assigns: instVarIndex
Answer whether the receiver writes to the instance variable with the given index

containsLiteral: anObject
Answer if the receiver contains a literal which is equal to anObject.

hasBytecode: byte between: firstIndex and: lastIndex
Answer whether the receiver includes the 'byte' bytecode in any of the indices between firstIndex and lastIndex.

jumpDestinationAt: anIndex forward: aBoolean
Answer where the jump at bytecode index 'anIndex' lands

reads: instVarIndex
Answer whether the receiver reads the instance variable with the given index

refersTo: anObject
Answer whether the receiver refers to the given object

sendsToSuper
Answer whether the receiver includes a send to super.

sourceCodeMap
Answer an array which maps bytecode indices to source code line numbers. 0 values represent invalid instruction pointer indices.

1.40.12 CompiledCode: translation

discardTranslation
Flush the just-in-time translated code for the receiver (if any).

1.41 CompiledMethod

Defined in namespace Smalltalk
Superclass: CompiledCode
Category: Language-Implementation
I represent methods that have been compiled. I can recompile methods from their source code, I can invoke Emacs to edit the source code for one of my instances, and I know how to access components of my instances.

1.41.1 CompiledMethod class: c call-outs

asyncCCall: descr numArgs: numArgs attributes: attributesArray
Return a CompiledMethod corresponding to a #asyncCCall:args: pragma with the given arguments.

cCall: descr numArgs: numArgs attributes: attributesArray
Return a CompiledMethod corresponding to a #cCall:returning:args: pragma with the given arguments.

1.41.2 CompiledMethod class: instance creation

literals: lits numArgs: numArg numTemps: numTemp attributes: attrArray bytecodes: bytecodes depth: depth
Answer a full fledged CompiledMethod. Construct the method header from the parameters, and set the literals and bytecodes to the provided ones. Also, the bytecodes are optimized and any embedded CompiledBlocks modified to refer to these literals and to the newly created CompiledMethod.

numArgs: args
Create a user-defined method (one that is sent #valueWithReceiver:-withArguments: when it is invoked) with numArgs arguments. This only makes sense when called for a subclass of CompiledMethod.

1.41.3 CompiledMethod class: lean images

stripSourceCode
Remove all the references to method source code from the system

1.41.4 CompiledMethod: accessing

allBlocksDo: aBlock
Evaluate aBlock, passing to it all the CompiledBlocks it holds

allLiterals Answer the literals referred to by the receiver and all the blocks in it

flags Private - Answer the optimization flags for the receiver

isOldSyntax
Answer whether the method was written with the old (chunk-format) syntax

method Answer the receiver, since it is already a method.

methodCategory
Answer the method category

methodCategory: aCategory
Set the method category to the given string

methodClass
Answer the class in which the receiver is installed.

methodClass: methodClass
Set the receiver's class instance variable

noteOldSyntax
Remember that the method is written with the old (chunk-format) syntax

numArgs Answer the number of arguments for the receiver

numTemps
Answer the number of temporaries for the receiver

primitive Answer the primitive called by the receiver

selector Answer the selector through which the method is called

selector: aSymbol
Set the selector through which the method is called

sourceCodeLinesDelta
Answer the delta from the numbers in LINE_NUMBER bytecodes to source code line numbers.

stackDepth
Answer the number of stack slots needed for the receiver

withAllBlocksDo: aBlock
Evaluate aBlock, passing the receiver and all the CompiledBlocks it holds

withNewMethodClass: class
Answer either the receiver or a copy of it, with the method class set to class

withNewMethodClass: class selector: selector
Answer either the receiver or a copy of it, with the method class set to class

1.41.5 CompiledMethod: attributes

attributeAt: aSymbol
Return a Message for the first attribute named aSymbol defined by the receiver, or answer an error if none was found.

attributeAt: aSymbol ifAbsent: aBlock
Return a Message for the first attribute named aSymbol defined by the receiver, or evaluate aBlock is none was found.

attributes Return an Array of Messages, one for each attribute defined by the receiver.

attributesDo: aBlock
Evaluate aBlock once for each attribute defined by the receiver, passing a Message each time.

isAnnotated
If the receiver has any attributes, answer true.

primitiveAttribute
If the receiver defines a primitive, return a Message resembling the attribute that was used to define it.

1.41.6 CompiledMethod: basic

= aMethod
Answer whether the receiver and aMethod are equal

hash Answer an hash value for the receiver

1.41.7 CompiledMethod: c call-outs

isValidCCall
Answer whether I appear to have the valid flags, information, and ops to invoke a C function and answer its result.

rewriteAsAsyncCCall: func args: argsArray
Not commented.

rewriteAsCCall: funcOrDescr for: aClass
Not commented.

rewriteAsCCall: func returning: returnType args: argsArray
Not commented.

1.41.8 CompiledMethod: compiling

methodFormattedSourceString
Answer the method source code as a string, formatted using the RBFormatter. Requires package Parser.

methodParseNode
Answer the parse tree for the receiver, or nil if there is an error. Requires package Parser.

parserClass
Answer a parser class, similar to Behavior>>parserClass, that can parse my source code. Requires package Parser.

recompile Recompile the method in the scope of the class where it leaves.

recompileNotifying: aNotifier
Recompile the method in the scope of the class where it leaves, notifying errors to aNotifier by sending it #error:.

1.41.9 CompiledMethod: invoking

valueWithReceiver: anObject withArguments: args
Execute the method within anObject, passing the elements of the args Array as parameters. The method need not reside on the hierarchy from the receiver's class to Object – it need not reside at all in a MethodDictionary, in fact – but doing bad things will compromise stability of the Smalltalk virtual machine (and don't blame anybody but yourself).

If the flags field of the method header is 6, this method instead provides a hook from which the virtual machine can call back whenever execution of the method is requested. In this case, invoking the method would cause an infinite loop (the VM asks the method to run, the method asks the VM to invoke it, and so on), so this method fails with a #subclassResponsibility error.

1.41.10 CompiledMethod: printing

printOn: aStream
Print the receiver's class and selector on aStream

storeOn: aStream
Print code to create the receiver on aStream

1.41.11 CompiledMethod: saving and loading

binaryRepresentationObject
This method is implemented to allow for a PluggableProxy to be used with CompiledMethods. Answer a DirectedMessage which sends #>> to the class object containing the receiver.

1.41.12 CompiledMethod: source code

methodRecompilationSourceString
Answer the method source code as a string, ensuring that it is in new syntax (it has brackets).

methodSourceCode
Answer the method source code (a FileSegment or String or nil)

methodSourceFile
Answer the file where the method source code is stored

methodSourcePos
Answer the location where the method source code is stored in the method-SourceFile

methodSourceString
Answer the method source code as a string

1.41.13 CompiledMethod: testing

accesses: instVarIndex
Answer whether the receiver or the blocks it contains accesses the instance variable with the given index

assigns: instVarIndex
Answer whether the receiver or the blocks it contains writes to the instance variable with the given index

isAbstract Answer whether the receiver is abstract.

reads: instVarIndex
Answer whether the receiver or the blocks it contains reads to the instance variable with the given index

sendsToSuper
Answer whether the receiver or the blocks it contains have sends to super

1.42 ContextPart

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation

My instances represent executing Smalltalk code, which represent the local environment of executable code. They contain a stack and also provide some methods that can be used in inspection or debugging.

1.42.1 ContextPart class: built ins

thisContext
Return the value of the thisContext variable. Called internally when the variable is accessed.

1.42.2 ContextPart class: exception handling

backtrace Print a backtrace from the caller to the bottom of the stack on the Transcript

backtraceOn: aStream
Print a backtrace from the caller to the bottom of the stack on aStream

1.42.3 ContextPart: accessing

at: index Answer the index-th slot in the receiver. Any read access from (self size + 1) to (self basicSize) will give nil.

at: index put: anObject
Answer the index-th slot in the receiver. Any write access from (self size + 1) to (self basicSize) will give an error unless nil is being written. This is because doing such an access first, and then updating sp, is racy: the garbage collector may trigger in the middle and move anObject, but the slot in the context won't be adjusted.

client Answer the client of this context, that is, the object that sent the message that created this context. Fail if the receiver has no parent

currentFileName
Answer the name of the file where the method source code is

environment
To create a valid execution environment for the interpreter even before it starts, GST creates a fake context whose selector is nil and which can be used as a marker for the current execution environment. This method answers that context. For processes, it answers the process block itself

home Answer the MethodContext to which the receiver refers

initialIP Answer the value of the instruction pointer when execution starts in the current context

ip Answer the current instruction pointer into the receiver

ip: newIP Set the instruction pointer for the receiver

isBlock Answer whether the receiver is a block context

isDisabled Answers whether the context is skipped when doing a return. Contexts are marked as disabled whenever a non-local return is done (either by returning from the enclosing method of a block, or with the #continue: method of ContextPart) and there are unwind contexts such as those created by #ensure:. All non-unwind contexts are then marked as disabled.

isEnvironment
To create a valid execution environment for the interpreter even before it starts, GST creates a fake context which invokes a special "termination" method. Such a context can be used as a marker for the current execution environment. Answer whether the receiver is that kind of context.

isProcess Answer whether the receiver represents a process context, i.e. a context created by BlockClosure>>#newProcess. Such a context can be recognized because it has no parent but its flags are different from those of the contexts created by the VM's prepareExecutionEnvironment function.

isUnwind Answers whether the context must continue execution even after a non-local return (a return from the enclosing method of a block, or a call to the #continue: method of ContextPart). Such contexts are created by #ensure:.

method Return the CompiledMethod being executed

methodClass
Return the class in which the CompiledMethod being executed is defined

numArgs Answer the number of arguments passed to the receiver

numTemps
Answer the number of temporaries used by the receiver

parentContext
Answer the context that called the receiver

parentContext: aContext
Set the context to which the receiver will return

push: anObject
Push an object on the receiver's stack.

receiver Return the receiver (self) for the method being executed

selector Return the selector for the method being executed

size Answer the number of valid fields for the receiver. Any read access from (self size + 1) to (self basicSize) will give nil.

sp Answer the current stack pointer into the receiver

sp: newSP
Set the stack pointer for the receiver.

validSize Answer how many elements in the receiver should be inspected

1.42.4 ContextPart: built ins

continue: anObject
Resume execution from the receiver, faking that the context on top of it in the execution chain has returned anObject. The receiver must belong to the same process as the executing context, otherwise the results are not predictable. All #ensure: (and possibly #ifCurtailed:) blocks between the currently executing context and the receiver are evaluated (which is not what would happen if you directly bashed at the parent context of thisContext).

1.42.5 ContextPart: copying

copyStack Answer a copy of the entire stack.

deepCopy Answer a copy of the entire stack, but don't copy any of the other instance variables of the context.

1.42.6 ContextPart: debugging

currentLine
Answer the 1-based number of the line that is pointed to by the receiver's instruction pointer. The DebugTools package caches information, thus making the implementation faster.

currentLineInFile
Answer the 1-based number of the line that is pointed to by the receiver's instruction pointer, relative to the method's file. The implementation is slow unless the DebugTools package is loaded.

debugger Answer the debugger that is attached to the given context. It is always nil unless the DebugTools package is loaded.

debuggerClass
Answer which debugger should be used to debug the current context chain. The class with the highest debugging priority is picked among those mentioned in the chain.

isInternalExceptionHandlingContext
Answer whether the receiver is a context that should be hidden to the user when presenting a backtrace.

1.42.7 ContextPart: enumerating

scanBacktraceFor: selectors do: aBlock
Scan the backtrace for contexts whose selector is among those listed in selectors; if one is found, invoke aBlock passing the context.

scanBacktraceForAttribute: selector do: aBlock
Scan the backtrace for contexts which have the attribute selector listed in selectors; if one is found, invoke aBlock passing the context and the attribute.

1.42.8 ContextPart: printing

backtrace Print a backtrace from the receiver to the bottom of the stack on the Transcript.

backtraceOn: aStream
Print a backtrace from the caller to the bottom of the stack on aStream.

1.42.9 ContextPart: security checks

checkSecurityFor: perm
Answer the receiver.

doSecurityCheckForName: name actions: actions target: target
Not commented.

securityCheckForName: name
Not commented.

securityCheckForName: name action: action
Not commented.

securityCheckForName: name actions: actions target: target
Not commented.

securityCheckForName: name target: target
Not commented.

1.43 Continuation

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation
At my heart, I am something like the goto instruction; my creation sets the label, and my methods do the jump. However, this is a really powerful kind of goto instruction. If your hair is turning green at this point, don't worry as you will probably only deal with users of continuations, rather than with the concept itself.

1.43.1 Continuation class: instance creation

current Return a continuation.

currentDo: aBlock
Pass a continuation to the one-argument block, aBlock and return the result of evaluating it.

escapeDo: aBlock
Pass a continuation to the one-argument block, knowing that aBlock does not fall off (either because it includes a method return, or because it yields control to another continuation). If it does, an exception will be signalled and the current process terminated.

1.43.2 Continuation: invocation

callCC Activate the original continuation, passing back in turn a continuation for the caller. The called continuation becomes unusable, and any attempt to reactivate it will cause an exception. This is not a limitation, in general, because this method is used to replace a continuation with another (see the implementation of the Generator class).

oneShotValue
Return nil to the original continuation, which becomes unusable. Attempting to reactivate it will cause an exception. This is an optimization over #value.

oneShotValue: v
Return anObject to the original continuation, which becomes unusable. Attempting to reactivate it will cause an exception. This is an optimization over #value:.

value Return nil to the original continuation, copying the stack to allow another activation.

value: anObject
Return anObject to the original continuation, copying the stack to allow another activation.

valueWithArguments: aCollection
Return the sole element of aCollection to the original continuation (or nil if aCollection is empty), copying the stack to allow another activation

1.44 CPtr

Defined in namespace Smalltalk
Superclass: CAggregate
Category: Language-C interface

1.44.1 CPtr: accessing

alignof Answer the receiver's required aligment

sizeof Answer the receiver's size

value Answer the address of the location pointed to by the receiver.

value: anObject
Set the address of the location pointed to by the receiver to anObject, which can be either an Integer or a CObject. if anObject is an Integer, it is interpreted as a 32-bit or 64-bit address. If it is a CObject, its address is stored.

1.45 CPtrCType

Defined in namespace Smalltalk
Superclass: CType
Category: Language-C interface

1.45.1 CPtrCType class: instance creation

elementType: aCType
Answer a new instance of CPtrCType that maps pointers to the given CType

from: type
Private - Called by computeAggregateType: for pointers

1.45.2 CPtrCType: accessing

elementType
Answer the type of the elements in the receiver's instances

1.45.3 CPtrCType: basic

= anObject
Return whether the receiver and anObject are equal.

hash Return a hash code for the receiver.

1.45.4 CPtrCType: storing

storeOn: aStream
Not commented.

1.46 CScalar

Defined in namespace Smalltalk
Superclass: CObject
Category: Language-C interface

1.46.1 CScalar class: instance creation

gcValue: anObject
Answer a newly allocated CObject containing the passed value, anObject, in garbage-collected storage.

type Answer a CType for the receiver—for example, CByteType if the receiver is CByte.

value: anObject
Answer a newly allocated CObject containing the passed value, anObject. Remember to call #addToBeFinalized if you want the CObject to be automatically freed

1.46.2 CScalar: accessing

cObjStoredType
Private - Provide a conversion from a CObject to a Smalltalk object to be stored by #at:put:

value Answer the value the receiver is pointing to. The exact returned value depends on the receiver's class

value: aValue
Set the receiver to point to the value, aValue. The exact meaning of aValue depends on the receiver's class

1.47 CScalarCType

Defined in namespace Smalltalk
Superclass: CType
Category: Language-C interface

1.47.1 CScalarCType: accessing

valueType valueType is used as a means to communicate to the interpreter the underlying type of the data. For scalars, it is supplied by the CObject subclass.

1.47.2 CScalarCType: storing

storeOn: aStream
Store Smalltalk code that compiles to the receiver

1.48 CShort

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.48.1 CShort class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.48.2 CShort: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.49 CSmalltalk

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.49.1 CSmalltalk class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.49.2 CSmalltalk: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.50 CString

Defined in namespace Smalltalk
Superclass: CPtr
Category: Language-C interface

Technically, CString is really a pointer to CChar. However, it can be very useful as a distinct datatype because it is a separate datatype in Smalltalk, so we allow developers to express their semantics more precisely by using a more descriptive type.

Note that like CChar is a pointer to char, CString is actually a *pointer* to string: a char ** in C terms. If you need to take a String out of a char *, use CChar>>#asString.

In general, I behave like a cross between an array of characters and a pointer to a character. I provide the protocol for both data types. My #value method returns a Smalltalk String, as you would expect for a scalar datatype.

1.50.1 CString class: accessing

cObjStoredType
Private - Provide a conversion from a CObject to a Smalltalk object to be stored by #at:put:

1.50.2 CString class: instance creation

type Answer a CType for the receiver—for example, CByteType if the receiver is CByte.

value: anObject
Answer a newly allocated CObject containing the passed value, anObject. Remember to call #addToBeFinalized if you want the CObject to be automatically freed

1.50.3 CString: accessing

cObjStoredType
Private - Provide a conversion from a CObject to a Smalltalk object to be stored by #at:put:

value Answer the value the receiver is pointing to. The exact returned value depends on the receiver's class

value: aValue
Set the receiver to point to the value, aValue. The exact meaning of aValue depends on the receiver's class

1.51 CStringCType

Defined in namespace Smalltalk
Superclass: CScalarCType
Category: Language-C interface

1.51.1 CStringCType: accessing

elementType
Answer the type of the elements in the receiver's instances

1.52 CStruct

Defined in namespace Smalltalk
Superclass: CCompound
Category: Language-C interface

1.52.1 CStruct class: subclass creation

declaration: array
Compile methods that implement the declaration in array.

1.53 CType

Defined in namespace Smalltalk
Superclass: Object
Category: Language-C interface

I am not part of the standard Smalltalk kernel class hierarchy. I contain type information used by subclasses of CObject, which represents external C data items.

My only instance variable, cObjectType, is used to hold onto the CObject subclass that gets created for a given CType. Used primarily in the C part of the interpreter because internally it cannot execute methods to get values, so it has a simple way to access instance variable which holds the desired subclass.

My subclasses have instances which represent the actual data types; for the scalar types, there is only one instance created of each, but for the aggregate types, there is at least one instance per base type and/or number of elements.

1.53.1 CType class: C instance creation

cObjectBinding: aCObjectSubclassBinding
Create a new CType for the given subclass of CObject

cObjectType: aCObjectSubclass
Create a new CType for the given subclass of CObject

computeAggregateType: type
Private - Called by from: for pointers/arrays. Format of type: (#array #int 3) or (#ptr #{FooStruct})

from: type
Private - Pass the size, alignment, and description of CType for aBlock, given the field description in 'type' (the second element of each pair).

1.53.2 CType class: initialization

initialize Initialize the receiver's TypeMap

1.53.3 CType: accessing

alignof Answer the size of the receiver's instances

arrayType: size
Answer a CArrayCType which represents an array with the given size of CObjects whose type is in turn represented by the receiver

cObjectType
Answer the CObject subclass whose instance is created when new is sent to the receiver

new: anInteger
Allocate a new CObject with room for anInteger C objects of the type (class) identified by the receiver. It is the caller's responsibility to free the memory allocated for it.

ptrType Answer a CPtrCType which represents a pointer to CObjects whose type is in turn represented by the receiver

sizeof Answer the size of the receiver's instances

valueType valueType is used as a means to communicate to the interpreter the underlying type of the data. For anything but scalars, it's just 'self'

1.53.4 CType: basic

= anObject
Return whether the receiver and anObject are equal.

hash Return a hash code for the receiver.

1.53.5 CType: C instance creation

address: cObjOrInt
Create a new CObject with the type (class) identified by the receiver, pointing to the given address (identified by an Integer or CObject).

gcNew Allocate a new CObject with the type (class) identified by the receiver. The object is movable in memory, but on the other hand it is garbage-collected automatically.

gcNew: anInteger
Allocate a new CObject with room for anInteger C object of the type (class) identified by the receiver. The object is movable in memory, but on the other hand it is garbage-collected automatically.

new Allocate a new CObject with the type (class) identified by the receiver. It is the caller's responsibility to free the memory allocated for it.

1.53.6 CType: storing

storeOn: aStream
Store Smalltalk code that compiles to the receiver

1.54 CUChar

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.54.1 CUChar class: getting info

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.54.2 CUChar: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.55 CUInt

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.55.1 CUInt class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.55.2 CUInt: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.56 CULong

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.56.1 CULong class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.56.2 CULong: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.57 CULongLong

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.57.1 CULongLong class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.57.2 CULongLong: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.58 CUnion

Defined in namespace Smalltalk
Superclass: CCompound
Category: Language-C interface

1.58.1 CUnion class: subclass creation

declaration: array
Compile methods that implement the declaration in array.

1.59 CUShort

Defined in namespace Smalltalk
Superclass: CScalar
Category: Language-C interface

1.59.1 CUShort class: accessing

alignof Answer the receiver's instances required aligment

cObjStoredType
Private - Answer an index referring to the receiver's instances scalar type

sizeof Answer the receiver's instances size

1.59.2 CUShort: accessing

alignof Answer the receiver's required aligment

cObjStoredType
Private - Answer an index referring to the receiver's scalar type

sizeof Answer the receiver's size

1.60 Date

Defined in namespace Smalltalk
Superclass: Magnitude
Category: Language-Data types
My instances represent dates. My base date is defined to be Jan 1, 1901. I provide methods for instance creation (including via "symbolic" dates, such as "Date newDay: 14 month: #Feb year: 1990".

PLEASE BE WARNED – use this class only for dates after 1582 AD; that's the beginning of the epoch. Dates before 1582 will not be correctly printed. In addition, since ten days were lost from October 5 through October 15, operations between a Gregorian date (after 15-Oct-1582) and a Julian date (before 5-Oct-1582) will give incorrect results; or, 4-Oct-1582 + 2 days will yield 6-Oct-1582 (a non-existent day!), not 16-Oct-1582.

In fact, if you pass a year < 1582 to a method like #newDay:month:year: it will assume that it is a two-digit year (e.g. 90=1990, 1000=2900). The only way to create Julian calendar dates is with the #fromDays: instance creation method.

1.60.1 Date class: basic

abbreviationOfDay: dayIndex
Answer the abbreviated name of the day of week corresponding to the given index

dayOfWeek: dayName
Answer the index of the day of week corresponding to the given name

daysInMonth: monthName forYear: yearInteger
Answer the number of days in the given (named) month for the given year

daysInYear: i
Answer the number of days in the given year

indexOfMonth: monthName
Answer the index of the month corresponding to the given name

initDayNameDict
Initialize the DayNameDict to the names of the days

initMonthNameDict
Initialize the MonthNameDict to the names of the months

initialize Initialize the receiver

nameOfDay: dayIndex
Answer the name of the day of week corresponding to the given index

nameOfMonth: monthIndex
Answer the name of the month corresponding to the given index

shortNameOfMonth: monthIndex
Answer the name of the month corresponding to the given index

1.60.2 Date class: instance creation (ANSI)

year: y day: d hour: h minute: min second: s
Answer a Date denoting the d-th day of the given year

year: y month: m day: d hour: h minute: min second: s
Answer a Date denoting the d-th day of the given (as a number) month and year

1.60.3 Date class: instance creation (Blue Book)

dateAndTimeNow
Answer an array containing the current date and time

fromDays: dayCount
Answer a Date denoting dayCount days past 1/1/1901

fromJulian: jd
Answer a Date denoting the jd-th day in the astronomical Julian calendar.

fromSeconds: time
Answer a Date denoting the date time seconds past Jan 1st, 1901

newDay: day month: monthName year: yearInteger
Answer a Date denoting the dayCount day of the given (named) month and year

newDay: day monthIndex: monthIndex year: yearInteger
Answer a Date denoting the dayCount day of the given (as a number) month and year

newDay: dayCount year: yearInteger
Answer a Date denoting the dayCount day of the yearInteger year

readFrom: aStream
Parse an instance of the receiver from aStream

today Answer a Date denoting the current date in local time

utcDateAndTimeNow
Answer an array containing the current date and time in Coordinated Universal Time (UTC)

utcToday Answer a Date denoting the current date in Coordinated Universal Time (UTC)

1.60.4 Date: basic

- aDate Answer a new Duration counting the number of days between the receiver and aDate.

addDays: dayCount
Answer a new Date pointing dayCount past the receiver

subtractDate: aDate
Answer the number of days between aDate and the receiver (negative if the receiver is before aDate)

subtractDays: dayCount
Answer a new Date pointing dayCount before the receiver

1.60.5 Date: compatibility (non-ANSI)

day Answer the day represented by the receiver

dayName Answer the day of week of the receiver as a Symbol

shortMonthName
Answer the abbreviated name of the month represented by the receiver

1.60.6 Date: date computations

asSeconds Answer the date as the number of seconds from 1/1/1901.

dayOfMonth
Answer the day represented by the receiver (same as #day)

dayOfWeek
Answer the day of week of the receiver. 1 = Monday, 7 = Sunday

dayOfWeekAbbreviation
Answer the day of week of the receiver as a Symbol

dayOfWeekName
Answer the day of week of the receiver as a Symbol

dayOfYear
Answer the days passed since 31/12 of last year; e.g. New Year's Day is 1

daysFromBaseDay
Answer the days passed since 1/1/1901

daysInMonth
Answer the days in the month represented by the receiver

daysInYear
Answer the days in the year represented by the receiver

daysLeftInMonth
Answer the days to the end of the month represented by the receiver

daysLeftInYear
Answer the days to the end of the year represented by the receiver

firstDayOfMonth
Answer a Date representing the first day of the month represented by the receiver

isLeapYear
Answer whether the receiver refers to a date in a leap year.

lastDayOfMonth
Answer a Date representing the last day of the month represented by the receiver

month Answer the index of the month represented by the receiver

monthAbbreviation
Answer the abbreviated name of the month represented by the receiver

monthIndex
Answer the index of the month represented by the receiver

monthName
Answer the name of the month represented by the receiver

year Answer the year represented by the receiver

1.60.7 Date: printing

printOn: aStream
Print a representation for the receiver on aStream

1.60.8 Date: still unclassified

+ aDuration
Answer a new Date or DateTime pointing aDuration time past the receiver.

1.60.9 Date: storing

storeOn: aStream
Store on aStream Smalltalk code compiling to the receiver

1.60.10 Date: testing

< aDate Answer whether the receiver indicates a date preceding aDate

= aDate Answer whether the receiver indicates the same date as aDate

hash Answer an hash value for the receievr

1.61 DateTime

Defined in namespace Smalltalk
Superclass: Date
Category: Language-Data types
My instances represent timestamps.

1.61.1 DateTime class: information

clockPrecision
Answer 'ClockPrecision'.

initialize Initialize the receiver's class variables

1.61.2 DateTime class: instance creation

now Answer an instance of the receiver referring to the current date and time.

readFrom: aStream
Parse an instance of the receiver from aStream

today Answer an instance of the receiver referring to midnight of today in local time.

year: y day: d hour: h minute: min second: s
Answer a DateTime denoting the d-th day of the given year, and setting the time part to the given hour, minute, and second

year: y day: d hour: h minute: min second: s offset: ofs
Answer a DateTime denoting the d-th day of the given year. Set the offset field to ofs (a Duration), and the time part to the given hour, minute, and second

year: y month: m day: d hour: h minute: min second: s
Answer a DateTime denoting the d-th day of the given (as a number) month and year, setting the time part to the given hour, minute, and second

year: y month: m day: d hour: h minute: min second: s offset: ofs
Answer a DateTime denoting the d-th day of the given (as a number) month and year. Set the offset field to ofs (a Duration), and the the time part to the given hour, minute, and second

1.61.3 DateTime class: instance creation (non-ANSI)

date: aDate time: aTime
Answer a DateTime denoting the given date and time. Set the offset field to ofs (a Duration).

date: aDate time: aTime offset: ofs
Answer a DateTime denoting the given date and time. Set the offset field to ofs (a Duration).

fromDays: days seconds: secs
Answer a DateTime denoting the given date (as days since January 1, 1901) and time (as seconds since UTC midnight).

fromDays: days seconds: secs offset: ofs
Answer a DateTime denoting the given date (as days since January 1, 1901) and time (as seconds since midnight). Set the offset field to ofs (a Duration).

fromSeconds: secs
Answer a DateTime denoting the given date and time (as seconds since January 1, 1901 midnight UTC).

fromSeconds: secs offset: ofs
Answer a DateTime denoting the given date and time (as seconds since January 1, 1901 midnight). Set the offset field to ofs (a Duration).

1.61.4 DateTime: basic

+ aDuration
Answer a new Date pointing aDuration time past the receiver

- aDateTimeOrDuration
Answer a new Date pointing dayCount before the receiver

1.61.5 DateTime: computations

asSeconds Answer the date as the number of seconds from 1/1/1901.

dayOfWeek
Answer the day of week of the receiver. Unlike Dates, DateAndTimes have 1 = Sunday, 7 = Saturday

hour Answer the hour in a 24-hour clock

hour12 Answer the hour in a 12-hour clock

hour24 Answer the hour in a 24-hour clock

meridianAbbreviation
Answer either #AM (for anti-meridian) or #PM (for post-meridian)

minute Answer the minute

second Answer the month represented by the receiver

1.61.6 DateTime: printing

printOn: aStream
Print a representation for the receiver on aStream

1.61.7 DateTime: splitting in dates & times

asDate Answer a Date referring to the same day as the receiver

asTime Answer a Time referring to the same time (from midnight) as the receiver

at: anIndex
Since in the past timestamps were referred to as Arrays containing a Date and a Time (in this order), this method provides access to DateTime objects like if they were two-element Arrays.

1.61.8 DateTime: storing

storeOn: aStream
Store on aStream Smalltalk code compiling to the receiver

1.61.9 DateTime: testing

< aDateTime
Answer whether the receiver indicates a date preceding aDate

= aDateTime
Answer whether the receiver indicates the same date as aDate

hash Answer an hash value for the receievr

1.61.10 DateTime: time zones

asLocal Answer the receiver, since DateTime objects store themselves in Local time

asUTC Convert the receiver to UTC time, and answer a new DateTime object.

offset Answer the receiver's offset from UTC to local time (e.g. +3600 seconds for Central Europe Time, -3600*6 seconds for Eastern Standard Time). The offset is expressed as a Duration

offset: anOffset
Answer a copy of the receiver with the offset from UTC to local time changed to anOffset (a Duration).

timeZoneAbbreviation
Answer an abbreviated indication of the receiver's offset, expressed as 'shhmm', where 'hh' is the number of hours and 'mm' is the number of minutes between

UTC and local time, and 's' can be '+' for the Eastern hemisphere and '-' for the Western hemisphere.

timeZoneName
Answer the time zone name for the receiver (currently, it is simply 'GMT +xxxx', where 'xxxx' is the receiver's #timeZoneAbbreviation).

1.62 DeferredVariableBinding

Defined in namespace Smalltalk
Superclass: LookupKey
Category: Language-Data types
I represent a binding to a variable that is not tied to a particular dictionary until the first access. Then, lookup rules for global variables in the scope of a given class are used.

1.62.1 DeferredVariableBinding class: basic

key: aSymbol class: aClass defaultDictionary: aDictionary
Answer a binding that will look up aSymbol as a variable in aClass's environment at first access. See #resolveBinding's comment for aDictionary's meaning.

path: anArray class: aClass defaultDictionary: aDictionary
As with #key:class:defaultDictionary:, but accepting an array of symbols, representing a namespace path, instead.

1.62.2 DeferredVariableBinding: basic

path Answer the path followed after resolving the first key.

value Answer a new instance of the receiver with the given key and value

value: anObject
Answer a new instance of the receiver with the given key and value

1.62.3 DeferredVariableBinding: storing

printOn: aStream
Put on aStream some Smalltalk code compiling to the receiver

storeOn: aStream
Put on aStream some Smalltalk code compiling to the receiver

1.63 Delay

Defined in namespace Smalltalk
Superclass: Object
Category: Kernel-Processes
I am the ultimate agent for frustration in the world. I cause things to wait (sometimes much more than is appropriate, but it is those losing operating systems' fault). When a process sends one of my instances a wait message, that process goes to sleep for the interval specified when the instance was created.

1.63.1 Delay class: instance creation

forMilliseconds: millisecondCount
Answer a Delay waiting for millisecondCount milliseconds

forNanoseconds: nanosecondCount
Answer a Delay waiting for nanosecondCount nanoseconds

forSeconds: secondCount
Answer a Delay waiting for secondCount seconds

untilMilliseconds: millisecondCount
Answer a Delay waiting until millisecondCount milliseconds after startup

untilNanoseconds: nanosecondCount
Answer a Delay waiting until nanosecondCount nanoseconds after startup

1.63.2 Delay class: still unclassified

update: aspect
Prime the timer event loop when the image starts running.

1.63.3 Delay class: timer process

activeDelay
Return the delay at the head of the queue.

handleDelayRequestor
Handle a timer event; which can be either: - a schedule or unschedule request (DelayRequestor notNil) - a timer signal (not explicitly specified) We check for timer expiry every time we get a signal.

runDelayProcess
Run the timer event loop.

scheduleDelay: aDelay
Private - Schedule this Delay. Run in the timer process, which is the only one that manipulates Queue.

startDelayLoop
Start the timer event loop.

unscheduleDelay: aDelay
Private - Unschedule this Delay. Run in the timer process, which is the only one that manipulates Queue.

1.63.4 Delay: accessing

asAbsolute
Answer a delay that waits until the current delay's resumptionTime, or delayDuration milliseconds from now if that would be nil. May answer the receiver if it is already waiting until an absolute time.

delayDuration
Answer the time I have left to wait, in milliseconds.

isAbsolute Answer whether the receiver waits until an absolute time on the millisecond clock.

resumptionTime
Answer 'resumptionTime'.

1.63.5 Delay: comparing

= aDelay Answer whether the receiver and aDelay denote the same delay

hash Answer an hash value for the receiver

1.63.6 Delay: copying

postCopy Adjust the current delay so that it behaves as if it had just been created.

1.63.7 Delay: delaying

timedWaitOn: aSemaphore
Schedule this Delay and wait on it. The current process will be suspended for the amount of time specified when this Delay was created, or until aSemaphore is signaled.

wait Schedule this Delay and wait on it. The current process will be suspended for the amount of time specified when this Delay was created.

1.63.8 Delay: initialization

initForNanoseconds: value
Initialize a Delay waiting for millisecondCount milliseconds

1.63.9 Delay: instance creation

initUntilNanoseconds: value
Initialize a Delay waiting for millisecondCount milliseconds after startup

1.63.10 Delay: testing

isActive Answer whether this Delay is being waited on.

1.63.11 Delay: timeout

value: aBlock onTimeoutDo: aTimeoutBlock
Execute aBlock for up to the time of my own delay; in case the code did not finish abort the execution, unwind the block and then evaluate aTimeoutBlock.

1.64 DelayedAdaptor

Defined in namespace Smalltalk
Superclass: PluggableAdaptor
Category: Language-Data types
I can be used where many expensive updates must be performed. My instances buffer the last value that was set, and only actually set the value when the #trigger message is sent. Apart from this, I'm equivalent to PluggableAdaptor.

1.64.1 DelayedAdaptor: accessing

trigger Really set the value of the receiver.

value Get the value of the receiver.

value: anObject
Set the value of the receiver - actually, the value is cached and is not set until the #trigger method is sent.

1.65 Dictionary

Defined in namespace Smalltalk
Superclass: HashedCollection
Category: Collections-Keyed

I implement a dictionary, which is an object that is indexed by unique objects (typcially instances of Symbol), and associates another object with that index. I use the equality operator = to determine equality of indices.

In almost all places where you would use a plain Dictionary, a LookupTable would be more efficient; see LookupTable's comment before you use it. I do have a couple of special features that are useful in certain special cases.

1.65.1 Dictionary class: instance creation

from: anArray
Answer a new dictionary created from the keys and values of Associations in anArray, such as {1 -> 2. 3 -> 4}. anArray should be specified using brace-syntax.

new Create a new dictionary with a default size

1.65.2 Dictionary: accessing

add: newObject
Add the newObject association to the receiver

addAll: aCollection
Adds all the elements of 'aCollection' to the receiver, answer aCollection

associationAt: key
Answer the key/value Association for the given key. Fail if the key is not found

associationAt: key ifAbsent: aBlock
Answer the key/value Association for the given key. Evaluate aBlock (answering the result) if the key is not found

associations
Returns the content of a Dictionary as a Set of Associations.

at: key Answer the value associated to the given key. Fail if the key is not found

at: key ifAbsent: aBlock
Answer the value associated to the given key, or the result of evaluating aBlock if the key is not found

at: aKey ifAbsentPut: aBlock
Answer the value associated to the given key. If the key is not found, evaluate aBlock and associate the result to aKey before returning.

at: aKey ifPresent: aBlock
If aKey is absent, answer nil. Else, evaluate aBlock passing the associated value and answer the result of the invocation

at: key put: value
Store value as associated to the given key

atAll: keyCollection
Answer a Dictionary that only includes the given keys. Fail if any of them is not found

keyAtValue: value
Answer the key associated to the given value, or nil if the value is not found

keyAtValue: value ifAbsent: exceptionBlock
Answer the key associated to the given value. Evaluate exceptionBlock (answering the result) if the value is not found. IMPORTANT: == is used to compare values

keys Answer a kind of Set containing the keys of the receiver

values Answer an Array containing the values of the receiver

1.65.3 Dictionary: awful ST-80 compatibility hacks

findKeyIndex: key
Tries to see if key exists as a the key of an indexed variable. As soon as nil or an association with the correct key is found, the index of that slot is answered

1.65.4 Dictionary: compilation

scopeDictionary
Answer the dictionary that is used when the receiver is before a period in Smalltalk source code.

1.65.5 Dictionary: dictionary enumerating

associationsDo: aBlock
Pass each association in the dictionary to aBlock

collect: aBlock
Answer a new dictionary where the keys are the same and the values are obtained by passing each value to aBlock and collecting the return values

do: aBlock
Pass each value in the dictionary to aBlock

keysAndValuesDo: aBlock
Pass each key/value pair in the dictionary as two distinct parameters to aBlock

keysDo: aBlock
Pass each key in the dictionary to aBlock

reject: aBlock

Answer a new dictionary containing the key/value pairs for which aBlock returns false. aBlock only receives the value part of the pairs.

select: aBlock

Answer a new dictionary containing the key/value pairs for which aBlock returns true. aBlock only receives the value part of the pairs.

1.65.6 Dictionary: dictionary removing

remove: anAssociation

Remove anAssociation's key from the dictionary

remove: anAssociation ifAbsent: aBlock

Remove anAssociation's key from the dictionary

removeAllKeys: keys

Remove all the keys in keys, without raising any errors

removeAllKeys: keys ifAbsent: aBlock

Remove all the keys in keys, passing the missing keys as parameters to aBlock as they're encountered

removeKey: key

Remove the passed key from the dictionary, fail if it is not found

removeKey: key ifAbsent: aBlock

Remove the passed key from the dictionary, answer the result of evaluating aBlock if it is not found

1.65.7 Dictionary: dictionary testing

includes: anObject

Answer whether the receiver contains anObject as one of its values

includesAssociation: anAssociation

Answer whether the receiver contains the key which is anAssociation's key and its value is anAssociation's value

includesKey: key

Answer whether the receiver contains the given key

occurrencesOf: aValue

Answer whether the number of occurrences of aValue as one of the receiver's values

1.65.8 Dictionary: namespace protocol

allSuperspaces

Answer all the receiver's superspaces in a collection

allSuperspacesDo: aBlock

Evaluate aBlock once for each of the receiver's superspaces (which is none for BindingDictionary).

definedKeys
Answer a kind of Set containing the keys of the receiver

definesKey: key
Answer whether the receiver defines the given key. 'Defines' means that the receiver's superspaces, if any, are not considered.

hereAssociationAt: key
Return the association for the variable named as specified by 'key' *in this namespace*. If the key is not found search will *not* be carried on in superspaces and the method will fail.

hereAssociationAt: key ifAbsent: aBlock
Return the association for the variable named as specified by 'key' *in this namespace*. If the key is not found search will *not* be carried on in superspaces and aBlock will be immediately evaluated.

hereAt: key
Return the value associated to the variable named as specified by 'key' *in this namespace*. If the key is not found search will *not* be carried on in superspaces and the method will fail.

hereAt: key ifAbsent: aBlock
Return the value associated to the variable named as specified by 'key' *in this namespace*. If the key is not found search will *not* be carried on in superspaces and aBlock will be immediately evaluated.

inheritsFrom: aNamespace
Answer whether aNamespace is one of the receiver's direct and indirect superspaces

superspace
Answer the receiver's superspace, which is nil for BindingDictionary.

withAllSuperspaces
Answer the receiver and all of its superspaces in a collection, which is none for BindingDictionary

withAllSuperspacesDo: aBlock
Invokes aBlock for the receiver and all superspaces, both direct and indirect (though a BindingDictionary does not have any).

1.65.9 Dictionary: printing

examineOn: aStream
Print all the instance variables and objects in the receiver on aStream

printOn: aStream
Print a representation of the receiver on aStream

1.65.10 Dictionary: rehashing

rehash Rehash the receiver

1.65.11 Dictionary: removing

removeAllKeysSuchThat: aBlock
Remove from the receiver all keys for which aBlock returns true.

1.65.12 Dictionary: storing

storeOn: aStream
Print Smalltalk code compiling to the receiver on aStream

1.65.13 Dictionary: testing

= aDictionary
Answer whether the receiver and aDictionary are equal

hash Answer the hash value for the receiver

1.66 DirectedMessage

Defined in namespace Smalltalk
Superclass: Message
Category: Language-Implementation
I represent a message send: I contain the receiver, selector and arguments for a message.

1.66.1 DirectedMessage class: creating instances

receiver: anObject selector: aSymbol
Create a new instance of the receiver

receiver: receiverObject selector: aSymbol argument: argumentObject
Create a new instance of the receiver

receiver: anObject selector: aSymbol arguments: anArray
Create a new instance of the receiver

selector: aSymbol arguments: anArray
This method should not be called for instances of this class.

selector: aSymbol arguments: anArray receiver: anObject
Create a new instance of the receiver

1.66.2 DirectedMessage: accessing

receiver Answer the receiver

receiver: anObject
Change the receiver

1.66.3 DirectedMessage: basic

printOn: aStream
Print a representation of the receiver on aStream

send Send the message

value Send the message (this message provides interoperability between DirectedMessages and blocks)

value: anObject
Send the message with the sole argument anObject (this message provides interoperability between DirectedMessages and blocks)

value: obj1 value: obj2
Send the message with the arguments obj1 and obj2 (this message provides interoperability between DirectedMessages and blocks)

valueWithArguments: anArray
Send the message with the arguments replaced by anArray (this message provides interoperability between DirectedMessages and blocks)

1.66.4 DirectedMessage: multiple process

fork Create a new process executing the receiver and start it

forkAt: priority
Create a new process executing the receiver with given priority and start it

newProcess
Create a new process executing the receiver in suspended state. The priority is the same as for the calling process. The receiver must not contain returns

1.66.5 DirectedMessage: saving and loading

reconstructOriginalObject
This method is used when DirectedMessages are used together with PluggableProxies (see ObjectDumper). It sends the receiver to reconstruct the object that was originally stored.

1.67 Directory

Defined in namespace Smalltalk
Superclass: Object
Category: Streams-Files
I am the counterpart of File in a tree-structured file system. I provide the notion of a current working directory and know several well-known places in the file system.

However, all navigation methods for directories are under FilePath or File for efficiency reasons. Refer to the manual of FilePath for information on how to use the instances returned by my class methods.

1.67.1 Directory class: file name management

append: fileName to: directory
Answer the name of a file named ‘fileName’ which resides in a directory named ‘directory’.

pathSeparator
Answer (as a Character) the character used to separate directory names

pathSeparatorString
: Answer (in a String) the character used to separate directory names

1.67.2 Directory class: file operations

allFilesMatching: aPattern do: aBlock
: Invoke #allFilesMatching:do: on the current working directory.

create: dirName
: Create a directory named dirName and answer it.

createTemporary: prefix
: Create an empty directory whose name starts with prefix and answer it.

working
: Answer the current working directory, not following symlinks.

working: dirName
: Change the current working directory to dirName.

1.67.3 Directory class: reading system defaults

execPrefix
: Answer the path to GNU Smalltalk's executable installation prefix

home
: Answer the path to the user's home directory

image
: Answer the path to GNU Smalltalk's image file

kernel
: Answer the path in which a local version of the GNU Smalltalk kernel's Smalltalk source files were searched when the image was created.

libexec
: Answer the path to GNU Smalltalk's auxiliary executables

localKernel
: Answer the path to the GNU Smalltalk kernel's Smalltalk source files. Same as 'Directory kernel' since GNU Smalltalk 3.0.

module
: Answer the path to GNU Smalltalk's dynamically loaded modules

prefix
: Answer the path to GNU Smalltalk's installation prefix

systemKernel
: Answer the path to the installed Smalltalk kernel source files.

temporary
: Answer the path in which temporary files can be created. This is read from the environment, and guessed if that fails.

userBase
: Answer the base path under which file for user customization of GNU Smalltalk are stored.

1.68 DLD

Defined in namespace Smalltalk
Superclass: Object
Category: Language-C interface

...and Gandalf said: "Many folk like to know beforehand what is to be set on the table; but those who have laboured to prepare the feast like to keep their secret; for wonder makes the words of praise louder."

I am just an ancillary class used to reference some C functions. Most of my actual functionality is used by redefinitions of methods in CFunctionDescriptor.

1.68.1 DLD class: C call-outs

defineCFunc: aName as: aFuncAddr
Register aFuncAddr as the target for cCalls to aName.

1.68.2 DLD class: dynamic linking

addLibrary: library
Add library to the search path of libraries to be used by DLD.

addLibraryHandle: libraryHandle
This is called internally by gst_dlopen. The library will be open and put in the search path.

addModule: library
Add library to the list of modules to be loaded when the image is started. The gst_initModule function in the library is called, but the library will not be put in the search path used whenever a C function is requested but not registered.

defineExternFunc: aFuncName
This method calls #primDefineExternFunc: to try to link to a function with the given name, and answers whether the linkage was successful. You can redefine this method to restrict the ability to do dynamic linking.

initialize Private - Initialize the receiver's class variables

libraryList
Answer a copy of the search path of libraries to be used by DLD

moduleList
Answer a copy of the modules reloaded when the image is started

primDefineExternFunc: aFuncName
This method tries to link to a function with the given name, and answers whether the linkage was successful. It should not be overridden.

update: aspect
Called on startup - Make DLD re-link and reset the addresses of all the externally defined functions

1.69 DumperProxy

Defined in namespace Smalltalk
Superclass: Object
Category: Streams-Files
I am an helper class for ObjectDumper. When an object cannot be saved in the standard way, you can register a subclass of me to provide special means to save that object.

1.69.1 DumperProxy class: accessing

acceptUsageForClass: aClass
: The receiver was asked to be used as a proxy for the class aClass. Answer whether the registration is fine. By default, answer true

loadFrom: anObjectDumper
: Reload a proxy stored in anObjectDumper and reconstruct the object

1.69.2 DumperProxy class: instance creation

on: anObject
: Answer a proxy to be used to save anObject. This method MUST be overridden and anObject must NOT be stored in the object's instance variables unless you override #dumpTo:, because that would result in an infinite loop!

1.69.3 DumperProxy: saving and restoring

dumpTo: anObjectDumper
: Dump the proxy to anObjectDumper – the #loadFrom: class method will reconstruct the original object.

object Reconstruct the object stored in the proxy and answer it

1.70 Duration

Defined in namespace Smalltalk
Superclass: Time
Category: Language-Data types
: My instances represent differences between timestamps.

1.70.1 Duration class: instance creation

days: d Answer a duration of 'd' days

days: d hours: h minutes: m seconds: s
: Answer a duration of 'd' days and the given number of hours, minutes, and seconds.

initialize Initialize the receiver's instance variables

milliseconds: msec
: Answer a duration of 'msec' milliseconds

readFrom: aStream
: Parse an instance of the receiver (hours/minutes/seconds) from aStream

weeks: w Answer a duration of 'w' weeks

zero Answer a duration of zero seconds.

1.70.2 Duration class: instance creation (non ANSI)

fromDays: days seconds: secs offset: unused
: Answer a duration of 'd' days and 'secs' seconds. The last parameter is unused; this message is available for interoperability with the DateTime class.

1.70.3 Duration: arithmetics

*** factor** Answer a Duration that is 'factor' times longer than the receiver

+ aDuration
Answer a Duration that is the sum of the receiver and aDuration's lengths.

- aDuration
Answer a Duration that is the difference of the receiver and aDuration's lengths.

/ factorOrDuration
If the parameter is a Duration, answer the ratio between the receiver and factorOrDuration. Else divide the receiver by factorOrDuration (a Number) and answer a new Duration that is correspondingly shorter.

abs Answer a Duration that is as long as the receiver, but always in the future.

days Answer the number of days in the receiver

isZero Answer whether the receiver correspond to a duration of zero seconds.

negated Answer a Duration that is as long as the receiver, but with past and future exchanged.

negative Answer whether the receiver is in the past.

positive Answer whether the receiver is a zero-second duration or is in the future.

printOn: aStream
Print a represention of the receiver on aStream.

1.70.4 Duration: processes

wait Answer a Delay waiting for the amount of time represented by the receiver and start waiting on it.

1.70.5 Duration: storing

storeOn: aStream
Store on aStream Smalltalk code compiling to the receiver

1.71 DynamicVariable

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Utilities

I am a variable that is visible only in the stackframes outgoing from this one. Do not use DynamicVariable directly, instead create a subclass for each variable you want to use.

You can override the #value class method, and call #valueIfAbsent: from there if you want the default value to be something else than nil.

1.71.1 DynamicVariable class: evaluating

use: anObject during: aBlock
Not commented.

value
Not commented.

valueIfAbsent: aBlock
Not commented.

1.72 Error

Defined in namespace Smalltalk
Superclass: Exception
Category: Language-Exceptions
Error represents a fatal error. Instances of it are not resumable.

1.72.1 Error: exception description

description
Answer a textual description of the exception.

isResumable
Answer false. Error exceptions are by default unresumable; subclasses can override this method if desired.

1.73 Exception

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Exceptions
My instances describe an exception that has happened, and are passed to exception handlers. Classes describe the kind of exception.

Apart from containing information on the generated exception, my instances contain methods that allow you to resume execution, leave the #on:do:... block, and pass the exception to an handler with a lower priority.

1.73.1 Exception class: comparison

goodness: anExceptionClass
Answer how good the receiver is at handling the given exception. A negative value indicates that the receiver is not able to handle the exception.

handles: anException
Answer whether the receiver handles ‘anException’.

1.73.2 Exception class: creating ExceptionCollections

, aTrappableEvent
Answer an ExceptionCollection containing all the exceptions in the receiver and all the exceptions in aTrappableEvent

1.73.3 Exception class: instance creation

new Create an instance of the receiver, which you will be able to signal later.

signal Create an instance of the receiver, give it default attributes, and signal it immediately.

signal: messageText
Create an instance of the receiver, set its message text, and signal it immediately.

1.73.4 Exception class: interoperability with TrappableEvents

allExceptionsDo: aBlock
Private - Pass ourselves to aBlock

1.73.5 Exception: accessing

basicMessageText
Answer an exception's message text. Do not override this method.

messageText
Answer an exception's message text.

messageText: aString
Set an exception's message text.

tag Answer an exception's tag value. If not specified, it is the same as the message text.

tag: anObject
Set an exception's tag value. If nil, the tag value will be the same as the message text.

1.73.6 Exception: built ins

resignalAsUnhandled: message
This might start the debugger... Note that we use #basicPrint 'cause #printOn: might invoke an error.

1.73.7 Exception: comparison

= anObject
Answer whether the receiver is equal to anObject. This is true if either the receiver or its class are the same object as anObject.

1.73.8 Exception: copying

postCopy Modify the receiver so that it does not refer to any instantiated exception handler.

1.73.9 Exception: exception description

defaultAction
Execute the default action that is attached to the receiver.

description
Answer a textual description of the exception.

isResumable
Answer true. Exceptions are by default resumable.

1.73.10 Exception: exception handling

context Return the execution context for the #on:do: snippet

isNested Answer whether the current exception handler is within the scope of another handler for the same exception.

outer Raise the exception that instantiated the receiver, passing the same parameters. If the receiver is resumable and the evaluated exception action resumes then the result returned from #outer will be the resumption value of the evaluated exception action. If the receiver is not resumable or if the exception action does not resume then this message will not return, and #outer will be equivalent to #pass.

pass Yield control to the enclosing exception action for the receiver. Similar to #outer, but control does not return to the currently active exception handler.

resignalAs: replacementException
Reinstate all handlers and execute the handler for 'replacementException'; control does not return to the currently active exception handler. The new Signal object that is created has the same contents as the receiver (this might or not be correct – if it isn't you can use an idiom such as 'sig retryUsing: [replacementException signal])

resume If the exception is resumable, resume the execution of the block that raised the exception; the method that was used to signal the exception will answer the receiver. Use this method IF AND ONLY IF you know who caused the exception and if it is possible to resume it in that particular case

resume: anObject
If the exception is resumable, resume the execution of the block that raised the exception; the method that was used to signal the exception will answer anObject. Use this method IF AND ONLY IF you know who caused the exception and if it is possible to resume it in that particular case

retry Re-execute the receiver of the #on:do: message. All handlers are reinstated: watch out, this can easily cause an infinite loop.

retryUsing: aBlock
Execute aBlock reinstating all handlers, and return its result from the #signal method.

return Exit the #on:do: snippet, answering nil to its caller.

return: anObject
Exit the #on:do: snippet, answering anObject to its caller.

1.73.11 Exception: exception signaling

signal Raise the exceptional event represented by the receiver

signal: messageText
Raise the exceptional event represented by the receiver, setting its message text to messageText.

1.73.12 Exception: still unclassified

signalingContext
Return the execution context for the place that signaled the exception, or nil if it is not available anymore (for example if the exception handler has returned.

1.74 ExceptionSet

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Exceptions
My instances are not real exceptions: they can only be used as arguments to #on:do:... methods in BlockClosure. They act as shortcuts that allows you to use the same handler for many exceptions without having to write duplicate code

1.74.1 ExceptionSet class: instance creation

new Private - Answer a new, empty ExceptionSet

1.74.2 ExceptionSet: enumerating

allExceptionsDo: aBlock
Private - Evaluate aBlock for every exception in the receiver. Answer the receiver

goodness: exception
Answer how good the receiver is at handling the given exception. A negative value indicates that the receiver is not able to handle the exception.

handles: exception
Answer whether the receiver handles 'exception'.

1.74.3 ExceptionSet: instance creation

, aTrappableEvent
Answer an ExceptionSet containing all the exceptions in the receiver and all the exceptions in aTrappableEvent

1.75 False

Defined in namespace Smalltalk
Superclass: Boolean
Category: Language-Data types

I always tell lies. I have a single instance in the system, which represents the value false.

1.75.1 False: basic

& aBoolean
We are false – anded with anything, we always answer false

and: aBlock
We are false – anded with anything, we always answer false

eqv: aBoolean
Answer whether the receiver and aBoolean represent the same boolean value

ifFalse: falseBlock
We are false – evaluate the falseBlock

ifFalse: falseBlock ifTrue: trueBlock
We are false – evaluate the falseBlock

ifTrue: trueBlock
We are false – answer nil

ifTrue: trueBlock ifFalse: falseBlock
We are false – evaluate the falseBlock

not We are false – answer true

or: aBlock We are false – ored with anything, we always answer the other operand, so evaluate aBlock

xor: aBoolean
Answer whether the receiver and aBoolean represent different boolean values

| aBoolean
We are false – ored with anything, we always answer the other operand

1.75.2 False: C hacks

asCBooleanValue
Answer '0'.

1.75.3 False: printing

printOn: aStream
Print a representation of the receiver on aStream

1.76 File

Defined in namespace Smalltalk
Superclass: FilePath
Category: Streams-Files

I enable access to the properties of files that are on disk.

1.76.1 File class: C functions

errno
Answer the current value of C errno.

stringError: errno
Answer C strerror's result for errno.

1.76.2 File class: file operations

checkError
Return whether an error had been reported or not. If there had been one, raise an exception too

checkError: errno
The error with the C code 'errno' has been reported. If errno >= 1, raise an exception

remove: fileName
Remove the file with the given path name

rename: oldFileName to: newFileName
Rename the file with the given path name oldFileName to newFileName

symlink: srcName as: destName
Create a symlink for the srcName file with the given path name

symlink: destName from: srcName
Create a symlink named destName file from the given path (relative to destName)

touch: fileName
Update the timestamp of the file with the given path name.

1.76.3 File class: initialization

initialize
Initialize the receiver's class variables

1.76.4 File class: instance creation

name: aName
Answer a new file with the given path. The path is turned into an absolute path.

path: aString
Answer a new file with the given path. The path is not validated until some of the fields of the newly created objects are accessed

1.76.5 File class: reading system defaults

executable
Answer the full path to the executable being run.

image Answer the full path to the image being used.

1.76.6 File class: testing

exists: fileName
Answer whether a file with the given name exists

isAccessible: fileName
Answer whether a directory with the given name exists and can be accessed

isExecutable: fileName
Answer whether a file with the given name exists and can be executed

isReadable: fileName
Answer whether a file with the given name exists and is readable

isWriteable: fileName
Answer whether a file with the given name exists and is writeable

1.76.7 File: accessing

asString Answer the name of the file identified by the receiver

at: aString
Answer a File or Directory object as appropriate for a file named 'aName' in the directory represented by the receiver.

creationTime
Answer the creation time of the file identified by the receiver. On some operating systems, this could actually be the last change time (the 'last change time' has to do with permissions, ownership and the like).

isDirectory
Answer whether the file is a directory.

isSocket Answer whether the file is an AF_UNIX socket.

isSymbolicLink
Answer whether the file is a symbolic link.

lastAccessTime
Answer the last access time of the file identified by the receiver

lastChangeTime
Answer the last change time of the file identified by the receiver (the 'last change time' has to do with permissions, ownership and the like). On some operating systems, this could actually be the file creation time.

lastModifyTime
Answer the last modify time of the file identified by the receiver (the 'last modify time' has to do with the actual file contents).

mode Answer the permission bits for the file identified by the receiver

mode: anInteger
Set the permission bits for the file identified by the receiver to be anInteger.

name Answer the name of the file identified by the receiver

pathTo: destName
Compute the relative path from the receiver to destName.

refresh Refresh the statistics for the receiver

size Answer the size of the file identified by the receiver

1.76.8 File: basic

= aFile Answer whether the receiver represents the same file as the receiver.

hash Answer a hash value for the receiver.

1.76.9 File: directory operations

createDirectory
Create the receiver as a directory.

namesDo: aBlock
Evaluate aBlock once for each file in the directory represented by the receiver, passing its name. aBlock should not return.

1.76.10 File: file name management

full Answer the full name of the receiver, resolving the '.' and '..' directory entries, and answer the result. Answer nil if the name is invalid (such as '/usr/../../badname')

1.76.11 File: file operations

lastAccessTime: accessDateTime lastModifyTime: modifyDateTime
Set the receiver's timestamps to be accessDateTime and modifyDateTime.

open: class mode: mode ifFail: aBlock
Open the receiver in the given mode (as answered by FileStream's class constant methods)

owner: ownerString group: groupString
Set the receiver's owner and group to be ownerString and groupString.

pathFrom: dir
Compute the relative path from the directory dirName to the receiver

remove Remove the file with the given path name

renameTo: newFileName
Rename the file with the given path name to newFileName

symlinkAs: destName
Create destName as a symbolic link of the receiver. The appropriate relative path is computed automatically.

symlinkFrom: srcName
Create the receiver as a symlink from path destName

1.76.12 File: still unclassified

, aName Answer an object of the same kind as the receiver, whose name is suffixed with aName.

1.76.13 File: testing

exists Answer whether a file with the name contained in the receiver does exist.

isAbsolute Answer whether the receiver identifies an absolute path.

isAccessible
Answer whether a directory with the name contained in the receiver does exist and is accessible

isExecutable
Answer whether a file with the name contained in the receiver does exist and is executable

isFileSystemPath
Answer whether the receiver corresponds to a real filesystem path.

isReadable
Answer whether a file with the name contained in the receiver does exist and is readable

isWriteable
Answer whether a file with the name contained in the receiver does exist and is writeable

1.77 FileDescriptor

Defined in namespace Smalltalk
Superclass: Stream
Category: Streams-Files
My instances are what conventional programmers think of as files. My instance creation methods accept the name of a disk file (or any named file object, such as /dev/rmt0 on UNIX or MTA0: on VMS). In addition, they accept a virtual filesystem path like 'configure.gz#ugz' which can be used to transparently extract or decompress files from archives, or do arbitrary processing on the files.

1.77.1 FileDescriptor class: initialization

initialize Initialize the receiver's class variables

update: aspect
Close open files before quitting

1.77.2 FileDescriptor class: instance creation

append Open for writing. The file is created if it does not exist. The stream is positioned at the end of the file.

create Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.

fopen: fileName mode: fileMode
Open fileName in the required mode - answered by #append, #create, #readWrite, #read or #write - and fail if the file cannot be opened. Else answer a new FileStream. For mode anyway you can use any standard C non-binary fopen mode. The file will be automatically closed upon GC if the object is not referenced anymore, but it is better to close it as soon as you're finished with it anyway, using #close. To keep a file open even when no references exist anymore, send it #removeToBeFinalized

fopen: fileName mode: fileMode ifFail: aBlock
Open fileName in the required mode - answered by #append, #create, #readWrite, #read or #write - and evaluate aBlock if the file cannot be opened. Else answer a new FileStream. For mode anyway you can use any The file will be automatically closed upon GC if the object is not referenced anymore, but it is better to close it as soon as you're finished with it anyway, using #close. To keep a file open even when no references exist anymore, send it #removeToBeFinalized

on: fd Open a FileDescriptor on the given file descriptor. Read-write access is assumed.

open: fileName
Open fileName in read-write mode - fail if the file cannot be opened. Else answer a new FileStream. The file will be automatically closed upon GC if the object is not referenced anymore, but you should close it with #close anyway. To keep a file open, send it #removeToBeFinalized

open: fileName mode: fileMode ifFail: aBlock
Open fileName in the required mode - answered by #append, #create, #readWrite, #read or #write - and evaluate aBlock if the file cannot be opened. Else answer a new instance of the receiver. For mode anyway you can use any standard C non-binary fopen mode. fileName can be a 'virtual filesystem' path, including URLs and '#' suffixes that are inspected by the virtual filesystem layers and replaced with tasks such as un-gzipping a file or extracting a file from an archive.

The file will be automatically closed upon GC if the object is not referenced anymore, but it is better to close it as soon as you're finished with it anyway, using #close. To keep a file open even when no references exist anymore, send it #removeToBeFinalized

openTemporaryFile: baseName
Open for writing a file whose name starts with baseName, followed by six random alphanumeric characters. The file is created with mode read/write and permissions 0666 or 0600 on most recent operating systems (beware, the former

behavior might constitute a security problem). The file is opened with the O_EXCL flag, guaranteeing that when the method returns successfully we are the only user.

popen: commandName dir: direction
Open a pipe on the given command and fail if the file cannot be opened. Else answer a new FileStream. The pipe will not be automatically closed upon GC, even if the object is not referenced anymore, because when you close a pipe you have to wait for the associated process to terminate. direction is returned by #read or #write ('r' or 'w') and is interpreted from the point of view of Smalltalk: reading means Smalltalk reads the standard output of the command, writing means Smalltalk writes the standard input of the command. The other channel (stdin when reading, stdout when writing) is the same as GST's, unless commandName alters it.

popen: commandName dir: direction ifFail: aBlock
Open a pipe on the given command and evaluate aBlock file cannot be opened. Else answer a new FileStream. The pipe will not be automatically closed upon GC, even if the object is not referenced anymore, because when you close a pipe you have to wait for the associated process to terminate. direction is interpreted from the point of view of Smalltalk: reading means that Smalltalk reads the standard output of the command, writing means that Smalltalk writes the standard input of the command

read Open text file for reading. The stream is positioned at the beginning of the file.

readWrite Open for reading and writing. The stream is positioned at the beginning of the file.

write Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.

1.77.3 FileDescriptor class: still unclassified

open: fileName mode: fileMode
Open fileName in the required mode - answered by #append, #create, #readWrite, #read or #write - and fail if the file cannot be opened. Else answer a new FileStream. For mode anyway you can use any standard C non-binary fopen mode. fileName can be a 'virtual filesystem' path, including URLs and '#' suffixes that are inspected by the virtual filesystem layers and replaced with tasks such as un-gzipping a file or extracting a file from an archive.

The file will be automatically closed upon GC if the object is not referenced anymore, but it is better to close it as soon as you're finished with it anyway, using #close. To keep a file open even when no references exist anymore, send it #removeToBeFinalized

1.77.4 FileDescriptor: accessing

canRead Answer whether the file is open and we can read from it

canWrite Answer whether the file is open and we can write from it

ensureReadable
If the file is open, wait until data can be read from it. The wait allows other Processes to run.

ensureWriteable
If the file is open, wait until we can write to it. The wait allows other Processes to run.

exceptionalCondition
Answer whether the file is open and an exceptional condition (such as presence of out of band data) has occurred on it

fd Return the OS file descriptor of the file

file Return the name of the file

isOpen Answer whether the file is still open

isPeerAlive
Present for compatibility with sockets. For files, it answers whether the file is still open

isPipe Answer whether the file is a pipe or an actual disk file

name Return the name of the file

waitForException
If the file is open, wait until an exceptional condition (such as presence of out of band data) has occurred on it. The wait allows other Processes to run.

1.77.5 FileDescriptor: basic

checkError
Perform error checking. By default, we call File class>>#checkError.

close Close the file

contents Answer the whole contents of the file

copyFrom: from to: to
Answer the contents of the file between the two given positions

finalize Close the file if it is still open by the time the object becomes garbage.

invalidate Invalidate a file descriptor

next Return the next character in the file, or nil at eof

nextByte Return the next byte in the file, or nil at eof

nextPut: aCharacter
Store aCharacter on the file

nextPutByte: anInteger
Store the byte, anInteger, on the file

nextPutByteArray: aByteArray
Store aByteArray on the file

peek Returns the next element of the stream without moving the pointer. Returns nil when at end of stream.

peekFor: anObject
Returns whether the next element of the stream is equal to anObject, without moving the pointer if it is not.

position Answer the zero-based position from the start of the file

position: n
Set the file pointer to the zero-based position n

reset Reset the stream to its beginning

shutdown Close the transmission side of a full-duplex connection. This is useful on read-write pipes.

size Return the current size of the file, in bytes

truncate Truncate the file at the current position

1.77.6 FileDescriptor: binary I/O

nextByteArray: numBytes
Return the next numBytes bytes in the byte array

nextDouble
Return the next 64-bit float in the byte array

nextFloat Return the next 32-bit float in the byte array

nextLong Return the next 4 bytes in the byte array, interpreted as a 32 bit signed int

nextLongLong
Return the next 8 bytes in the byte array, interpreted as a 64 bit signed int

nextPutDouble: aDouble
Store aDouble as a 64-bit float in the byte array

nextPutFloat: aFloat
Return the next 32-bit float in the byte array

nextPutInt64: anInteger
Store anInteger (range: -2^63..2^64-1) on the byte array as 8 bytes

nextPutLong: anInteger
Store anInteger (range: -2^31..2^32-1) on the byte array as 4 bytes

nextPutShort: anInteger
Store anInteger (range: -32768..65535) on the byte array as 2 bytes

nextShort Return the next 2 bytes in the byte array, interpreted as a 16 bit signed int

nextSignedByte
Return the next byte in the byte array, interpreted as a 8 bit signed number

nextUint64
Return the next 8 bytes in the byte array, interpreted as a 64 bit unsigned int

nextUlong Return the next 4 bytes in the byte array, interpreted as a 32 bit unsigned int

nextUshort
Return the next 2 bytes in the byte array, interpreted as a 16 bit unsigned int

1.77.7 FileDescriptor: built ins

fileIn File in the contents of the receiver. During a file in operation, global variables (starting with an uppercase letter) that are not declared don't yield an 'unknown variable' error. Instead, they are defined as nil in the 'Undeclared' dictionary (a global variable residing in Smalltalk). As soon as you add the variable to a namespace (for example by creating a class) the Association will be removed from Undeclared and reused in the namespace, so that the old references will automagically point to the new value.

fileOp: ioFuncIndex
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

fileOp: ioFuncIndex with: arg1
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex with: arg1 ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

fileOp: ioFuncIndex with: arg1 with: arg2
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex with: arg1 with: arg2 ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

fileOp: ioFuncIndex with: arg1 with: arg2 with: arg3
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex with: arg1 with: arg2 with: arg3 ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

fileOp: ioFuncIndex with: arg1 with: arg2 with: arg3 with: arg4
Private - Used to limit the number of primitives used by FileStreams

fileOp: ioFuncIndex with: arg1 with: arg2 with: arg3 with: arg4 ifFail: aBlock
Private - Used to limit the number of primitives used by FileStreams.

1.77.8 FileDescriptor: class type methods

isBinary We answer characters, so answer false

isExternalStream
We stream on an external entity (a file), so answer true

isText We answer characters, so answer true

1.77.9 FileDescriptor: initialize-release

addToBeFinalized
Add me to the list of open files.

initialize Initialize the receiver's instance variables

readStream
Answer myself, or an alternate stream coerced for reading.

removeToBeFinalized
Remove me from the list of open files.

1.77.10 FileDescriptor: low-level access

next: n putAll: aCollection startingAt: position
Put the characters in the supplied range of aCollection in the file

nextAvailable: n into: aCollection startingAt: position
Ignoring any buffering, try to fill the given range of aCollection with the contents of the file

1.77.11 FileDescriptor: overriding inherited methods

isEmpty Answer whether the receiver is empty

nextPutAllOn: aStream
Put all the characters of the receiver in aStream.

reverseContents
Return the contents of the file from the last byte to the first

setToEnd Reset the file pointer to the end of the file

skip: anInteger
Skip anInteger bytes in the file

1.77.12 FileDescriptor: polymorphism

pastEnd The end of the stream has been reached. Signal a Notification.

1.77.13 FileDescriptor: positioning

isPositionable
Answer true if the stream supports moving backwards with #skip:.

1.77.14 FileDescriptor: printing

printOn: aStream
Print a representation of the receiver on aStream

1.77.15 FileDescriptor: testing

atEnd Answer whether data has come to an end

1.78 FilePath

Defined in namespace Smalltalk
Superclass: Object
Category: Streams-Files

I expose the syntax of file names, including paths. I know how to manipulate such a path by splitting it into its components. In addition, I expose information about files (both real and virtual) such as their size and timestamps.

1.78.1 FilePath class: file name management

append: fileName to: directory
Answer the name of a file named 'fileName' which resides in a directory named 'directory'.

extensionFor: aString
Answer the extension of a file named 'aString'. Note: the extension includes an initial dot.

fullNameFor: aString
Answer the full path to a file called 'aString', resolving the '.' and '..' directory entries, and answer the result. '/..' is the same as '/'.

pathFor: aString
Determine the path of the name of a file called 'aString', and answer the result. With the exception of the root directory, the final slash is stripped.

pathFor: aString ifNone: aBlock
Determine the path of the name of a file called 'aString', and answer the result. With the exception of the root directory, the final slash is stripped. If there is no path, evaluate aBlock and return the result.

pathFrom: srcName to: destName
Answer the relative path to destName when the current directory is srcName's directory.

stripExtensionFrom: aString
Remove the extension from the name of a file called 'aString', and answer the result.

stripFileNameFor: aString
Determine the path of the name of a file called 'aString', and answer the result as a directory name including the final slash.

stripPathFrom: aString
Remove the path from the name of a file called 'aString', and answer the file name plus extension.

1.78.2 FilePath class: still unclassified

isAbsolute: aString
Answer whether aString is an absolute ptah.

1.78.3 FilePath: accessing

at: aName
Answer a File or Directory object as appropriate for a file named 'aName' in the directory represented by the receiver.

creationTime
Answer the creation time of the file identified by the receiver. On some operating systems, this could actually be the last change time (the 'last change time' has to do with permissions, ownership and the like).

group: aString
Set the group of the file identified by the receiver to be aString.

includes: aName
Answer whether a file named 'aName' exists in the directory represented by the receiver.

lastAccessTime
Answer the last access time of the file identified by the receiver

lastAccessTime: aDateTime
Update the last access time of the file corresponding to the receiver, to be aDateTime.

lastAccessTime: accessDateTime lastModifyTime: modifyDateTime
Update the timestamps of the file corresponding to the receiver, to be accessDateTime and modifyDateTime.

lastChangeTime
Answer the last change time of the file identified by the receiver (the 'last change time' has to do with permissions, ownership and the like). On some operating systems, this could actually be the file creation time.

lastModifyTime
Answer the last modify time of the file identified by the receiver (the 'last modify time' has to do with the actual file contents).

lastModifyTime: aDateTime
Update the last modification timestamp of the file corresponding to the receiver, to be aDateTime.

mode Answer the permission bits for the file identified by the receiver

mode: anInteger
Set the permission bits for the file identified by the receiver to be anInteger.

owner: aString
Set the owner of the file identified by the receiver to be aString.

owner: ownerString group: groupString
Set the owner and group of the file identified by the receiver to be aString.

pathTo: destName
Compute the relative path from the receiver to destName.

refresh Refresh the statistics for the receiver

size Answer the size of the file identified by the receiver

1.78.4 FilePath: converting

asFile Answer the receiver.

1.78.5 FilePath: decoration

all Return a decorator of the receiver that will provide recursive descent into directories for iteration methods. Furthermore, iteration on the returned wrapper will not include '.' or '..' directory entries, and will include the receiver (directly, not via '.').

1.78.6 FilePath: directory operations

createDirectories
Create the receiver as a directory, together with all its parents.

createDirectory
Create the receiver as a directory, together with all its parents.

nameAt: aName
Answer a FilePath for a file named 'aName' residing in the directory represented by the receiver.

1.78.7 FilePath: enumerating

allFilesMatching: aPattern do: aBlock
Evaluate aBlock on the File objects that match aPattern (according to String>>#match:) in the directory named by the receiver. Recursively descend into directories.

directories
Answer an Array with Directory objects for the subdirectories of the directory represented by the receiver.

do: aBlock
Evaluate aBlock once for each file in the directory represented by the receiver, passing a FilePath object (or a subclass) to it. It depends on the subclass whether iteration will include the '.' and '..' directory entries.

entries Answer an Array with File or Directory objects for the contents of the directory represented by the receiver.

entryNames
Answer an Array with the names of the files in the directory represented by the receiver.

files Answer an Array with File objects for the contents of the directory represented by the receiver.

filesMatching: aPattern
Evaluate aBlock once for each file in the directory represented by the receiver, passing a File or Directory object to aBlock. Returns the *names* of the files for which aBlock returns true.

filesMatching: aPattern do: block
Evaluate block on the File objects that match aPattern (according to String>>#match:) in the directory named by the receiver.

namesDo: aBlock
Evaluate aBlock once for each file in the directory represented by the receiver, passing its name. It depends on the subclass whether iteration will include the '.' and '..' directory entries.

namesMatching: aPattern do: block
Evaluate block on the file names that match aPattern (according to String>>#match:) in the directory named by the receiver.

reject: aBlock
Evaluate aBlock once for each file in the directory represented by the receiver, passing a File or Directory object to aBlock. Returns the *names* of the files for which aBlock returns true.

select: aBlock
Evaluate aBlock once for each file in the directory represented by the receiver, passing a File or Directory object to aBlock. Returns the *names* of the files for which aBlock returns true.

1.78.8 FilePath: file name management

directory Answer the Directory object for the receiver's path

extension Answer the extension of the receiver

full Answer the full name of the receiver, resolving the '.' and '..' directory entries, and answer the result. Answer nil if the name is invalid (such as '/usr/../../badname')

fullName Answer a String with the full path to the receiver (same as #name; it is useless to override this method).

name Answer String with the full path to the receiver (same as #fullName).

parent Answer the Directory object for the receiver's path

path Answer the path (if any) of the receiver

stripExtension
Answer the path (if any) and file name of the receiver

stripFileName
Answer the path of the receiver, always including a directory name (possibly '.') and the final directory separator

stripPath Answer the file name and extension (if any) of the receiver

1.78.9 FilePath: file operations

contents Open a read-only FileStream on the receiver, read its contents, close the stream and answer the contents

fileIn File in the receiver

open: mode
Open the receiver in the given mode (as answered by FileStream's class constant methods)

open: mode ifFail: aBlock
Open the receiver in the given mode (as answered by FileStream's class constant methods). Upon failure, evaluate aBlock.

open: class mode: mode ifFail: aBlock
Open the receiver in the given mode (as answered by FileStream's class constant methods)

openDescriptor: mode
Open the receiver in the given mode (as answered by FileStream's class constant methods)

openDescriptor: mode ifFail: aBlock
Open the receiver in the given mode (as answered by FileStream's class constant methods). Upon failure, evaluate aBlock.

pathFrom: dirName
Compute the relative path from the directory dirName to the receiver

readStream
Open a read-only FileStream on the receiver

remove Remove the file identified by the receiver

renameTo: newName
Rename the file identified by the receiver to newName

symlinkAs: destName
Create destName as a symbolic link of the receiver. The appropriate relative path is computed automatically.

symlinkFrom: srcName
Create the receiver as a symbolic link from srcName (relative to the path of the receiver).

touch Update the timestamp of the file corresponding to the receiver.

withReadStreamDo: aBlock
Answer the result of invoking aBlock with a reading stream open on me, closing it when the dynamic extent of aBlock ends.

withWriteStreamDo: aBlock
Answer the result of invoking aBlock with a writing stream open on me, closing it when the dynamic extent of aBlock ends.

writeStream
Open a write-only FileStream on the receiver

1.78.10 FilePath: printing

asString Print a representation of the receiver on aStream.

displayOn: aStream
Print a representation of the receiver on aStream.

printOn: aStream
Print a representation of the receiver on aStream.

withShellEscapes
Return the representation of the receiver with shell characters escaped.

1.78.11 FilePath: still unclassified

/ aName Answer a File or Directory object as appropriate for a file named 'aName' in the directory represented by the receiver.

1.78.12 FilePath: testing

exists Answer whether a file with the name contained in the receiver does exist.

isAbsolute Answer whether the receiver identifies an absolute path.

isAccessible
Answer whether a directory with the name contained in the receiver does exist and can be accessed

isDirectory
Answer whether a file with the name contained in the receiver does exist and identifies a directory.

isExecutable
Answer whether a file with the name contained in the receiver does exist and is executable

isFile Answer whether a file with the name contained in the receiver does exist and does not identify a directory.

isFileSystemPath
Answer whether the receiver corresponds to a real filesystem path.

isReadable
Answer whether a file with the name contained in the receiver does exist and is readable

isRelative Answer whether the receiver identifies a relative path.

isSymbolicLink
Answer whether a file with the name contained in the receiver does exist and identifies a symbolic link.

isWriteable
Answer whether a file with the name contained in the receiver does exist and is writeable

1.78.13 FilePath: virtual filesystems

zip Not commented.

1.79 FileSegment

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation

My instances represent sections of files. I am primarily used by the compiler to record source code locations. I am not a part of the normal Smalltalk-80 kernel; I am specific to the GNU Smalltalk implementation.

1.79.1 FileSegment class: basic

on: aFile startingAt: startPos for: sizeInteger
Create a new FileSegment referring to the contents of the given file, from the startPos-th byte and for sizeInteger bytes. Note that FileSegments should always be created with full paths because relative paths are interpreted to be relative to the kernel directory.

1.79.2 FileSegment class: installing

relocate Remove the kernel path from all paths that start with it. Needed to support $(DESTDIR) and relocatable installation.

1.79.3 FileSegment: basic

asString Answer a String containing the required segment of the file

copyFrom: from to: to
Answer a String containing the given subsegment of the file. As for streams, from and to are 0-based.

file Answer the File object for the file containing the segment

fileName Answer the name of the file containing the segment

filePos Answer the position in the file where the segment starts

relocateFrom: startPath map: map
If the path starts with startPath, remove that part of the path. map is a Dictionary that is used so that equal filenames stay equal, without increasing the amount of memory that the image uses.

size Answer the length of the segment

withFileDo: aBlock
Evaluate aBlock passing it the FileStream in which the segment identified by the receiver is stored

1.79.4 FileSegment: equality

= aFileSegment
Answer whether the receiver and aFileSegment are equal.

hash Answer an hash value for the receiver.

1.79.5 FileSegment: printing

printedFileName
Answer a printed representation of the file containing the segment. While introducing some ambiguity, this representation is compact eliminates the path for kernel files, and produces a relative path from the current working directory for other files.

1.80 FileStream

Defined in namespace Smalltalk
Superclass: FileDescriptor
Category: Streams-Files
My instances are what conventional programmers think of as files. My instance creation methods accept the name of a disk file (or any named file object, such as /dev/rmt0 on UNIX or MTA0: on VMS).

1.80.1 FileStream class: file-in

fileIn: aFileName
File in the aFileName file. During a file in operation, global variables (starting with an uppercase letter) that are not declared yet don't yield an 'unknown variable' error. Instead, they are defined as nil in the 'Undeclared' dictionary (a global variable residing in Smalltalk). As soon as you add the variable to a namespace (for example by creating a class) the Association will be removed from Undeclared and reused in the namespace, so that the old references will automagically point to the new value.

fileIn: aFileName ifMissing: aSymbol
Conditionally do a file in, only if the key (often a class) specified by 'aSymbol' is not present in the Smalltalk system dictionary already. During a file in operation, global variables (starting with an uppercase letter) that are not declared don't yield an 'unknown variable' error. Instead, they are defined as nil in the 'Undeclared' dictionary (a global variable residing in Smalltalk). As soon as you add the variable to a namespace (for example by creating a class) the Association will be removed from Undeclared and reused in the namespace, so that the old references will automagically point to the new value.

fileIn: aFileName ifTrue: aBoolean
Conditionally do a file in, only if the supplied boolean is true. During a file in operation, global variables (starting with an uppercase letter) that are not declared don't yield an 'unknown variable' error. Instead, they are defined as nil in the 'Undeclared' dictionary (a global variable residing in Smalltalk). As

soon as you add the variable to a namespace (for example by creating a class) the Association will be removed from Undeclared and reused in the namespace, so that the old references will automagically point to the new value.

fileIn: aFileName line: lineInteger from: realFileName at: aCharPos
File in the aFileName file giving errors such as if it was loaded from the given line, file name and starting position (instead of 1).

generateMakefileOnto: aStream
Generate a make file for the file-ins since record was last set to true. Store it on aStream

initialize Private - Initialize the receiver's class variables

record: recordFlag
Set whether Smalltalk should record information about nested file-ins. When recording is enabled, use #generateMakefileOnto: to automatically generate a valid makefile for the intervening file-ins.

require: assoc
Conditionally do a file in from the value of assoc, only if the key of assoc is not present in the Smalltalk system dictionary already. During a file in operation, global variables (starting with an uppercase letter) that are not declared don't yield an 'unknown variable' error. Instead, they are defined as nil in the 'Undeclared' dictionary (a global variable residing in Smalltalk). As soon as you add the variable to a namespace (for example by creating a class) the Association will be removed from Undeclared and reused in the namespace, so that the old references will automagically point to the new value.

verbose: verboseFlag
Set whether Smalltalk should output debugging messages when filing in

1.80.2 FileStream class: standard streams

stderr Answer a FileStream that is attached the Smalltalk program's standard error file handle, which can be used for error messages and diagnostics issued by the program.

stdin Answer a FileStream that is attached the Smalltalk program's standard input file handle, which is the normal source of input for the program.

stdout Answer a FileStream that is attached the Smalltalk program's standard output file handle; this is used for normal output from the program.

1.80.3 FileStream: basic

bufferStart
Private - Answer the offset from the start of the file corresponding to the beginning of the read buffer.

copyFrom: from to: to
Answer the contents of the file between the two given positions

next Return the next character in the file, or nil at eof

nextPut: aCharacter
Store aCharacter on the file

peek Return the next character in the file, or nil at eof. Don't advance the file pointer.

position Answer the zero-based position from the start of the file

position: n
Set the file pointer to the zero-based position n

size Return the current size of the file, in bytes

truncate Truncate the file at the current position

1.80.4 FileStream: buffering

bufferSize Answer the file's current buffer

bufferSize: bufSize
Flush the file and set the buffer's size to bufSize

clean Synchronize the file descriptor's state with the object's state.

fill Private - Fill the input buffer

flush Flush the output buffer.

newBuffer Private - Answer a String to be used as the receiver's buffer

next: n bufferAll: aCollection startingAt: pos
Private - Assuming that the buffer has space for n characters, store n characters of aCollection in the buffer, starting from the pos-th.

nextAvailable: anInteger into: aCollection startingAt: pos
Read up to anInteger bytes from the stream and store them into aCollection. Return the number of bytes read.

nextAvailable: anInteger putAllOn: aStream
Copy up to anInteger bytes from the stream into aStream. Return the number of bytes read.

pendingWrite
Answer whether the output buffer is full.

1.80.5 FileStream: compiling

segmentFrom: startPos to: endPos
Answer an object that, when sent #asString, will yield the result of sending 'copyFrom: startPos to: endPos' to the receiver

1.80.6 FileStream: initialize-release

initialize Initialize the receiver's instance variables

1.80.7 FileStream: overriding inherited methods

next: n putAll: aCollection startingAt: pos
Write n values from aCollection, the first being at pos.

nextLine Returns a collection of the same type that the stream accesses, containing the next line up to the next new-line character. Returns the entire rest of the stream's contents if no new-line character is found.

nextPutAllOn: aStream
Put all the characters of the receiver in aStream.

upTo: aCharacter
Returns a collection of the same type that the stream accesses, containing data up to aCharacter. Returns the entire rest of the stream's contents if no such character is found.

1.80.8 FileStream: testing

atEnd Answer whether data has come to an end

1.81 Float

Defined in namespace Smalltalk
Superclass: Number
Category: Language-Data types
My instances represent floating point numbers that have arbitrary precision. Besides the standard numerical operations, they provide transcendental operations too. They implement IEEE-754 correctly if the hardware supports it.

1.81.1 Float class: byte-order dependancies

signByte Answer the byte of the receiver that contains the sign bit

1.81.2 Float class: characterization

denormalized
Answer whether instances of the receiver can be in denormalized form.

e Returns the value of e. Hope is that it is precise enough

epsilon Return the smallest Float x for which is 1 + x ~= 1

fmin Return the smallest Float that is > 0.

fminDenormalized
Return the smallest Float that is > 0 if denormalized values are supported, else return 0.

ln10 Returns the value of ln 10. Hope is that it is precise enough

log10Base2
Returns the value of log2 10. Hope is that it is precise enough

pi Returns the value of pi. Hope is that it is precise enough

radix Answer the base in which computations between instances of the receiver are made. This should be 2 on about every known computer, so GNU Smalltalk always answers 2.

1.81.3 Float: arithmetic

integerPart
Return the receiver's integer part

negated Return the negation of the receiver. Unlike 0-self, this converts correctly signed zeros.

raisedToInteger: anInteger
Return self raised to the anInteger-th power

1.81.4 Float: basic

hash Answer an hash value for the receiver. Not-a-number values do not have a hash code and cannot be put in a hashed collection.

1.81.5 Float: built ins

arcCos Answer the arc-cosine of the receiver

arcSin Answer the arc-sine of the receiver

arcTan Answer the arc-tangent of the receiver

ceiling Answer the integer part of the receiver, truncated towards +infinity

cos Answer the cosine of the receiver

exp Answer 'e' (2.718281828459...) raised to the receiver

floor Answer the integer part of the receiver, truncated towards -infinity

ln Answer the logarithm of the receiver in base 'e' (2.718281828459...)

primHash Private - Answer an hash value for the receiver

raisedTo: aNumber
Answer the receiver raised to its aNumber power

sin Answer the sine of the receiver

sqrt Answer the square root of the receiver

tan Answer the tangent of the receiver

1.81.6 Float: coercing

asExactFraction
Convert the receiver into a fraction with optimal approximation, but with usually huge terms.

asFraction Convert the receiver into a fraction with a good (but undefined) approximation

truncated Convert the receiver to an Integer. Only used for LargeIntegers, there are primitives for the other cases.

1.81.7 Float: coercion

asCNumber
Convert the receiver to a kind of number that is understood by the C call-out mechanism.

1.81.8 Float: comparing

max: aNumber
Answer the maximum between the receiver and aNumber. Redefine in subclasses if necessary to ensure that if either self or aNumber is a NaN, it is always answered.

min: aNumber
Answer the minimum between the receiver and aNumber. Redefine in subclasses if necessary to ensure that if either self or aNumber is a NaN, it is always answered.

withSignOf: aNumber
Answer the receiver, with its sign possibly changed to match that of aNumber.

1.81.9 Float: compiler

literalEquals: anObject
Not commented.

literalHash
Not commented.

1.81.10 Float: converting

half Answer 0.5 in the representation of the receiver

1.81.11 Float: floating point

predecessor
Not commented.

successor Not commented.

1.81.12 Float: misc math

log: aNumber
Answer log base aNumber of the receiver

1.81.13 Float: printing

printOn: aStream
Print a representation of the receiver on aStream

1.81.14 Float: storing

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

storeLiteralOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

storeOn: aStream
Print a representation of the receiver on aStream

1.81.15 Float: testing

isExact Answer whether the receiver performs exact arithmetic. Floats do not.

isFinite Answer whether the receiver does not represent infinity, nor a NaN

isInfinite Answer whether the receiver represents positive or negative infinity

isNaN Answer whether the receiver represents a NaN

negative Answer whether the receiver is negative

positive Answer whether the receiver is positive. Negative zero is not positive, so the definition is not simply >= 0.

sign Answer 1 if the receiver is greater than 0, -1 if less than 0, else 0. Negative zero is the same as positive zero.

strictlyPositive
Answer whether the receiver is > 0

1.81.16 Float: testing functionality

isFloat Answer 'true'.

1.81.17 Float: transcendental operations

asFloat Just defined for completeness. Return the receiver.

ceilingLog: radix
Answer (self log: radix) ceiling. Use exact arithmetic if radix is not a floating point value.

estimatedLog
Answer an estimate of (self abs floorLog: 10)

floorLog: radix
Answer (self log: radix) floor. Use exact arithmetic if radix is not a floating point value.

log Answer log base 10 of the receiver.

1.81.18 Float: truncation and round off

rounded Answer the receiver, rounded to the nearest integer

1.82 FloatD

Defined in namespace Smalltalk
Superclass: Float
Category: Language-Data types

My instances represent floating point numbers that have the same accuracy as C's "double" numbers.

1.82.1 FloatD class: byte-order dependencies

fromBytes: aByteArray
Answer a float with the bytes in aByteArray, which are in big-endian format.

signByte Answer the byte of the receiver that contains the sign bit

1.82.2 FloatD class: characterization

decimalDigits
Return the number of decimal digits of precision for a FloatD. Technically, if P is the precision for the representation, then the decimal precision Q is the maximum number of decimal digits such that any floating point number with Q base 10 digits can be rounded to a floating point number with P base 2 digits and back again, without change to the Q decimal digits.

emax Return the maximum allowable exponent for a FloatD that is finite.

emin Return the maximum allowable exponent for a FloatD that is finite.

fmax Return the largest normalized FloatD that is not infinite.

fminNormalized
Return the smallest normalized FloatD that is > 0

infinity Return a FloatD that represents positive infinity.

nan Return a FloatD that represents a mathematically indeterminate value (e.g. Inf - Inf, Inf / Inf).

negativeInfinity
Return a FloatD that represents negative infinity.

precision Answer the number of bits in the mantissa. 1 + (2^-precision) = 1

1.82.3 FloatD class: converting

coerce: aNumber
Answer aNumber converted to a FloatD

1.82.4 FloatD: built ins

*** arg** Multiply the receiver and arg and answer another Number

+ arg Sum the receiver and arg and answer another Number

- arg Subtract arg from the receiver and answer another Number

/ arg Divide the receiver by arg and answer another FloatD

< arg Answer whether the receiver is less than arg

<= arg Answer whether the receiver is less than or equal to arg

= arg Answer whether the receiver is equal to arg

> arg Answer whether the receiver is greater than arg

>= arg Answer whether the receiver is greater than or equal to arg

asFloatE Answer the receiver converted to a FloatE

asFloatQ Answer the receiver converted to a FloatQ

exponent Answer the exponent of the receiver in mantissa*2^exponent representation (|mantissa|<=1)

fractionPart
Answer the fractional part of the receiver

timesTwoPower: arg
Answer the receiver multiplied by 2^arg

truncated Truncate the receiver towards zero and answer the result

~= arg Answer whether the receiver is not equal to arg

1.82.5 FloatD: coercing

asFloatD Just defined for completeness. Return the receiver.

coerce: aNumber
Coerce aNumber to the receiver's class

generality Answer the receiver's generality

unity Coerce 1 to the receiver's class

zero Coerce 0 to the receiver's class

1.82.6 FloatD: converting

half Coerce 0.5 to the receiver's class

1.83 FloatE

Defined in namespace Smalltalk
Superclass: Float
Category: Language-Data types
My instances represent floating point numbers that have the same accuracy as C's "float" numbers.

1.83.1 FloatE class: byte-order dependancies

signByte Answer the byte of the receiver that contains the sign bit

1.83.2 FloatE class: byte-order dependencies

fromBytes: aByteArray
Answer a float with the bytes in aByteArray, which are in big-endian format.

1.83.3 FloatE class: characterization

decimalDigits
Return the number of decimal digits of precision for a FloatE. Technically, if P is the precision for the representation, then the decimal precision Q is the maximum number of decimal digits such that any floating point number with Q base 10 digits can be rounded to a floating point number with P base 2 digits and back again, without change to the Q decimal digits.

e Returns the value of e. Hope is that it is precise enough

emax Return the maximum allowable exponent for a FloatE that is finite.

emin Return the maximum allowable exponent for a FloatE that is finite.

fmax Return the largest normalized FloatE that is not infinite.

fminNormalized
Return the smallest normalized FloatE that is > 0

infinity Return a FloatE that represents positive infinity.

ln10 Returns the value of ln 10. Hope is that it is precise enough

log10Base2
Returns the value of log2 10. Hope is that it is precise enough

nan Return a FloatE that represents a mathematically indeterminate value (e.g. Inf - Inf, Inf / Inf).

negativeInfinity
Return a FloatE that represents negative infinity.

pi Returns the value of pi. Hope is that it is precise enough

precision Answer the number of bits in the mantissa. 1 + (2^-precision) = 1

1.83.4 FloatE class: converting

coerce: aNumber
Answer aNumber converted to a FloatE

1.83.5 FloatE: built ins

*** arg** Multiply the receiver and arg and answer another Number

+ arg Sum the receiver and arg and answer another Number

- arg Subtract arg from the receiver and answer another Number

/ arg Divide the receiver by arg and answer another FloatE

< arg Answer whether the receiver is less than arg

<= arg Answer whether the receiver is less than or equal to arg

= arg Answer whether the receiver is equal to arg

> arg Answer whether the receiver is greater than arg

>= arg Answer whether the receiver is greater than or equal to arg

asFloatD Answer the receiver converted to a FloatD

asFloatQ Answer the receiver converted to a FloatQ

exponent Answer the exponent of the receiver in mantissa*2^exponent representation (|mantissa|<=1)

fractionPart
Answer the fractional part of the receiver

timesTwoPower: arg
Answer the receiver multiplied by 2^arg

truncated Truncate the receiver towards zero and answer the result

~= arg Answer whether the receiver is not equal to arg

1.83.6 FloatE: coercing

asFloatE Just defined for completeness. Return the receiver.

coerce: aNumber
Coerce aNumber to the receiver's class

generality Answer the receiver's generality

unity Coerce 1 to the receiver's class

zero Coerce 0 to the receiver's class

1.83.7 FloatE: converting

half Coerce 0.5 to the receiver's class

1.84 FloatQ

Defined in namespace Smalltalk
Superclass: Float
Category: Language-Data types
My instances represent floating point numbers that have the same accuracy as C's "long double" numbers.

1.84.1 FloatQ class: byte-order dependancies

signByte Answer the byte of the receiver that contains the sign bit

1.84.2 FloatQ class: characterization

decimalDigits
Return the number of decimal digits of precision for a FloatQ. Technically, if P is the precision for the representation, then the decimal precision Q is the maximum number of decimal digits such that any floating point number with Q base 10 digits can be rounded to a floating point number with P base 2 digits and back again, without change to the Q decimal digits.

e Returns the value of e. Hope is that it is precise enough

emax Return the maximum allowable exponent for a FloatQ that is finite.

emin Return the maximum allowable exponent for a FloatQ that is finite.

fmax Return the largest normalized FloatQ that is not infinite.

fminNormalized
Return the smallest normalized FloatQ that is > 0

infinity Return a FloatQ that represents positive infinity.

ln10 Returns the value of ln 10. Hope is that it is precise enough

log10Base2
Returns the value of log2 10. Hope is that it is precise enough

nan Return a FloatQ that represents a mathematically indeterminate value (e.g. Inf - Inf, Inf / Inf).

negativeInfinity
Return a FloatQ that represents negative infinity.

pi Returns the value of pi. Hope is that it is precise enough

precision Answer the number of bits in the mantissa. 1 + (2^-precision) = 1

1.84.3 FloatQ class: converting

coerce: aNumber
Answer aNumber converted to a FloatQ

1.84.4 FloatQ: built ins

*** arg** Multiply the receiver and arg and answer another Number

+ arg Sum the receiver and arg and answer another Number

- arg Subtract arg from the receiver and answer another Number

/ arg Divide the receiver by arg and answer another FloatQ

< arg Answer whether the receiver is less than arg

<= arg Answer whether the receiver is less than or equal to arg

= arg Answer whether the receiver is equal to arg

> arg Answer whether the receiver is greater than arg

>= arg Answer whether the receiver is greater than or equal to arg

asFloatD Answer the receiver converted to a FloatD

asFloatE Answer the receiver converted to a FloatE

exponent Answer the exponent of the receiver in mantissa*2^exponent representation (|mantissa|<=1)

fractionPart
Answer the fractional part of the receiver

timesTwoPower: arg
Answer the receiver multiplied by 2^arg

truncated Truncate the receiver towards zero and answer the result

~= arg Answer whether the receiver is not equal to arg

1.84.5 FloatQ: coercing

asFloatQ Just defined for completeness. Return the receiver.

coerce: aNumber
Coerce aNumber to the receiver's class

generality Answer the receiver's generality

unity Coerce 1 to the receiver's class

zero Coerce 0 to the receiver's class

1.84.6 FloatQ: converting

half Coerce 0.5 to the receiver's class

1.85 Fraction

Defined in namespace Smalltalk
Superclass: Number
Category: Language-Data types
I represent rational numbers in the form (p/q) where p and q are integers. The arithmetic operations *, +, -, /, on fractions, all return a reduced fraction.

1.85.1 Fraction class: converting

coerce: aNumber
Answer aNumber converted to a Fraction

1.85.2 Fraction class: instance creation

initialize Initialize the receiver's class variables

numerator: nInteger denominator: dInteger
Answer a new instance of fraction (nInteger/dInteger)

1.85.3 Fraction: accessing

denominator
Answer the receiver's denominator

numerator Answer the receiver's numerator

1.85.4 Fraction: arithmetic

*** aNumber**
Multiply two numbers and answer the result.

+ aNumber
Sum two numbers and answer the result.

- aNumber
Subtract aNumber from the receiver and answer the result.

/ aNumber
Divide the receiver by aNumber and answer the result.

// aNumber
Return the integer quotient of dividing the receiver by aNumber with truncation towards negative infinity.

\\\\ aNumber
Return the remainder from dividing the receiver by aNumber, (using //).

estimatedLog
Answer an estimate of (self abs floorLog: 10)

1.85.5 Fraction: coercing

ceiling Truncate the receiver towards positive infinity and return the truncated result

coerce: aNumber
Coerce aNumber to the receiver's class

floor Truncate the receiver towards negative infinity and return the truncated result

generality Return the receiver's generality

truncated Truncate the receiver and return the truncated result

unity Coerce 1 to the receiver's class

zero Coerce 0 to the receiver's class

1.85.6 Fraction: coercion

asCNumber
Convert the receiver to a kind of number that is understood by the C call-out mechanism.

1.85.7 Fraction: comparing

< arg	Test if the receiver is less than arg.
<= arg	Test if the receiver is less than or equal to arg.
= arg	Test if the receiver equals arg.
> arg	Test if the receiver is more than arg.
>= arg	Test if the receiver is greater than or equal to arg.
hash	Answer an hash value for the receiver

1.85.8 Fraction: converting

asExactFraction	Answer the receiver, it is already a Fraction
asFloatD	Answer the receiver converted to a FloatD
asFloatE	Answer the receiver converted to a FloatD
asFloatQ	Answer the receiver converted to a FloatD
asFraction	Answer the receiver, it is already a Fraction
integerPart	Answer the integer part of the receiver, expressed as a Fraction

1.85.9 Fraction: optimized cases

negated	Return the receiver, with its sign changed.
raisedToInteger: anInteger	Return self raised to the anInteger-th power.
reciprocal	Return the reciprocal of the receiver
sqrt	Return the square root of the receiver.
squared	Return the square of the receiver.

1.85.10 Fraction: printing

printOn: aStream	Print a representation of the receiver on aStream
storeOn: aStream	Store Smalltalk code compiling to the receiver on aStream

1.85.11 Fraction: testing

isRational	Answer whether the receiver is rational - true

1.86 Generator

Defined in namespace Smalltalk
Superclass: Stream
Category: Streams-Generators

A Generator object provides a way to use blocks to define a Stream of many return values. The return values are computed one at a time, as needed, and hence need not even be finite.

A generator block is converted to a Generator with "Generator on: [...]". The Generator itself is passed to the block, and as soon as a message like #next, #peek, #atEnd or #peekFor: is sent to the generator, execution of the block starts/resumes and goes on until the generator's #yield: method is called: then the argument of #yield: will be the Generator's next element. If the block goes on to the end without calling #yield:, the Generator will produce no more elements and #atEnd will return true.

You could achieve the effect of generators manually by writing your own class and storing all the local variables of the generator as instance variables. For example, returning a list of integers could be done by setting a variable to 0, and having the #next method increment it and return it. However, for a moderately complicated generator, writing a corresponding class would be much messier (and might lead to code duplication or inefficiency if you want to support #peek, #peekFor: and/or #atEnd): in general, providing a #do:-like interface is easy, but not providing a Stream-like one (think binary trees).

The idea of generators comes from other programming languages, in particular this interface looks much like Scheme streams and Python generators. But Python in turn mutuated the idea for example from Icon, where the idea of generators is central. In Icon, every expression and function call behaves like a generator, and if a statement manages scalars, it automatically uses up all the results that the corresponding generator provides; on the other hand, Icon does not represent generators as first-class objects like Python and Smalltalk do.

1.86.1 Generator class: instance creation

inject: aValue into: aBlock
Return an infinite generator; the first item is aValue, the following items are obtained by passing the previous value to aBlock.

on: aBlock
Return a generator and pass it to aBlock. When #next is sent to the generator, the block will start execution, and will be suspended again as soon as #yield: is sent from the block to the generator.

on: aCollection do: aBlock
Return a generator; for each item of aCollection, evaluate aBlock passing the generator and the item.

1.86.2 Generator: stream protocol

atEnd Answer whether more data can be generated.

next Evaluate the generator until it generates the next value or decides that nothing else can be generated.

peek Evaluate the generator until it generates the next value or decides that nothing else can be generated, and save the value so that #peek or #next will return it again.

peekFor: anObject
Evaluate the generator until it generates the next value or decides that nothing else can be generated, and if it is not equal to anObject, save the value so that #peek or #next will return it again.

yield: anObject
When entering from the generator the code in the block is executed and control flow goes back to the consumer. When entering from the consumer, the code after the continuation is executed, which resumes execution of the generator block.

1.87 Getopt

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Data types
This class is usually not instantiated. Class methods provide a way to parse command lines from Smalltalk.

1.87.1 Getopt class: instance creation

parse: args with: pattern do: actionBlock
Parse the command-line arguments in args according to the syntax specified in pattern. For every command-line option found, the two-argument block actionBlock is evaluated passing the option name and the argument. For file names (or in general, other command-line arguments than options) the block's first argument will be nil. For options without arguments, or with unspecified optional arguments, the block's second argument will be nil. The option name will be passed as a character object for short options, and as a string for long options.

If an error is found, nil is returned. For more information on the syntax of pattern, see #parse:with:do:ifError:.

parse: args with: pattern do: actionBlock ifError: errorBlock
Parse the command-line arguments in args according to the syntax specified in pattern. For every command-line option found, the two-argument block actionBlock is evaluated passing the option name and the argument. For file names (or in general, other command-line arguments than options) the block's first argument will be nil. For options without arguments, or with unspecified optional arguments, the block's second argument will be nil. The option name will be passed as a character object for short options, and as a string for long options.

If an error is found, the parsing is interrupted, errorBlock is evaluated, and the returned value is answered.

Every whitespace-separated part ('word') of pattern specifies a command-line option. If a word ends with a colon, the option will have a mandatory argument. If a word ends with two colons, the option will have an optional argument. Before the colons, multiple option names (either short names like '-l' or long names like '–long') can be specified. Before passing the option to actionBlock, the name will be canonicalized to the last one.

Prefixes of long options are accepted as long as they're unique, and they are canonicalized to the full name before passing it to actionBlock. Additionally, the full name of an option is accepted even if it is the prefix of a longer option.

Mandatory arguments can appear in the next argument, or in the same argument (separated by an = for arguments to long options). Optional arguments must appear in the same argument.

1.88 Halt

Defined in namespace Smalltalk
Superclass: Exception
Category: Language-Exceptions

Halt represents a resumable error, usually a bug.

1.88.1 Halt: description

description

Answer a textual description of the exception.

isResumable

Answer true. #halt exceptions are by default resumable.

1.89 HashedCollection

Defined in namespace Smalltalk
Superclass: Collection
Category: Collections-Unordered

I am an hashed collection that can store objects uniquely and give fast responses on their presence in the collection.

1.89.1 HashedCollection class: instance creation

new Answer a new instance of the receiver with a default size

new: anInteger

Answer a new instance of the receiver with the given capacity

withAll: aCollection

Answer a collection whose elements are all those in aCollection

1.89.2 HashedCollection: accessing

add: newObject
Add newObject to the set, if and only if the set doesn't already contain an occurrence of it. Don't fail if a duplicate is found. Answer anObject

at: index This method should not be called for instances of this class.

at: index put: value
This method should not be called for instances of this class.

1.89.3 HashedCollection: builtins

primAt: anIndex
Private - Answer the anIndex-th item of the hash table for the receiver. Using this instead of basicAt: allows for easier changes in the representation

primAt: anIndex put: value
Private - Store value in the anIndex-th item of the hash table for the receiver. Using this instead of basicAt:put: allows for easier changes in the representation

primSize Private - Answer the size of the hash table for the receiver. Using this instead of basicSize allows for easier changes in the representation

1.89.4 HashedCollection: copying

deepCopy Returns a deep copy of the receiver (the instance variables are copies of the receiver's instance variables)

shallowCopy
Returns a shallow copy of the receiver (the instance variables are not copied)

1.89.5 HashedCollection: enumerating the elements of a collection

do: aBlock
Enumerate all the non-nil members of the set

1.89.6 HashedCollection: rehashing

rehash Rehash the receiver

1.89.7 HashedCollection: removing

remove: oldObject ifAbsent: anExceptionBlock
Remove oldObject from the set. If it is found, answer oldObject. Otherwise, evaluate anExceptionBlock and answer its value.

1.89.8 HashedCollection: saving and loading

postLoad Called after loading an object; rehash the collection because identity objects will most likely mutate their hashes.

postStore Called after an object is dumped. Do nothing – necessary because by default this calls #postLoad by default

1.89.9 HashedCollection: storing

storeOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

1.89.10 HashedCollection: testing collections

= aHashedCollection
Returns true if the two sets have the same membership, false if not

capacity Answer how many elements the receiver can hold before having to grow.

hash Return the hash code for the members of the set. Since order is unimportant, we use a commutative operator to compute the hash value.

includes: anObject
Answer whether the receiver contains an instance of anObject.

isEmpty Answer whether the receiver is empty.

occurrencesOf: anObject
Return the number of occurrences of anObject. Since we're a set, this is either 0 or 1. Nil is never directly in the set, so we special case it (the result is always 1).

size Answer the receiver's size

1.90 HomedAssociation

Defined in namespace Smalltalk
Superclass: Association
Category: Language-Data types
My instances represent know about their parent namespace, which is of use when implementing weak collections and finalizations.

1.90.1 HomedAssociation class: basic

key: aKey value: aValue environment: aNamespace
Answer a new association with the given key and value

1.90.2 HomedAssociation: accessing

environment
Answer the namespace in which I live.

environment: aNamespace
Set the namespace in which I live to be aNamespace.

1.90.3 HomedAssociation: finalization

mourn This message is sent to the receiver when the object is made ephemeron (which is common when HomedAssociations are used by a WeakKeyDictionary or a WeakSet). The mourning of the object's key is first of all demanded to the environment (which will likely remove the object from itself), and then performed as usual by clearing the key and value fields.

1.90.4 HomedAssociation: storing

storeOn: aStream
Put on aStream some Smalltalk code compiling to the receiver

1.91 IdentityDictionary

Defined in namespace Smalltalk
Superclass: LookupTable
Category: Collections-Keyed
I am similar to LookupTable, except that I use the object identity comparision message == to determine equivalence of indices.

1.92 IdentitySet

Defined in namespace Smalltalk
Superclass: Set
Category: Collections-Unordered
I am the typical set object; I can store any objects uniquely. I use the == operator to determine duplication of objects.

1.92.1 IdentitySet: testing

identityIncludes: anObject
Answer whether we include the anObject object; for IdentitySets this is identical to #includes:

1.93 Integer

Defined in namespace Smalltalk
Superclass: Number
Category: Language-Data types
I am the abstract integer class of the GNU Smalltalk system. My subclasses' instances can represent signed integers of various sizes (a subclass is picked according to the size), with varying efficiency.

1.93.1 Integer class: converting

coerce: aNumber
Answer aNumber converted to a kind of Integer

1.93.2 Integer: accessing

denominator
Answer '1'.

numerator Answer the receiver.

1.93.3 Integer: basic

hash Answer an hash value for the receiver

1.93.4 Integer: bit operators

allMask: anInteger
True if all 1 bits in anInteger are 1 in the receiver

anyMask: anInteger
True if any 1 bits in anInteger are 1 in the receiver

bitAt: index
Answer the index-th bit of the receiver (the LSB has an index of 1)

bitAt: index put: value
Answer an integer which is identical to the receiver, possibly with the exception of the index-th bit of the receiver (the LSB having an index of 1), which assumes a value equal to the low-order bit of the second parameter.

bitClear: aMask
Answer an Integer equal to the receiver, except that all the bits that are set in aMask are cleared.

bitInvert Return the 1's complement of the bits of the receiver

clearBit: index
Clear the index-th bit of the receiver and answer a new Integer

digitAt: index
Answer the index-th base-256 digit of the receiver (byte), expressed in two's complement

highBit Return the index of the highest order 1 bit of the receiver.

isBitSet: index
Answer whether the index-th bit of the receiver is set

lowBit Return the index of the lowest order 1 bit of the receiver.

noMask: anInteger
Answer true if no 1 bits in anInteger are 1 in the receiver.

setBit: index
Set the index-th bit of the receiver and answer a new Integer

1.93.5 Integer: converting

asCharacter
Return self as a Character or UnicodeCharacter object.

asFraction Return the receiver converted to a fraction

asScaledDecimal: n
Answer the receiver, converted to a ScaledDecimal object. The scale is forced to be 0.

ceiling Return the receiver - it's already truncated

coerce: aNumber
Coerce aNumber to the receiver's class.

floor Return the receiver - it's already truncated

rounded Return the receiver - it's already truncated

truncated Return the receiver - it's already truncated

1.93.6 Integer: extension

alignTo: anInteger
Answer the receiver, truncated to the first higher or equal multiple of anInteger (which must be a power of two)

1.93.7 Integer: iterators

timesRepeat: aBlock
Evaluate aBlock a number of times equal to the receiver's value. Compiled in-line for no argument aBlocks without temporaries, and therefore not overridable.

1.93.8 Integer: math methods

binomial: anInteger
Compute the number of combinations of anInteger objects among a number of objects given by the receiver.

ceilingLog: radix
Answer (self log: radix) ceiling. Optimized to answer an integer.

estimatedLog
Answer an estimate of (self abs floorLog: 10)

even Return whether the receiver is even

factorial Return the receiver's factorial.

floorLog: radix
Answer (self log: radix) floor. Optimized to answer an integer.

gcd: anInteger
Return the greatest common divisor (Euclid's algorithm) between the receiver and anInteger

lcm: anInteger
Return the least common multiple between the receiver and anInteger

odd Return whether the receiver is odd

1.93.9 Integer: printing

displayOn: aStream
Print on aStream the base 10 representation of the receiver

displayString
Return the base 10 representation of the receiver

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

printOn: aStream
Print on aStream the base 10 representation of the receiver

printOn: aStream base: b
Print on aStream the base b representation of the receiver

printOn: aStream paddedWith: padding to: size
Print on aStream the base 10 representation of the receiver, padded if necessary to size characters with copies of padding.

printOn: aStream paddedWith: padding to: size base: baseInteger
Print on aStream the base b representation of the receiver, padded if necessary to size characters with copies of padding.

printPaddedWith: padding to: size
Return the base baseInteger representation of the receiver, padded if necessary to size characters with copies of padding.

printPaddedWith: padding to: size base: baseInteger
Return the base baseInteger representation of the receiver, padded if necessary to size characters with copies of padding.

printString
Return the base 10 representation of the receiver

printString: baseInteger
Return the base baseInteger representation of the receiver

printStringRadix: baseInteger
Return the base baseInteger representation of the receiver, with BBr in front of it

radix: baseInteger
Return the base baseInteger representation of the receiver, with BBr in front of it. This method is deprecated, use #printStringRadix: instead.

storeLiteralOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

storeOn: aStream base: b
Print on aStream Smalltalk code compiling to the receiver, represented in base b

1.93.10 Integer: storing

storeOn: aStream
Print on aStream the base 10 representation of the receiver

storeString
Return the base 10 representation of the receiver

1.93.11 Integer: testing functionality

isInteger Answer ‘true’.

isRational Answer whether the receiver is rational - true

1.94 Interval

Defined in namespace Smalltalk
Superclass: ArrayedCollection
Category: Collections-Sequenceable

My instances represent ranges of objects, typically Number type objects. I provide iteration/enumeration messages for producing all the members that my instance represents.

1.94.1 Interval class: instance creation

from: startInteger to: stopInteger
Answer an Interval going from startInteger to the stopInteger, with a step of 1

from: startInteger to: stopInteger by: stepInteger
Answer an Interval going from startInteger to the stopInteger, with a step of stepInteger

withAll: aCollection
Answer an Interval containing the same elements as aCollection. Fail if it is not possible to create one.

1.94.2 Interval: basic

at: index Answer the index-th element of the receiver.

at: index put: anObject
This method should not be called for instances of this class.

collect: aBlock
Evaluate the receiver for each element in aBlock, collect in an array the result of the evaluations.

copyFrom: startIndex to: stopIndex
Not commented.

do: aBlock
Evaluate the receiver for each element in aBlock

isEmpty Answer whether the receiver is empty.

reverse Answer a copy of the receiver with all of its items reversed

size Answer the number of elements in the receiver.

species Answer ‘Array’.

1.94.3 Interval: printing

first Not commented.

increment Answer ‘step’.

last Answer the last value.

printOn: aStream
Print a representation for the receiver on aStream

1.94.4 Interval: storing

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.94.5 Interval: testing

= anInterval
Answer whether anInterval is the same interval as the receiver

hash Answer an hash value for the receiver

isExact Answer whether elements of the receiver are computed using exact arithmetic. This is true as long as the start and step value are exact (i.e. not floating-point).

1.95 Iterable

Defined in namespace Smalltalk
Superclass: Object
Category: Collections
I am an abstract class. My instances are collections of objects that can be iterated. The details on how they can be mutated (if at all possible) are left to the subclasses.

1.95.1 Iterable class: multibyte encodings

isUnicode Answer true; the receiver is able to store arbitrary Unicode characters.

1.95.2 Iterable: enumeration

, anIterable
Answer an iterable that enumerates first the elements of the receiver and then the elements of anIterable.

allSatisfy: aBlock
Search the receiver for an element for which aBlock returns false. Answer true if none does, false otherwise.

anySatisfy: aBlock
Search the receiver for an element for which aBlock returns true. Answer true if some does, false otherwise.

collect: aBlock
Answer a new instance of a Collection containing all the results of evaluating aBlock passing each of the receiver's elements

conform: aBlock
Search the receiver for an element for which aBlock returns false. Answer true if none does, false otherwise.

contains: aBlock
Search the receiver for an element for which aBlock returns true. Answer true if some does, false otherwise.

count: aBlock
Count the elements of the receiver for which aBlock returns true, and return their number.

detect: aBlock
Search the receiver for an element for which aBlock returns true. If some does, answer it. If none does, fail

detect: aBlock ifNone: exceptionBlock
Search the receiver for an element for which aBlock returns true. If some does, answer it. If none does, answer the result of evaluating aBlock

do: aBlock
Enumerate each object of the receiver, passing them to aBlock

do: aBlock separatedBy: separatorBlock
Enumerate each object of the receiver, passing them to aBlock. Between every two invocations of aBlock, invoke separatorBlock

fold: binaryBlock
First, pass to binaryBlock the first and second elements of the receiver; for each subsequent element, pass the result of the previous evaluation and an element. Answer the result of the last invocation, or the first element if the collection has size 1. Fail if the collection is empty.

inject: thisValue into: binaryBlock
First, pass to binaryBlock thisValue and the first element of the receiver; for each subsequent element, pass the result of the previous evaluation and an element. Answer the result of the last invocation.

noneSatisfy: aBlock
Search the receiver for an element for which aBlock returns true. Answer true if none does, false otherwise.

reject: aBlock
Answer a new instance of a Collection containing all the elements in the receiver which, when passed to aBlock, don't answer true

select: aBlock
Answer a new instance of a Collection containing all the elements in the receiver which, when passed to aBlock, answer true

1.95.3 Iterable: iteration

ifNil: nilBlock ifNotNilDo: iterableBlock
Evaluate nilBlock if the receiver is nil, else evaluate iterableBlock with each element of the receiver (which should be an Iterable).

ifNotNilDo: iterableBlock
Evaluate iterableBlock with each element of the receiver (which should be an Iterable) if not nil. Else answer nil

ifNotNilDo: iterableBlock ifNil: nilBlock
Evaluate nilBlock if the receiver is nil, else evaluate iterableBlock, passing each element of the receiver (which should be an Iterable).

1.95.4 Iterable: streaming

nextPutAllOn: aStream
Write all the objects in the receiver to aStream

readStream
Return a stream with the same contents as the receiver.

1.96 LargeArray

Defined in namespace Smalltalk
Superclass: LargeArrayedCollection
Category: Collections-Sequenceable
I am similar to a plain array, but I'm specially designed to save memory when lots of items are nil.

1.96.1 LargeArray: overridden

newCollection: size
Create an Array of the given size

1.97 LargeArrayedCollection

Defined in namespace Smalltalk
Superclass: ArrayedCollection
Category: Collections-Sequenceable
I am an abstract class specially designed to save memory when lots of items have the same value.

1.97.1 LargeArrayedCollection class: instance creation

new: anInteger
Answer a new instance of the receiver, with room for anInteger elements.

1.97.2 LargeArrayedCollection: accessing

at: anIndex
Answer the anIndex-th item of the receiver.

at: anIndex put: anObject
Replace the anIndex-th item of the receiver with anObject.

compress Arrange the representation of the array for maximum memory saving.

1.97.3 LargeArrayedCollection: basic

= aLargeArray
Answer whether the receiver and aLargeArray have the same contents

hash Answer an hash value for the receiver

size Answer the maximum valid index for the receiver

1.98 LargeByteArray

Defined in namespace Smalltalk
Superclass: LargeArrayedCollection
Category: Collections-Sequenceable

I am similar to a plain ByteArray, but I'm specially designed to save memory when lots of items are zero.

1.98.1 LargeByteArray: overridden

costOfNewIndex

Answer the maximum number of consecutive items set to the defaultElement that can be present in a compressed array.

defaultElement

Answer the value which is hoped to be the most common in the array

newCollection: size

Create a ByteArray of the given size

1.99 LargeInteger

Defined in namespace Smalltalk
Superclass: Integer
Category: Language-Data types

I represent a large integer, which has to be stored as a long sequence of bytes. I have methods to do arithmetics and comparisons, but I need some help from my children, LargePositiveInteger and LargeNegativeInteger, to speed them up a bit.

1.99.1 LargeInteger: accessing

raisedToInteger: n

Return self raised to the anInteger-th power

1.99.2 LargeInteger: arithmetic

*** aNumber**

Multiply aNumber and the receiver, answer the result

+ aNumber

Sum the receiver and aNumber, answer the result

- aNumber

Subtract aNumber from the receiver, answer the result

/ aNumber

Divide aNumber and the receiver, answer the result (an Integer or Fraction)

// aNumber

Divide aNumber and the receiver, answer the result truncated towards -infinity

\\ aNumber
Divide aNumber and the receiver, answer the remainder truncated towards -infinity

divExact: aNumber
Dividing receiver by arg assuming that the remainder is zero, and answer the result

estimatedLog
Answer an estimate of (self abs floorLog: 10)

negated Answer the receiver's negated

quo: aNumber
Divide aNumber and the receiver, answer the result truncated towards 0

rem: aNumber
Divide aNumber and the receiver, answer the remainder truncated towards 0

1.99.3 LargeInteger: bit operations

bitAnd: aNumber
Answer the receiver ANDed with aNumber

bitAt: aNumber
Answer the aNumber-th bit in the receiver, where the LSB is 1

bitInvert Answer the receiver's 1's complement

bitOr: aNumber
Answer the receiver ORed with aNumber

bitShift: aNumber
Answer the receiver shifted by aNumber places

bitXor: aNumber
Answer the receiver XORed with aNumber

lowBit Return the index of the lowest order 1 bit of the receiver.

1.99.4 LargeInteger: built-ins

at: anIndex
Answer the anIndex-th byte in the receiver's representation

at: anIndex put: aNumber
Set the anIndex-th byte in the receiver's representation

digitAt: anIndex
Answer the index-th base-256 digit of the receiver (byte), expressed in two's complement

digitAt: anIndex put: aNumber
Set the anIndex-th base-256 digit in the receiver's representation

digitLength
Answer the number of base-256 digits in the receiver

hash Answer an hash value for the receiver

primReplaceFrom: start to: stop with: replacementString startingAt: replaceStart
Private - Replace the characters from start to stop with new characters contained in replacementString (which, actually, can be any variable byte class), starting at the replaceStart location of replacementString

size Answer the number of indexed instance variable in the receiver

1.99.5 LargeInteger: coercion

asCNumber
Convert the receiver to a kind of number that is understood by the C call-out mechanism.

coerce: aNumber
Truncate the number; if needed, convert it to LargeInteger representation.

generality Answer the receiver's generality

unity Coerce 1 to the receiver's class

zero Coerce 0 to the receiver's class

1.99.6 LargeInteger: disabled

asObject This method always fails. The number of OOPs is far less than the minimum number represented with a LargeInteger.

asObjectNoFail
Answer 'nil'.

1.99.7 LargeInteger: primitive operations

basicLeftShift: totalShift
Private - Left shift the receiver by aNumber places

basicRightShift: totalShift
Private - Right shift the receiver by 'shift' places

largeNegated
Private - Same as negated, but always answer a LargeInteger

1.99.8 LargeInteger: testing

< aNumber
Answer whether the receiver is smaller than aNumber

<= aNumber
Answer whether the receiver is smaller than aNumber or equal to it

= aNumber
Answer whether the receiver and aNumber identify the same number.

> aNumber
Answer whether the receiver is greater than aNumber

>= aNumber
Answer whether the receiver is greater than aNumber or equal to it

~= aNumber
Answer whether the receiver and aNumber identify different numbers.

1.100 LargeNegativeInteger

Defined in namespace Smalltalk
Superclass: LargeInteger
Category: Language-Data types
Just like my brother LargePositiveInteger, I provide a few methods that allow LargeInteger to determine the sign of a large integer in a fast way during its calculations. For example, I know that I am smaller than any LargePositiveInteger

1.100.1 LargeNegativeInteger: converting

asFloatD Answer the receiver converted to a FloatD

asFloatE Answer the receiver converted to a FloatE

asFloatQ Answer the receiver converted to a FloatQ

1.100.2 LargeNegativeInteger: numeric testing

abs Answer the receiver's absolute value.

negative Answer whether the receiver is < 0

positive Answer whether the receiver is >= 0

sign Answer the receiver's sign

strictlyPositive
Answer whether the receiver is > 0

1.100.3 LargeNegativeInteger: reverting to LargePositiveInteger

+ aNumber
Sum the receiver and aNumber, answer the result

- aNumber
Subtract aNumber from the receiver, answer the result

gcd: anInteger
Return the greatest common divisor between the receiver and anInteger

highBit Answer the receiver's highest bit's index

1.101 LargePositiveInteger

Defined in namespace Smalltalk
Superclass: LargeInteger
Category: Language-Data types

Just like my brother LargeNegativeInteger, I provide a few methods that allow LargeInteger to determine the sign of a large integer in a fast way during its calculations. For example, I know that I am larger than any LargeNegativeInteger. In addition I implement the guts of arbitrary precision arithmetic.

1.101.1 LargePositiveInteger: arithmetic

+ aNumber
Sum the receiver and aNumber, answer the result

- aNumber
Subtract aNumber from the receiver, answer the result

gcd: anInteger
Calculate the GCD between the receiver and anInteger

highBit Answer the receiver's highest bit's index

1.101.2 LargePositiveInteger: converting

asFloatD Answer the receiver converted to a FloatD

asFloatE Answer the receiver converted to a FloatE

asFloatQ Answer the receiver converted to a FloatQ

replace: str withStringBase: radix
Return in a String str the base radix representation of the receiver.

1.101.3 LargePositiveInteger: helper byte-level methods

bytes: byteArray1 from: j compare: byteArray2
Private - Answer the sign of byteArray2 - byteArray1; the j-th byte of byteArray1 is compared with the first of byteArray2, the j+1-th with the second, and so on.

bytes: byteArray1 from: j subtract: byteArray2
Private - Sutract the bytes in byteArray2 from those in byteArray1

bytes: bytes multiply: anInteger
Private - Multiply the bytes in bytes by anInteger, which must be < 255. Put the result back in bytes.

bytesLeftShift: aByteArray
Private - Left shift by 1 place the bytes in aByteArray

bytesLeftShift: aByteArray big: totalShift
Private - Left shift the bytes in aByteArray by totalShift places

bytesLeftShift: aByteArray n: shift
Private - Left shift by shift places the bytes in aByteArray (shift <= 7)

bytesRightShift: aByteArray big: totalShift
Private - Right shift the bytes in aByteArray by totalShift places

bytesRightShift: bytes n: aNumber
Private - Right shift the bytes in 'bytes' by 'aNumber' places (shift <= 7)

bytesTrailingZeros: bytes
Private - Answer the number of trailing zero bits in the receiver

primDivide: rhs
Private - Implements Knuth's divide and correct algorithm from 'Seminumerical Algorithms' 3rd Edition, section 4.3.1 (which is basically an enhanced version of the divide 'algorithm' for two-digit divisors which is taught in primary school!!!)

1.101.4 LargePositiveInteger: numeric testing

abs Answer the receiver's absolute value

negative Answer whether the receiver is < 0

positive Answer whether the receiver is >= 0

sign Answer the receiver's sign

strictlyPositive
Answer whether the receiver is > 0

1.101.5 LargePositiveInteger: primitive operations

divide: aNumber using: aBlock
Private - Divide the receiver by aNumber (unsigned division). Evaluate aBlock passing the result ByteArray, the remainder ByteArray, and whether the division had a remainder

isSmall Private - Answer whether the receiver is small enough to employ simple scalar algorithms for division and multiplication

multiply: aNumber
Private - Multiply the receiver by aNumber (unsigned multiply)

1.102 LargeWordArray

Defined in namespace Smalltalk
Superclass: LargeArrayedCollection
Category: Collections-Sequenceable
I am similar to a plain WordArray, but I'm specially designed to save memory when lots of items are zero.

1.102.1 LargeWordArray: overridden

defaultElement
Answer the value which is hoped to be the most common in the array

newCollection: size
Create a WordArray of the given size

1.103 LargeZeroInteger

Defined in namespace Smalltalk
Superclass: LargePositiveInteger
Category: Language-Data types

I am quite a strange class. Indeed, the concept of a "large integer" that is zero is a weird one. Actually my only instance is zero but is represented like LargeIntegers, has the same generality as LargeIntegers, and so on. That only instance is stored in the class variable Zero, and is used in arithmetical methods, when we have to coerce a parameter that is zero.

1.103.1 LargeZeroInteger: accessing

at: anIndex
Answer '0'.

hash Answer '0'.

size Answer '0'.

1.103.2 LargeZeroInteger: arithmetic

*** aNumber**
Multiply aNumber and the receiver, answer the result

+ aNumber
Sum the receiver and aNumber, answer the result

- aNumber
Subtract aNumber from the receiver, answer the result

/ aNumber
Divide aNumber and the receiver, answer the result (an Integer or Fraction)

// aNumber
Divide aNumber and the receiver, answer the result truncated towards -infinity

\\ aNumber
Divide aNumber and the receiver, answer the remainder truncated towards -infinity

quo: aNumber
Divide aNumber and the receiver, answer the result truncated towards 0

rem: aNumber
Divide aNumber and the receiver, answer the remainder truncated towards 0

1.103.3 LargeZeroInteger: numeric testing

sign Answer the receiver's sign

strictlyPositive
Answer whether the receiver is > 0

1.103.4 LargeZeroInteger: printing

replace: str withStringBase: radix
Return in a string the base radix representation of the receiver.

1.104 Link

Defined in namespace Smalltalk
Superclass: Object
Category: Collections-Sequenceable
I represent simple linked lists. Generally, I am not used by myself, but rather a subclass adds other instance variables that hold the information for each node, and I hold the glue that keeps them together.

1.104.1 Link class: instance creation

nextLink: aLink
Create an instance with the given next link

1.104.2 Link: basic

nextLink Answer the next item in the list

nextLink: aLink
Set the next item in the list

1.104.3 Link: iteration

at: index Retrieve a node (instance of Link) that is at a distance of 'index' after the receiver.

at: index put: object
This method should not be called for instances of this class.

do: aBlock
Evaluate aBlock for each element in the list

size Answer the number of elements in the list. Warning: this is O(n)

1.105 LinkedList

Defined in namespace Smalltalk
Superclass: SequenceableCollection
Category: Collections-Sequenceable
I provide methods that access and manipulate linked lists. I assume that the elements of the linked list are subclasses of Link, because I use the methods that class Link supplies to implement my methods.

1.105.1 LinkedList: accessing

at: index Return the element that is index into the linked list.

at: index put: object
This method should not be called for instances of this class.

1.105.2 LinkedList: adding

add: aLink
Add aLink at the end of the list; return aLink.

addFirst: aLink
Add aLink at the head of the list; return aLink.

addLast: aLink
Add aLink at then end of the list; return aLink.

remove: aLink ifAbsent: aBlock
Remove aLink from the list and return it, or invoke aBlock if it's not found in the list.

removeFirst
Remove the first element from the list and return it, or error if the list is empty.

removeLast
Remove the final element from the list and return it, or error if the list is empty.

1.105.3 LinkedList: enumerating

do: aBlock
Enumerate each object in the list, passing it to aBlock (actual behavior might depend on the subclass of Link that is being used).

identityIncludes: anObject
Answer whether we include the anObject object

includes: anObject
Answer whether we include anObject

1.105.4 LinkedList: iteration

first Retrieve the first element of the list and return it, or error if the list is empty.

last Retrieve the last element of the list and return it, or error if the list is empty.

1.105.5 LinkedList: testing

isEmpty Returns true if the list contains no members

notEmpty Returns true if the list contains at least a member

size Answer the number of elements in the list. Warning: this is O(n)

1.106 LookupKey

Defined in namespace Smalltalk
Superclass: Magnitude
Category: Language-Data types
I represent a key for looking up entries in a data structure. Subclasses of me, such as Association, typically represent dictionary entries.

1.106.1 LookupKey class: basic

key: aKey Answer a new instance of the receiver with the given key and value

1.106.2 LookupKey: accessing

key Answer the receiver's key

key: aKey Set the receiver's key to aKey

1.106.3 LookupKey: printing

printOn: aStream
Put on aStream a representation of the receiver

1.106.4 LookupKey: storing

storeOn: aStream
Put on aStream some Smalltalk code compiling to the receiver

1.106.5 LookupKey: testing

< aLookupKey
Answer whether the receiver's key is less than aLookupKey's

= aLookupKey
Answer whether the receiver's key and value are the same as aLookupKey's, or false if aLookupKey is not an instance of the receiver

hash Answer an hash value for the receiver

1.107 LookupTable

Defined in namespace Smalltalk
Superclass: Dictionary
Category: Collections-Keyed
I am a more efficient variant of Dictionary that cannot be used as a pool dictionary of variables, as I don't use Associations to store key-value pairs. I also cannot have nil as a key; if you need to be able to store nil as a key, use Dictionary instead. I use the object equality comparison message #= to determine equivalence of indices.

1.107.1 LookupTable class: instance creation

new Create a new LookupTable with a default size

1.107.2 LookupTable: accessing

add: anAssociation
Add the anAssociation key to the receiver

associationAt: key ifAbsent: aBlock
Answer the key/value Association for the given key. Evaluate aBlock (answering the result) if the key is not found

at: key ifAbsent: aBlock
Answer the value associated to the given key, or the result of evaluating aBlock if the key is not found

at: aKey ifPresent: aBlock
If aKey is absent, answer nil. Else, evaluate aBlock passing the associated value and answer the result of the invocation

at: key put: value
Store value as associated to the given key

1.107.3 LookupTable: enumerating

associationsDo: aBlock
Pass each association in the LookupTable to aBlock.

do: aBlock
Pass each value in the LookupTable to aBlock.

keysAndValuesDo: aBlock
Pass each key/value pair in the LookupTable as two distinct parameters to aBlock.

keysDo: aBlock
Pass each key in the LookupTable to aBlock.

1.107.4 LookupTable: hashing

hash Answer the hash value for the receiver

1.107.5 LookupTable: rehashing

rehash Rehash the receiver

1.107.6 LookupTable: removing

remove: anAssociation
Remove anAssociation's key from the dictionary

remove: anAssociation ifAbsent: aBlock
Remove anAssociation's key from the dictionary

removeKey: key ifAbsent: aBlock
Remove the passed key from the LookupTable, answer the result of evaluating aBlock if it is not found

1.107.7 LookupTable: storing

storeOn: aStream
Print Smalltalk code compiling to the receiver on aStream

1.108 Magnitude

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Data types

I am an abstract class. My objects represent things that are discrete and map to a number line. My instances can be compared with < and >.

1.108.1 Magnitude: basic

< aMagnitude
Answer whether the receiver is less than aMagnitude

<= aMagnitude
Answer whether the receiver is less than or equal to aMagnitude

= aMagnitude
Answer whether the receiver is equal to aMagnitude

> aMagnitude
Answer whether the receiver is greater than aMagnitude

>= aMagnitude
Answer whether the receiver is greater than or equal to aMagnitude

1.108.2 Magnitude: misc methods

between: min and: max
Returns true if object is inclusively between min and max.

max: aMagnitude
Returns the greatest object between the receiver and aMagnitude

min: aMagnitude
Returns the least object between the receiver and aMagnitude

1.109 MappedCollection

Defined in namespace Smalltalk
Superclass: Collection
Category: Collections-Keyed

I represent collections of objects that are indirectly indexed by names. There are really two collections involved: domain and a map. The map maps between external names and indices into domain, which contains the real association. In order to work properly, the domain must be an instance of a subclass of SequenceableCollection, and the map must be an instance of Dictionary, or of a subclass of SequenceableCollection.

As an example of using me, consider implenting a Dictionary whose elements are indexed. The domain would be a SequenceableCollection with n elements, the map a Dictionary associating each key to an index in the domain. To access by key, to perform enumeration, etc. you would ask an instance of me; to access by index, you would access the domain directly.

Another idea could be to implement row access or column access to a matrix implemented as a single n*m Array: the Array would be the domain, while the map would be an Interval.

1.109.1 MappedCollection class: instance creation

collection: aCollection map: aMap
Answer a new MappedCollection using the given domain (aCollection) and map

new This method should not be used; instead, use #collection:map: to create MappedCollection.

1.109.2 MappedCollection: basic

add: anObject
This method should not be called for instances of this class.

at: key Answer the object at the given key

at: key put: value
Store value at the given key

atAll: keyCollection
Answer a new MappedCollection that only includes the given keys. The new MappedCollection might use keyCollection or consecutive integers for the keys, depending on the map's type. Fail if any of them is not found in the map.

collect: aBlock
Answer a Collection with the same keys as the map, where accessing a key yields the value obtained by passing through aBlock the value accessible from the key in the receiver. The result need not be another MappedCollection

contents Answer a bag with the receiver's values

copyFrom: a to: b
Answer a new collection containing all the items in the receiver from the a-th to the b-th.

do: aBlock
Evaluate aBlock for each object

domain Answer the receiver's domain

keys Answer the keys that can be used to access this collection.

keysAndValuesDo: aBlock
Evaluate aBlock passing two arguments, one being a key that can be used to access this collection, and the other one being the value.

keysDo: aBlock
Evaluate aBlock on the keys that can be used to access this collection.

map Answer the receiver's map

reject: aBlock
Answer the objects in the domain for which aBlock returns false

select: aBlock
Answer the objects in the domain for which aBlock returns true

size Answer the receiver's size

1.110 Memory

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation
I provide access to actual machine addresses of OOPs and objects. I have no instances; you send messages to my class to map between an object and the address of its OOP or object. In addition I provide direct memory access with different C types (ints, chars, OOPs, floats,...).

1.110.1 Memory class: accessing

at: anAddress
Access the Smalltalk object (OOP) at the given address.

at: anAddress put: aValue
Store a pointer (OOP) to the Smalltalk object identified by 'value' at the given address.

bigEndian Answer whether we're running on a big- or little-endian system.

charAt: anAddress
Access the C char at the given address. The value is returned as a Smalltalk Character.

charAt: anAddress put: aValue
Store as a C char the Smalltalk Character or Integer object identified by 'value', at the given address, using sizeof(char) bytes - i.e. 1 byte.

deref: anAddress
Access the C int pointed by the given address

doubleAt: anAddress
Access the C double at the given address.

doubleAt: anAddress put: aValue
Store the Smalltalk Float object identified by 'value', at the given address, writing it like a C double.

floatAt: anAddress
Access the C float at the given address.

floatAt: anAddress put: aValue
Store the Smalltalk Float object identified by 'value', at the given address, writing it like a C float.

intAt: anAddress
Access the C int at the given address.

intAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(int) bytes.

longAt: anAddress
Access the C long int at the given address.

longAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(long) bytes.

longDoubleAt: anAddress
Access the C long double at the given address.

longDoubleAt: anAddress put: aValue
Store the Smalltalk Float object identified by 'value', at the given address, writing it like a C long double.

shortAt: anAddress
Access the C short int at the given address.

shortAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(short) bytes.

stringAt: anAddress
Access the string pointed by the C 'char *' at the given given address.

stringAt: anAddress put: aValue
Store the Smalltalk String object identified by 'value', at the given address in memory, writing it like a *FRESHLY ALLOCATED* C string. It is the caller's responsibility to free it if necessary.

ucharAt: anAddress put: aValue
Store as a C char the Smalltalk Character or Integer object identified by 'value', at the given address, using sizeof(char) bytes - i.e. 1 byte.

uintAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(int) bytes.

ulongAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(long) bytes.

unsignedCharAt: anAddress
Access the C unsigned char at the given address. The value is returned as a Smalltalk Character.

unsignedCharAt: anAddress put: aValue
Store as a C char the Smalltalk Character or Integer object identified by 'value', at the given address, using sizeof(char) bytes - i.e. 1 byte.

unsignedIntAt: anAddress
Access the C unsigned int at the given address.

unsignedIntAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(int) bytes.

unsignedLongAt: anAddress
Access the C unsigned long int at the given address.

unsignedLongAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(long) bytes.

unsignedShortAt: anAddress
Access the C unsigned short int at the given address.

unsignedShortAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(short) bytes.

ushortAt: anAddress put: aValue
Store the Smalltalk Integer object identified by 'value', at the given address, using sizeof(short) bytes.

1.111 Message

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation
I represent a message send. My instances are created to hold a message that has failed, so that error reporting methods can examine the sender and arguments, but also to represent method attributes (like <primitive: 111> since their syntax is isomorphic to that of a message send.

1.111.1 Message class: creating instances

selector: aSymbol argument: anObject
Create a new Message with the given selector and argument

selector: aSymbol arguments: anArray
Create a new Message with the given selector and arguments

1.111.2 Message: accessing

argument Answer the first of the receiver's arguments

arguments Answer the receiver's arguments

arguments: anArray
Set the receiver's arguments

selector Answer the receiver's selector

selector: aSymbol
Set the receiver's selector

1.111.3 Message: basic

printAsAttributeOn: aStream
Print a representation of the receiver on aStream, modeling it after the source code for a attribute.

1.111.4 Message: printing

printOn: aStream
Print a representation of the receiver on aStream

reinvokeFor: aReceiver
Resend to aReceiver - present for compatibility

sendTo: aReceiver
Resend to aReceiver

1.112 MessageNotUnderstood

Defined in namespace Smalltalk
Superclass: Error
Category: Language-Exceptions
MessageNotUnderstood represents an error during message lookup. Signaling it is the default action of the #doesNotUnderstand: handler

1.112.1 MessageNotUnderstood: accessing

message Answer the message that wasn't understood

receiver Answer the object to whom the message send was directed

1.112.2 MessageNotUnderstood: description

description
Answer a textual description of the exception.

isResumable
Answer true. #doesNotUnderstand: exceptions are by default resumable.

1.113 Metaclass

Defined in namespace Smalltalk
Superclass: ClassDescription
Category: Language-Implementation
I am the root of the class hierarchy. My instances are metaclasses, one for each real class. My instances have a single instance, which they hold onto, which is the class that they are the metaclass of. I provide methods for creation of actual class objects from metaclass object, and the creation of metaclass objects, which are my instances. If this is confusing to you, it should be...the Smalltalk metaclass system is strange and complex.

1.113.1 Metaclass class: instance creation

subclassOf: superMeta
Answer a new metaclass representing a subclass of superMeta

1.113.2 Metaclass: accessing

instanceClass
Answer the only instance of the metaclass

primaryInstance
Answer the only instance of the metaclass - present for compatibility

soleInstance
Answer the only instance of the metaclass - present for compatibility

1.113.3 Metaclass: basic

name: className environment: aNamespace subclassOf: theSuperclass
Private - create a full featured class and install it, or change the superclass or shape of an existing one; instance variable names, class variable names and pool dictionaries are left untouched.

name: className environment: aNamespace subclassOf: newSuperclass instanceVariableArray: variableArray shape: shape classPool: classVarDict poolDictionaries: sharedPoolNames category: categoryName
Private - create a full featured class and install it, or change an existing one

name: newName environment: aNamespace subclassOf: theSuperclass instanceVariableNames: stringOfInstVarNames shape: shape classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryName
Private - parse the instance and class variables, and the pool dictionaries, then create the class.

newMeta: className environment: aNamespace subclassOf: theSuperclass instanceVariableArray: arrayOfInstVarNames shape: shape classPool: classVarDict poolDictionaries: sharedPoolNames category: categoryName
Private - create a full featured class and install it

1.113.4 Metaclass: compiling methods

poolResolution
Use my instance's poolResolution.

1.113.5 Metaclass: delegation

addClassVarName: aString
Add a class variable with the given name to the class pool dictionary

addSharedPool: aDictionary
Add the given shared pool to the list of the class' pool dictionaries

allClassVarNames
Answer the names of the variables in the receiver's class pool dictionary and in each of the superclasses' class pool dictionaries

allSharedPoolDictionariesDo: aBlock
Answer the shared pools visible from methods in the metaclass, in the correct search order.

allSharedPools
Return the names of the shared pools defined by the class and any of its superclasses

category Answer the class category

classPool Answer the class pool dictionary

classVarNames
Answer the names of the variables in the class pool dictionary

comment Answer the class comment

debuggerClass
Answer the debugger class that was set in the instance class

environment
Answer the namespace in which the receiver is implemented

name Answer the class name - it has none, actually

pragmaHandlerFor: aSymbol
Answer the (possibly inherited) registered handler for pragma aSymbol, or nil if not found.

removeClassVarName: aString
Removes the class variable from the class, error if not present, or still in use.

removeSharedPool: aDictionary
Remove the given dictionary to the list of the class' pool dictionaries

sharedPools
Return the names of the shared pools defined by the class

1.113.6 Metaclass: filing

fileOutOn: aFileStream
File out complete class description: class definition, class and instance methods

1.113.7 Metaclass: printing

nameIn: aNamespace
Answer the class name when the class is referenced from aNamespace.

printOn: aStream
Print a represention of the receiver on aStream

printOn: aStream in: aNamespace
Print on aStream the class name when the class is referenced from aNamespace.

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.113.8 Metaclass: testing functionality

asClass Answer 'instanceClass'.

isMetaclass
Answer 'true'.

1.114 MethodContext

Defined in namespace Smalltalk
Superclass: ContextPart
Category: Language-Implementation

My instances represent an actively executing method. They record various bits of information about the execution environment, and contain the execution stack.

1.114.1 MethodContext: accessing

home Answer the MethodContext to which the receiver refers (i.e. the receiver itself)

isBlock Answer whether the receiver is a block context

isDisabled Answers whether the receiver has actually ended execution and will be skipped when doing a return. BlockContexts are removed from the chain whenever a non-local return is done, but MethodContexts need to stay there in case there is a non-local return from the #ensure: block.

isEnvironment
To create a valid execution environment for the interpreter even before it starts, GST creates a fake context which invokes a special "termination" method. Such a context can be used as a marker for the current execution environment. Answer whether the receiver is that kind of context.

isUnwind Answers whether the context must continue execution even after a non-local return (a return from the enclosing method of a block, or a call to the #continue: method of ContextPart). Such contexts are created only by #ensure:.

mark To create a valid execution environment for the interpreter even before it starts, GST creates a fake context which invokes a special "termination" method. A similar context is created by #valueWithUnwind, by using this method.

sender Return the context from which the receiver was sent

1.114.2 MethodContext: debugging

isInternalExceptionHandlingContext
Answer whether the receiver is a context that should be hidden to the user when presenting a backtrace. Such contexts are identified through the #exceptionHandlingInternal: attribute: if there is such a context in the backtrace, all those above it are marked as internal.

That is, the attribute being set to true means that the context and all those above it are to be hidden, while the attribute being set to false means that the contexts above it must be hidden, but not the context itself.

1.114.3 MethodContext: printing

printOn: aStream
Print a representation for the receiver on aStream

1.115 MethodDictionary

Defined in namespace Smalltalk
Superclass: IdentityDictionary
Category: Language-Implementation
I am similar to an IdentityDictionary, except that removal and rehashing operations inside my instances look atomic to the interpreter.

1.115.1 MethodDictionary: adding

at: key put: value
Store value as associated to the given key

1.115.2 MethodDictionary: rehashing

rehash Rehash the receiver

1.115.3 MethodDictionary: removing

remove: anAssociation
Remove anAssociation's key from the dictionary

removeKey: anElement ifAbsent: aBlock
Remove the passed key from the dictionary, answer the result of evaluating aBlock if it is not found

1.116 MethodInfo

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation
I provide information about particular methods. I can produce the category that a method was filed under, and can be used to access the source code of the method.

1.116.1 MethodInfo: accessing

category Answer the method category

category: aCategory
Set the method category

methodClass
Answer the class in which the method is defined

methodClass: aClass
Set the class in which the method is defined

selector Answer the selector through which the method is called

selector: aSymbol
Set the selector through which the method is called

sourceCode
Answer a FileSegment or String or nil containing the method source code

sourceFile Answer the name of the file where the method source code is

sourcePos Answer the starting position of the method source code in the sourceFile

sourceString
Answer a String containing the method source code

stripSourceCode
Remove the reference to the source code for the method

1.116.2 MethodInfo: equality

= aMethodInfo
Compare the receiver and aMethodInfo, answer whether they're equal

hash Answer an hash value for the receiver

1.117 Namespace

Defined in namespace Smalltalk
Superclass: AbstractNamespace
Category: Language-Implementation
I am a Namespace that has a super-namespace.

1.117.1 Namespace class: accessing

current Answer the current namespace

current: aNamespaceOrClass
Set the current namespace to be aNamespace or, if it is a class, its class pool (the Dictionary that holds class variables).

1.117.2 Namespace class: disabling instance creation

new Disabled - use #addSubspace: to create instances

new: size Disabled - use #addSubspace: to create instances

1.117.3 Namespace class: initialization

initialize This actually is not needed, the job could be done in dict.c (function namespace_new). But I'm lazy and I prefer to rely on the Smalltalk implementation of IdentitySet.

1.117.4 Namespace: accessing

inheritedKeys
Answer a Set of all the keys in the receiver and its superspaces

1.117.5 Namespace: namespace hierarchy

siblings Answer all the other namespaces that inherit from the receiver's superspace.

siblingsDo: aBlock
Evaluate aBlock once for each of the other namespaces that inherit from the receiver's superspace, passing the namespace as a parameter.

1.117.6 Namespace: overrides for superspaces

associationAt: key ifAbsent: aBlock
Return the key/value pair associated to the variable named as specified by 'key'. If the key is not found search will be brought on in superspaces, finally evaluating aBlock if the variable cannot be found in any of the superspaces.

associationsDo: aBlock
Pass each association in the namespace to aBlock

at: key ifAbsent: aBlock
Return the value associated to the variable named as specified by 'key'. If the key is not found search will be brought on in superspaces, finally evaluating aBlock if the variable cannot be found in any of the superspaces.

at: key ifPresent: aBlock
If aKey is absent from the receiver and all its superspaces, answer nil. Else, evaluate aBlock passing the associated value and answer the result of the invocation

do: aBlock
Pass each value in the namespace to aBlock

includesKey: key
Answer whether the receiver or any of its superspaces contain the given key

keysAndValuesDo: aBlock
Pass to aBlock each of the receiver's keys and values, in two separate parameters

keysDo: aBlock
Pass to aBlock each of the receiver's keys

set: key to: newValue ifAbsent: aBlock
Assign newValue to the variable named as specified by 'key'. This method won't define a new variable; instead if the key is not found it will search in superspaces and evaluate aBlock if it is not found. Answer newValue.

size Answer the number of keys in the receiver and each of its superspaces

1.117.7 Namespace: printing

nameIn: aNamespace
Answer Smalltalk code compiling to the receiver when the current namespace is aNamespace

printOn: aStream in: aNamespace
Print on aStream some Smalltalk code compiling to the receiver when the current namespace is aNamespace

storeOn: aStream
Store Smalltalk code compiling to the receiver

1.118 NetClients.URIResolver

Defined in namespace Smalltalk.NetClients
Superclass: Object
Category: NetClients-URIResolver
This class publishes methods to download files from the Internet.

1.118.1 NetClients.URIResolver class: api

openOn: aURI
Always raise an error, as this method is not supported without loading the additional NetClients package.

openOn: aURI ifFail: aBlock
Always evaluate aBlock and answer the result if the additional NetClients package is not loaded. If it is, instead, return a WebEntity with the contents of the resource specified by anURI, and only evaluate the block if loading the resource fails.

openStreamOn: aURI
Check if aURI can be fetched from the Internet or from the local system, and if so return a Stream with its contents. If this is not possible, raise an exception.

openStreamOn: aURI ifFail: aBlock
Check if aURI can be fetched from the Internet or from the local system, and if so return a Stream with its contents. If this is not possible, instead, evaluate the zero-argument block aBlock and answer the result of the evaluation.

1.118.2 NetClients.URIResolver class: instance creation

on: anURL
Answer a new URIResolver that will do its best to fetch the data for anURL from the Internet.

1.119 NetClients.URL

Defined in namespace Smalltalk.NetClients
Superclass: Object
Category: NetClients-URIResolver
Copyright (c) Kazuki Yasumatsu, 1995. All rights reserved.

1.119.1 NetClients.URL class: encoding URLs

decode: aString
Decode a text/x-www-form-urlencoded String into a text/plain String.

encode: anURL
Encode a text/plain into a text/x-www-form-urlencoded String (those things with lots of % in them).

initialize Initialize the receiver's class variables.

1.119.2 NetClients.URL class: instance creation

fromString: aString
Parse the given URL and answer an URL object based on it.

new Answer a 'blank' URL.

scheme: schemeString host: hostString path: pathString
Answer an URL object made from all the parts passed as arguments.

scheme: schemeString host: hostString port: portNumber path: pathString
Answer an URL object made from all the parts passed as arguments.

scheme: schemeString path: pathString
Answer an URL object made from all the parts passed as arguments.

scheme: schemeString username: userString password: passwordString host: hostString port: portNumber path: pathString
Answer an URL object made from all the parts passed as arguments.

1.119.3 NetClients.URL: accessing

asString Answer the full request string corresponding to the URL. This is how the URL would be printed in the address bar of a web browser, except that the query data is printed even if it is to be sent through a POST request.

decodedFields
Convert the form fields to a Dictionary, answer nil if no question mark is found in the URL.

decodedFile
Answer the file part of the URL, decoding it from x-www-form-urlencoded format.

decodedFragment
Answer the fragment part of the URL, decoding it from x-www-form-urlencoded format.

fragment Answer the fragment part of the URL, leaving it in x-www-form-urlencoded format.

fragment: aString
Set the fragment part of the URL, which should be in x-www-form-urlencoded format.

fullRequestString
Answer the full request string corresponding to the URL. This is how the URL would be printed in the address bar of a web browser, except that the query data is printed even if it is to be sent through a POST request.

hasPostData
Answer whether the URL has a query part but is actually for an HTTP POST request and not really part of the URL (as it would be for the HTTP GET request).

hasPostData: aBoolean
Set whether the query part of the URL is actually the data for an HTTP POST request and not really part of the URL (as it would be for the HTTP GET request).

host Answer the host part of the URL.

host: aString
Set the host part of the URL to aString.

newsGroup
If the receiver is an nntp url, return the news group.

password Answer the password part of the URL.

password: aString
Set the password part of the URL to aString.

path Answer the path part of the URL.

path: aString
Set the path part of the URL to aString.

port Answer the port number part of the URL.

port: anInteger
Set the port number part of the URL to anInteger.

postData Answer whether the URL has a query part and it is meant for an HTTP POST request, answer it. Else answer nil.

postData: aString
Associate to the URL some data that is meant to be sent through an HTTP POST request, answer it.

query Answer the query data associated to the URL.

query: aString
Set the query data associated to the URL to aString.

requestString
Answer the URL as it would be sent in an HTTP stream (that is, the path and the query data, the latter only if it is to be sent with an HTTP POST request).

scheme Answer the URL's scheme.

scheme: aString
Set the URL's scheme to be aString.

username Answer the username part of the URL.

username: aString
Set the username part of the URL to aString.

1.119.4 NetClients.URL: comparing

= anURL Answer whether the two URLs are equal. The file and anchor are converted to full 8-bit ASCII (contrast with urlencoded) and the comparison is case-sensitive; on the other hand, the protocol and host are compared without regard to case.

hash Answer an hash value for the receiver

1.119.5 NetClients.URL: copying

copyWithoutAuxiliaryParts
: Answer a copy of the receiver where the fragment and query parts of the URL have been cleared.

copyWithoutFragment
: Answer a copy of the receiver where the fragment parts of the URL has been cleared.

postCopy
: All the variables are copied when an URL object is copied.

1.119.6 NetClients.URL: initialize-release

initialize
: Initialize the object to a consistent state.

1.119.7 NetClients.URL: printing

printOn: stream
: Print a representation of the URL on the given stream.

1.119.8 NetClients.URL: still unclassified

contents
: Not commented.

entity
: Not commented.

readStream
: Not commented.

1.119.9 NetClients.URL: testing

canCache
: Answer whether the URL is cacheable. The current implementation considers file URLs not to be cacheable, and everything else to be.

hasFragment
: Answer whether the URL points to a particular fragment (anchor) of the resource.

hasQuery
: Answer whether the URL includes query arguments to be submitted when retrieving the resource.

isFileScheme
: Answer whether the URL is a file URL.

isFragmentOnly
: Answer whether the URL only includes the name of a particular fragment (anchor) of the resource to which it refers.

1.119.10 NetClients.URL: utilities

construct: anURL
: Construct an absolute URL based on the relative URL anURL and the base path represented by the receiver

1.120 Notification

Defined in namespace Smalltalk
Superclass: Exception
Category: Language-Exceptions

Notification represents a resumable, exceptional yet non-erroneous, situation. Signaling a notification in absence of an handler simply returns nil.

1.120.1 Notification: exception description

defaultAction
Do the default action for notifications, which is to resume execution of the context which signaled the exception.

description
Answer a textual description of the exception.

isResumable
Answer true. Notification exceptions are by default resumable.

1.121 NullProxy

Defined in namespace Smalltalk
Superclass: AlternativeObjectProxy
Category: Streams-Files

I am a proxy that does no special processing on the object to be saved. I can be used to disable proxies for particular subclasses. My subclasses add to the stored information, but share the fact that the format is about the same as that of #dump: without a proxy.

1.121.1 NullProxy class: instance creation

loadFrom: anObjectDumper
Reload the object stored in anObjectDumper

1.121.2 NullProxy: accessing

dumpTo: anObjectDumper
Dump the object stored in the proxy to anObjectDumper

1.122 NullValueHolder

Defined in namespace Smalltalk
Superclass: ValueAdaptor
Category: Language-Data types

I pretend to store my value in a variable, but I don't actually. You can use the only instance of my class (returned by 'ValueHolder null') if you're not interested in a value that is returned as described in ValueHolder's comment.

1.122.1 NullValueHolder class: creating instances

new Not used – use 'ValueHolder null' instead

uniqueInstance
Answer the sole instance of NullValueHolder

1.122.2 NullValueHolder: accessing

value Retrive the value of the receiver. Always answer nil

value: anObject
Set the value of the receiver. Do nothing, discard the value

1.123 Number

Defined in namespace Smalltalk
Superclass: Magnitude
Category: Language-Data types
I am an abstract class that provides operations on numbers, both floating point and integer. I provide some generic predicates, and supply the implicit type coercing code for binary operations.

1.123.1 Number class: converting

coerce: aNumber
Answer aNumber - whatever class it belongs to, it is good

readFrom: aStream
Answer the number read from the rest of aStream, converted to an instance of the receiver. If the receiver is number, the class of the result is undefined – but the result is good.

readFrom: aStream radix: anInteger
Answer the number read from the rest of aStream, converted to an instance of the receiver. If the receiver is number, the class of the result is undefined – but the result is good.

The exponent (for example 1.2e-1) is only parsed if anInteger is 10.

1.123.2 Number class: testing

isImmediate
Answer whether, if x is an instance of the receiver, x copy == x

1.123.3 Number: arithmetic

*** aNumber**
Subtract the receiver and aNumber, answer the result

+ aNumber
Sum the receiver and aNumber, answer the result

- aNumber
Subtract aNumber from the receiver, answer the result

/ aNumber
Divide the receiver by aNumber, answer the result (no loss of precision). Raise a ZeroDivide exception or return a valid (possibly infinite) continuation value if aNumber is zero.

// aNumber
Return the integer quotient of dividing the receiver by aNumber with truncation towards negative infinity. Raise a ZeroDivide exception if aNumber is zero

\\ aNumber
Return the remainder of dividing the receiver by aNumber with truncation towards negative infinity. Raise a ZeroDivide exception if aNumber is zero

quo: aNumber
Return the integer quotient of dividing the receiver by aNumber with truncation towards zero. Raise a ZeroDivide exception if aNumber is zero

reciprocal Return the reciprocal of the receiver

rem: aNumber
Return the remainder of dividing the receiver by aNumber with truncation towards zero. Raise a ZeroDivide exception if aNumber is zero

1.123.4 Number: coercion

asCNumber
Convert the receiver to a kind of number that is understood by the C call-out mechanism.

1.123.5 Number: comparing

max: aNumber
Answer the maximum between the receiver and aNumber. Redefine in subclasses if necessary to ensure that if either self or aNumber is a NaN, it is always answered.

min: aNumber
Answer the minimum between the receiver and aNumber. Redefine in subclasses if necessary to ensure that if either self or aNumber is a NaN, it is always answered.

1.123.6 Number: converting

asExactFraction
Return the receiver, converted to a Fraction retaining the exact value of the receiver.

asFloat Convert the receiver to an arbitrary subclass of Float

asFloatD This method's functionality should be implemented by subclasses of Number

asFloatE This method's functionality should be implemented by subclasses of Number

asFloatQ This method's functionality should be implemented by subclasses of Number

asFraction This method's functionality should be implemented by subclasses of Number

asNumber Answer the receiver, since it is already a number

asRectangle
Answer an empty rectangle whose origin is (self asPoint)

asScaledDecimal: n
Answer the receiver, converted to a ScaledDecimal object.

asScaledDecimal: denDigits radix: base scale: n
Answer the receiver, divided by base^denDigits and converted to a ScaledDecimal object.

asString Answer the receiver's #displayString, which should be a good enough conversion to String for a number.

coerce: aNumber
Answer aNumber, converted to an integer or floating-point number.

degreesToRadians
Convert the receiver to radians

generality Answer the receiver's generality

radiansToDegrees
Convert the receiver from radians to degrees

unity Coerce 1 to the receiver's class. The default implementation works, but is inefficient

zero Coerce 0 to the receiver's class. The default implementation works, but is inefficient

1.123.7 Number: copying

deepCopy Return the receiver - it's an immediate (immutable) object

shallowCopy
Return the receiver - it's an immediate (immutable) object

1.123.8 Number: error raising

arithmeticError: msg
Raise an ArithmeticError exception having msg as its message text.

zeroDivide
Raise a division-by-zero (ZeroDivide) exception whose dividend is the receiver.

1.123.9 Number: misc math

abs Answer the absolute value of the receiver

arcCos Answer the arc cosine of the receiver

arcCosh Answer the hyperbolic arc-cosine of the receiver.

arcSin Answer the arc sine of the receiver

arcSinh	Answer the hyperbolic arc-sine of the receiver.
arcTan	Answer the arc tangent of the receiver
arcTan: x	Answer the angle (measured counterclockwise) between (x, self) and a ray starting in (0, 0) and moving towards (1, 0) - i.e. 3 o'clock
arcTanh	Answer the hyperbolic arc-tangent of the receiver.
ceilingLog: radix	Answer (self log: radix) ceiling. Optimized to answer an integer.
cos	Answer the cosine of the receiver
cosh	Answer the hyperbolic cosine of the receiver.
estimatedLog	Answer an estimate of (self abs floorLog: 10). This method should be overridden by subclasses, but Number's implementation does not raise errors - simply, it gives a correct result, so it is slow.
exp	Answer e raised to the receiver
floorLog: radix	Answer (self log: radix) floor. Optimized to answer an integer.
ln	Answer log base e of the receiver
log	Answer log base 10 of the receiver
log: aNumber	Answer log base aNumber of the receiver
negated	Answer the negated of the receiver
positiveDifference: aNumber	Answer the positive difference of the receiver and aNumber, that is self - aNumber if it is positive, 0 otherwise.
raisedTo: aNumber	Return self raised to aNumber power
raisedToInteger: anInteger	Return self raised to the anInteger-th power
sin	Answer the sine of the receiver
sinh	Answer the hyperbolic sine of the receiver.
sqrt	Answer the square root of the receiver
squared	Answer the square of the receiver
tan	Answer the tangent of the receiver
tanh	Answer the hyperbolic tangent of the receiver.
withSignOf: aNumber	Answer the receiver, with its sign possibly changed to match that of aNumber.

1.123.10 Number: point creation

@ y Answer a new point whose x is the receiver and whose y is y

asPoint Answer a new point, self @ self

1.123.11 Number: retrying

retry: aSymbol coercing: aNumber
Coerce to the other number's class the one number between the receiver and aNumber which has the lowest, and retry calling aSymbol. aSymbol is supposed not to be #= or #~= (since those don't fail if aNumber is not a Number).

retryDifferenceCoercing: aNumber
Coerce to the other number's class the one number between the receiver and aNumber which has the lowest, and retry calling #-.

retryDivisionCoercing: aNumber
Coerce to the other number's class the one number between the receiver and aNumber which has the lowest, and retry calling #/.

retryEqualityCoercing: aNumber
Coerce to the other number's class the one number between the receiver and aNumber which has the lowest, and retry calling #=.

retryError
Raise an error—a retrying method was called with two arguments having the same generality.

retryInequalityCoercing: aNumber
Coerce to the other number's class the one number between the receiver and aNumber which has the lowest, and retry calling #~=.

retryMultiplicationCoercing: aNumber
Coerce to the other number's class the one number between the receiver and aNumber which has the lowest, and retry calling #*.

retryRelationalOp: aSymbol coercing: aNumber
Coerce to the other number's class the one number between the receiver and aNumber which has the lowest, and retry calling aSymbol (<, <=, >, >=).

retrySumCoercing: aNumber
Coerce to the other number's class the one number between the receiver and aNumber which has the lowest, and retry calling #+.

1.123.12 Number: shortcuts and iterators

to: stop Return an interval going from the receiver to stop by 1

to: stop by: step
Return an interval going from the receiver to stop with the given step

to: stop by: step collect: aBlock
Evaluate aBlock for each value in the interval going from the receiver to stop with the given step. The results are collected in an Array and returned.

to: stop by: step do: aBlock
Evaluate aBlock for each value in the interval going from the receiver to stop with the given step. Compiled in-line for integer literal steps, and for one-argument aBlocks without temporaries, and therefore not overridable.

to: stop collect: aBlock
Evaluate aBlock for each value in the interval going from the receiver to stop by 1. The results are collected in an Array and returned.

to: stop do: aBlock
Evaluate aBlock for each value in the interval going from the receiver to stop by 1. Compiled in-line for one-argument aBlocks without temporaries, and therefore not overridable.

1.123.13 Number: testing

closeTo: num
Answer whether the receiver can be considered sufficiently close to num (this is done by checking equality if num is not a number, and by checking with 0.01% tolerance if num is a number).

even Returns true if self is divisible by 2

isExact Answer whether the receiver performs exact arithmetic. Most numeric classes do (in fact the only exceptions is Float and its descendants), so the default is to answer true rather than calling #subclassResponsibility.

isFinite Answer whether the receiver represents a finite quantity. Most numeric classes are for finite quantities, so the default is to answer true rather than calling #subclassResponsibility.

isInfinite Answer whether the receiver represents an infinite quantity. Most numeric classes are for finite quantities, so the default is to answer false rather than calling #subclassResponsibility.

isNaN Answer whether the receiver is a Not-A-Number. Most numeric classes don't handle nans, so the default is to answer false rather than calling #subclassResponsibility.

isNumber Answer 'true'.

isRational Answer whether the receiver is rational - false by default

negative Answer whether the receiver is < 0

odd Returns true if self is not divisible by 2

positive Answer whether the receiver is >= 0

sign Returns the sign of the receiver.

strictlyPositive
Answer whether the receiver is > 0

1.123.14 Number: truncation and round off

asInteger Answer the receiver, rounded to the nearest integer

floor Return the integer nearest the receiver toward negative infinity.

fractionPart
Answer a number which, summed to the #integerPart of the receiver, gives the receiver itself.

integerPart
Answer the receiver, truncated towards zero

roundTo: aNumber
Answer the receiver, truncated to the nearest multiple of aNumber

rounded Returns the integer nearest the receiver

truncateTo: aNumber
Answer the receiver, truncated towards zero to a multiple of aNumber

truncated Answer the receiver, truncated towards zero

1.124 Object

Defined in namespace Smalltalk
Superclass: none
Category: Language-Implementation
I am the root of the Smalltalk class system. All classes in the system are subclasses of me.

1.124.1 Object class: initialization

dependencies
Answer a dictionary that associates an object with its dependents.

dependencies: anObject
Use anObject as the dictionary that associates an object with its dependents.

finalizableObjects
Answer a set of finalizable objects.

initialize Initialize the Dependencies dictionary to be a WeakKeyIdentityDictionary.

update: aspect
Do any global tasks for the ObjectMemory events.

1.124.2 Object: built ins

= arg Answer whether the receiver is equal to arg. The equality test is by default the same as that for identical objects. = must not fail; answer false if the receiver cannot be compared to arg

== arg Answer whether the receiver is the same object as arg. This is a very fast test and is called 'object identity'.

allOwners Return an Array of Objects that point to the receiver.

asOop Answer the object index associated to the receiver. The object index doesn't change when garbage collection is performed.

at: anIndex
Answer the index-th indexed instance variable of the receiver

at: anIndex put: value
Store value in the index-th indexed instance variable of the receiver

basicAt: anIndex
Answer the index-th indexed instance variable of the receiver. This method must not be overridden, override at: instead

basicAt: anIndex put: value
Store value in the index-th indexed instance variable of the receiver This method must not be overridden, override at:put: instead

basicPrint Print a basic representation of the receiver

basicSize Answer the number of indexed instance variable in the receiver

become: otherObject
Change all references to the receiver into references to otherObject. Depending on the implementation, references to otherObject might or might not be transformed into the receiver (respectively, 'two-way become' and 'one-way become'). Implementations doing one-way become answer the receiver (so that it is not lost). Most implementations doing two-way become answer otherObject, but this is not assured - so do answer the receiver for consistency. GNU Smalltalk does two-way become and answers otherObject, but this might change in future versions: programs should not rely on the behavior and results of #become: .

becomeForward: otherObject
Change all references to the receiver into references to otherObject. References to otherObject are not transformed into the receiver. Answer the receiver so that it is not lost.

changeClassTo: aBehavior
Mutate the class of the receiver to be aBehavior. Note: Tacitly assumes that the structure is the same for the original and new class!!

checkIndexableBounds: index
Private - Check the reason why an access to the given indexed instance variable failed

checkIndexableBounds: index ifAbsent: aBlock
Private - Check the reason why an access to the given indexed instance variable failed. Evaluate aBlock for an invalid index.

checkIndexableBounds: index put: object
Private - Check the reason why a store to the given indexed instance variable failed

class Answer the class to which the receiver belongs

halt Called to enter the debugger

hash Answer an hash value for the receiver. This hash value is ok for objects that do not redefine ==.

identityHash
Answer an hash value for the receiver. This method must not be overridden

instVarAt: index
Answer the index-th instance variable of the receiver. This method must not be overridden.

instVarAt: index put: value
Store value in the index-th instance variable of the receiver. This method must not be overridden.

isReadOnly
Answer whether the object's indexed instance variables can be written

isUntrusted
Answer whether the object is to be considered untrusted.

makeEphemeron
Make the object an 'ephemeron'. An ephemeron is marked after all other objects, and if no references are found to the key except from the object itself, it is sent the #mourn message.

makeFixed
Avoid that the receiver moves in memory across garbage collections.

makeReadOnly: aBoolean
Set whether the object's indexed instance variables can be written

makeUntrusted: aBoolean
Set whether the object is to be considered untrusted.

makeWeak
Make the object a 'weak' one. When an object is only referenced by weak objects, it is collected and the slots in the weak objects are changed to nils by the VM; the weak object is then sent the #mourn message.

mark: aSymbol
Private - use this method to mark code which needs to be reworked, removed, etc. You can then find all senders of #mark: to find all marked methods or you can look for all senders of the symbol that you sent to #mark: to find a category of marked methods.

nextInstance
Private - answer another instance of the receiver's class, or nil if the entire object table has been walked

notYetImplemented
Called when a method defined by a class is not yet implemented, but is going to be

perform: selectorOrMessageOrMethod
Send the unary message named selectorOrMessageOrMethod (if a Symbol) to the receiver, or the message and arguments it identifies (if a Message or DirectedMessage), or finally execute the method within the receiver (if a CompiledMethod). In the last case, the method need not reside on the hierarchy from the receiver's class to Object – it need not reside at all in a MethodDictionary, in fact – but doing bad things will compromise stability of the Smalltalk virtual machine (and don't blame anybody but yourself).

This method should not be overridden

perform: selectorOrMethod with: arg1
Send the message named selectorOrMethod (if a Symbol) to the receiver, passing arg1 to it, or execute the method within the receiver (if a CompiledMethod). In the latter case, the method need not reside on the hierarchy from the receiver's class to Object – it need not reside at all in a MethodDictionary, in fact – but doing bad things will compromise stability of the Smalltalk virtual machine (and don't blame anybody but yourself).

This method should not be overridden

perform: selectorOrMethod with: arg1 with: arg2
Send the message named selectorOrMethod (if a Symbol) to the receiver, passing arg1 and arg2 to it, or execute the method within the receiver (if a CompiledMethod). In the latter case, the method need not reside on the hierarchy from the receiver's class to Object – it need not reside at all in a MethodDictionary, in fact – but doing bad things will compromise stability of the Smalltalk virtual machine (and don't blame anybody but yourself).

This method should not be overridden

perform: selectorOrMethod with: arg1 with: arg2 with: arg3
Send the message named selectorOrMethod (if a Symbol) to the receiver, passing the other arguments to it, or execute the method within the receiver (if a CompiledMethod). In the latter case, the method need not reside on the hierarchy from the receiver's class to Object – it need not reside at all in a MethodDictionary, in fact – but doing bad things will compromise stability of the Smalltalk virtual machine (and don't blame anybody but yourself).

This method should not be overridden

perform: selectorOrMethod with: arg1 with: arg2 with: arg3 with: arg4
Send the message named selectorOrMethod (if a Symbol) to the receiver, passing the other arguments to it, or execute the method within the receiver (if a CompiledMethod). In the latter case, the method need not reside on the hierarchy from the receiver's class to Object – it need not reside at all in a MethodDictionary, in fact – but doing bad things will compromise stability of the Smalltalk virtual machine (and don't blame anybody but yourself).

This method should not be overridden

perform: selectorOrMethod withArguments: argumentsArray
Send the message named selectorOrMethod (if a Symbol) to the receiver, passing the elements of argumentsArray as parameters, or execute the method

within the receiver (if a CompiledMethod). In the latter case, the method need not reside on the hierarchy from the receiver's class to Object – it need not reside at all in a MethodDictionary, in fact – but doing bad things will compromise stability of the Smalltalk virtual machine (and don't blame anybody but yourself).

This method should not be overridden

primitiveFailed
Called when a VM primitive fails

shallowCopy
Returns a shallow copy of the receiver (the instance variables are not copied)

shouldNotImplement
Called when objects belonging to a class should not answer a selector defined by a superclass

size Answer the number of indexed instance variable in the receiver

subclassResponsibility
Called when a method defined by a class should be overridden in a subclass

tenure Move the object to oldspace.

1.124.3 Object: change and update

broadcast: aSymbol
Send the unary message aSymbol to each of the receiver's dependents

broadcast: aSymbol with: anObject
Send the message aSymbol to each of the receiver's dependents, passing anObject

broadcast: aSymbol with: arg1 with: arg2
Send the message aSymbol to each of the receiver's dependents, passing arg1 and arg2 as parameters

broadcast: aSymbol withArguments: anArray
Send the message aSymbol to each of the receiver's dependents, passing the parameters in anArray

broadcast: aSymbol withBlock: aBlock
Send the message aSymbol to each of the receiver's dependents, passing the result of evaluating aBlock with each dependent as the parameter

changed Send update: for each of the receiver's dependents, passing them the receiver

changed: aParameter
Send update: for each of the receiver's dependents, passing them aParameter

update: aParameter
Default behavior is to do nothing. Called by #changed and #changed:

1.124.4 Object: class type methods

species This method has no unique definition. Generally speaking, methods which always return the same type usually don't use #class, but #species. For example, a PositionableStream's species is the class of the collection on which it is streaming (used by upTo:, upToAll:, upToEnd). Stream uses species for obtaining the class of next:'s return value, Collection uses it in its #copyEmpty: message, which in turn is used by all collection-returning methods. An Interval's species is Array (used by collect:, select:, reject:, etc.).

yourself Answer the receiver

1.124.5 Object: compiler

literalEquals: anObject
Not commented.

literalHash
Not commented.

1.124.6 Object: conversion

asValue Answer a ValueHolder whose initial value is the receiver.

1.124.7 Object: copying

copy Returns a shallow copy of the receiver (the instance variables are not copied). The shallow copy receives the message postCopy and the result of postCopy is passed back.

deepCopy Returns a deep copy of the receiver (the instance variables are copies of the receiver's instance variables)

postCopy Performs any changes required to do on a copied object. This is the place where one could, for example, put code to replace objects with copies of the objects

1.124.8 Object: debugging

examine Print all the instance variables of the receiver on the Transcript

examineOn: aStream
Print all the instance variables of the receiver on aStream

inspect In a GUI environment, this opens a tool to examine and modify the receiver. In the default image, it just calls #examine.

validSize Answer how many elements in the receiver should be inspected

1.124.9 Object: dependents access

addDependent: anObject
Add anObject to the set of the receiver's dependents. Important: if an object has dependents, it won't be garbage collected.

dependents
Answer a collection of the receiver's dependents.

release Remove all of the receiver's dependents from the set and allow the receiver to be garbage collected.

removeDependent: anObject
Remove anObject to the set of the receiver's dependents. No problem if anObject is not in the set of the receiver's dependents.

1.124.10 Object: error raising

doesNotUnderstand: aMessage
Called by the system when a selector was not found. message is a Message containing information on the receiver

error: message
Display a walkback for the receiver, with the given error message. Signal an 'Error' exception.

halt: message
Display a walkback for the receiver, with the given error message. Signal an 'Halt' exception.

1.124.11 Object: finalization

addToBeFinalized
Arrange things so that #finalize is sent to the object when the garbage collector finds out there are only weak references to it.

finalize Do nothing by default

mourn This method is sent by the VM to weak and ephemeron objects when one of their fields is found out to be garbage collectable (this means, for weak objects, that there are no references to it from non-weak objects, and for ephemeron objects, that the only paths to the first instance variable pass through other instance variables of the same ephemeron). The default behavior is to do nothing.

removeToBeFinalized
Unregister the object, so that #finalize is no longer sent to the object when the garbage collector finds out there are only weak references to it.

1.124.12 Object: introspection

instVarNamed: aString
Answer the instance variable named aString in the receiver.

instVarNamed: aString put: anObject
Answer the instance variable named aString in the receiver.

1.124.13 Object: printing

basicPrintNl
Print a basic representation of the receiver, followed by a new line.

basicPrintOn: aStream
Print a represention of the receiver on aStream

display Print a represention of the receiver on the Transcript (stdout the GUI is not active). For most objects this is simply its #print representation, but for strings and characters, superfluous dollars or extra pair of quotes are stripped.

displayNl Print a represention of the receiver, then put a new line on the Transcript (stdout the GUI is not active). For most objects this is simply its #printNl representation, but for strings and characters, superfluous dollars or extra pair of quotes are stripped.

displayOn: aStream
Print a represention of the receiver on aStream. For most objects this is simply its #printOn: representation, but for strings and characters, superfluous dollars or extra pair of quotes are stripped.

displayString
Answer a String representing the receiver. For most objects this is simply its #printString, but for strings and characters, superfluous dollars or extra pair of quotes are stripped.

print Print a represention of the receiver on the Transcript (stdout the GUI is not active)

printNl Print a represention of the receiver on stdout, put a new line the Transcript (stdout the GUI is not active)

printOn: aStream
Print a represention of the receiver on aStream

printString
Answer a String representing the receiver

1.124.14 Object: relational operators

~= anObject
Answer whether the receiver and anObject are not equal

~~ anObject
Answer whether the receiver and anObject are not the same object

1.124.15 Object: saving and loading

binaryRepresentationObject
This method must be implemented if PluggableProxies are used with the receiver's class. The default implementation raises an exception.

postLoad Called after loading an object; must restore it to the state before 'preStore' was called. Do nothing by default

postStore Called after an object is dumped; must restore it to the state before 'preStore' was called. Call #postLoad by default

preStore Called before dumping an object; it must *change* it (it must not answer a new object) if necessary. Do nothing by default

reconstructOriginalObject
Used if an instance of the receiver's class is returned as the #binaryRepresentationObject of another object. The default implementation raises an exception.

1.124.16 Object: storing

store Put a String of Smalltalk code compiling to the receiver on the Transcript (stdout the GUI is not active)

storeLiteralOn: aStream
Put a Smalltalk literal compiling to the receiver on aStream

storeNl Put a String of Smalltalk code compiling to the receiver, followed by a new line, on the Transcript (stdout the GUI is not active)

storeOn: aStream
Put Smalltalk code compiling to the receiver on aStream

storeString
Answer a String of Smalltalk code compiling to the receiver

1.124.17 Object: syntax shortcuts

-> anObject
Creates a new instance of Association with the receiver being the key and the argument becoming the value

1.124.18 Object: testing functionality

ifNil: nilBlock
Evaluate nilBlock if the receiver is nil, else answer self

ifNil: nilBlock ifNotNil: notNilBlock
Evaluate nilBlock if the receiver is nil, else evaluate notNilBlock, passing the receiver.

ifNotNil: notNilBlock
Evaluate notNilBlock if the receiver is not nil, passing the receiver. Else answer nil.

ifNotNil: notNilBlock ifNil: nilBlock
Evaluate nilBlock if the receiver is nil, else evaluate notNilBlock, passing the receiver.

isArray Answer 'false'.

isBehavior Answer 'false'.

isCObject Answer 'false'.

isCharacter
Answer 'false'.

isCharacterArray
Answer 'false'.

isClass Answer ‘false’.

isFloat Answer ‘false’.

isInteger Answer ‘false’.

isKindOf: aClass
Answer whether the receiver’s class is aClass or a subclass of aClass

isMemberOf: aClass
Returns true if the receiver is an instance of the class ’aClass’

isMeta Same as isMetaclass

isMetaClass
Same as isMetaclass

isMetaclass
Answer ‘false’.

isNamespace
Answer ‘false’.

isNil Answer whether the receiver is nil

isNumber Answer ‘false’.

isSmallInteger
Answer ‘false’.

isString Answer ‘false’.

isSymbol Answer ‘false’.

notNil Answer whether the receiver is not nil

respondsTo: aSymbol
Returns true if the receiver understands the given selector

1.124.19 Object: VM callbacks

badReturnError
Called back when a block performs a bad return.

mustBeBoolean
Called by the system when ifTrue:*, ifFalse:*, and: or or: are sent to anything but a boolean

noRunnableProcess
Called back when all processes are suspended

userInterrupt
Called back when the user presses Ctrl-Break

1.125 ObjectDumper

Defined in namespace Smalltalk
Superclass: Stream
Category: Streams-Files

I'm not part of a normal Smalltalk system, but most Smalltalks provide a similar feature: that is, support for storing objects in a binary format; there are many advantages in using me instead of #storeOn: and the Smalltalk compiler.

The data is stored in a very compact format, which has the side effect of making loading much faster when compared with compiling the Smalltalk code prepared by #storeOn:. In addition, my instances support circular references between objects, while #storeOn: supports it only if you know of such references at design time and you override #storeOn: to deal with them

1.125.1 ObjectDumper class: establishing proxy classes

disableProxyFor: aClass
Disable proxies for instances of aClass and its descendants

hasProxyFor: aClass
Answer whether a proxy class has been registered for instances of aClass.

proxyClassFor: anObject
Answer the class of a valid proxy for an object, or nil if none could be found

proxyFor: anObject
Answer a valid proxy for an object, or the object itself if none could be found

registerProxyClass: aProxyClass for: aClass
Register the proxy class aProxyClass - descendent of DumperProxy - to be used for instances of aClass and its descendants

1.125.2 ObjectDumper class: instance creation

new This method should not be called for instances of this class.

on: aFileStream
Answer an ObjectDumper working on aFileStream.

1.125.3 ObjectDumper class: shortcuts

dump: anObject to: aFileStream
Dump anObject to aFileStream. Answer anObject

loadFrom: aFileStream
Load an object from aFileStream and answer it

1.125.4 ObjectDumper class: testing

example This is a real torture test: it outputs recursive objects, identical objects multiple times, classes, metaclasses, integers, characters and proxies (which is also a test of more complex objects)!

1.125.5 ObjectDumper: accessing

flush 'Forget' any information on previously stored objects.

stream Answer the ByteStream to which the ObjectDumper will write and from which it will read.

stream: aByteStream
Set the ByteStream to which the ObjectDumper will write and from which it will read.

1.125.6 ObjectDumper: loading/dumping objects

dump: anObject
Dump anObject on the stream associated with the receiver. Answer anObject

load Load an object from the stream associated with the receiver and answer it

1.125.7 ObjectDumper: stream interface

atEnd Answer whether the underlying stream is at EOF

next Load an object from the underlying stream

nextPut: anObject
Store an object on the underlying stream

1.126 ObjectMemory

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation
I provide a few methods that enable one to tune the virtual machine's usage of memory. In addition, I can signal to my dependants some 'events' that can happen during the virtual machine's life.

ObjectMemory has both class-side and instance-side methods. In general, class-side methods provide means to tune the parameters of the memory manager, while instance-side methods are used together with the #current class-side method to take a look at statistics on the memory manager's state.

1.126.1 ObjectMemory class: accessing

current Return a snapshot of the VM's memory management statistics.

1.126.2 ObjectMemory class: builtins

abort Quit the Smalltalk environment, dumping core.

addressOf: anObject
Returns the address of the actual object that anObject references. Note that, with the exception of fixed objects this address is only valid until the next garbage collection; thus it's pretty risky to count on the address returned by this method for very long.

addressOfOOP: anObject
Returns the address of the OOP (object table slot) for anObject. The address is an Integer and will not change over time (i.e. is immune from garbage collector action) except if the virtual machine is stopped and restarted.

bigObjectThreshold
Answer the smallest size for objects that are allocated outside the main heap in the hope of providing more locality of reference between small objects.

bigObjectThreshold: bytes
Set the smallest size for objects that are allocated outside the main heap in the hope of providing more locality of reference between small objects. bytes must be a positive SmallInteger.

compact
Force a full garbage collection, including compaction of oldspace

finishIncrementalGC
Do a step in the incremental garbage collection.

gcMessage
Answer whether messages indicating that garbage collection is taking place are printed on stdout

gcMessage: aBoolean
Set whether messages indicating that garbage collection is taking place are printed on stdout

globalGarbageCollect
Force a full garbage collection

growThresholdPercent
Answer the percentage of the amount of memory used by the system grows which has to be full for the system to allocate more memory

growThresholdPercent: growPercent
Set the percentage of the amount of memory used by the system grows which has to be full for the system to allocate more memory

growTo: numBytes
Grow the amount of memory used by the system grows to numBytes.

incrementalGCStep
Do a step in the incremental garbage collection.

quit
Quit the Smalltalk environment. Whether files are closed and other similar cleanup occurs depends on the platform

quit: exitStatus
Quit the Smalltalk environment, passing the exitStatus integer to the OS. Files are closed and other similar cleanups occur.

scavenge
Force a minor garbage collection

smoothingFactor
Answer the factor (between 0 and 1) used to smooth the statistics provided by the virtual machine about memory handling. 0 disables updating the averages, 1 disables the smoothing (the statistics return the last value).

smoothingFactor: rate
Set the factor (between 0 and 1) used to smooth the statistics provided by the virtual machine about memory handling. 0 disables updating the averages, 1 disables the smoothing (the statistics return the last value).

spaceGrowRate
Answer the rate with which the amount of memory used by the system grows

spaceGrowRate: rate
Set the rate with which the amount of memory used by the system grows

1.126.3 ObjectMemory class: initialization

changed: aSymbol
Not commented.

initialize Initialize the globals

1.126.4 ObjectMemory class: saving the image

snapshot Save a snapshot on the image file that was loaded on startup.

snapshot: aString
Save an image on the aString file

1.126.5 ObjectMemory: accessing

allocFailures
Answer the number of times that the old-space allocator found no block that was at least as big as requested, and had to ask the operating system for more memory.

allocMatches
Answer the number of times that the old-space allocator found a block that was exactly as big as requested.

allocProbes
Answer the number of free blocks that the old-space allocator had to examine so far to allocate all the objects that are in old-space

allocSplits Answer the number of times that the old-space allocator could not find a block that was exactly as big as requested, and had to split a larger free block in two parts.

bytesPerOOP
Answer the number of bytes that is taken by an ordinary object pointer (in practice, a field such as a named instance variable).

bytesPerOTE
Answer the number of bytes that is taken by an object table entry (in practice, the overhead incurred by every object in the system, with the sole exception of SmallIntegers).

edenSize Answer the number of bytes in the 'eden' area of the young generation (in practice, the number of allocated bytes between two scavenges).

edenUsedBytes
Answer the number of bytes that are currently filled in the 'eden' area of the young generation.

fixedSpaceSize
Answer the number of bytes in the special heap devoted to objects that the garbage collector cannot move around in memory.

fixedSpaceUsedBytes
Answer the number of bytes that are currently filled in the special heap devoted to objects that the garbage collector cannot move around in memory.

numCompactions
Answer the number of oldspace compactions that happened since the VM was started.

numFixedOOPs
Answer the number of objects that the garbage collector cannot move around in memory.

numFreeOTEs
Answer the number of entries that are currently free in the object table.

numGlobalGCs
Answer the number of global garbage collections (collection of the entire heap) that happened since the VM was started.

numGrowths
Answer the number of times that oldspace was grown since the VM was started.

numOTEs Answer the number of entries that are currently allocated for the object table.

numOldOOPs
Answer the number of objects that reside in the old generation.

numScavenges
Answer the number of scavenges (fast collections of the young generation) that happened since the VM was started.

numWeakOOPs
Answer the number of weak objects that the garbage collector is currently tracking.

oldSpaceSize
Answer the number of bytes in the old generation.

oldSpaceUsedBytes
Answer the number of bytes that are currently filled in the old generation.

reclaimedBytesPerGlobalGC
Answer the average number of bytes that are found to be garbage during a global garbage collections.

reclaimedBytesPerScavenge
Answer the average number of bytes that are found to be garbage during a scavenge.

reclaimedPercentPerScavenge
Answer the average percentage of allocated bytes that are found to be garbage during a scavenge. If this number falls below 60-70 you should definitely increment the size of the eden, because you risk that scavenging is eating a considerable fraction of your execution time; do the measurement on a restarted image, so that the extra tenuring incurred when creating long-lived objects such as classes or methods is not considered.

survSpaceSize
Answer the number of bytes in the 'survivor' area of the young generation (the area to which young objects are relocated during scavenges).

survSpaceUsedBytes
Answer the number of bytes that are currently filled in the 'survivor' area of the young generation.

tenuredBytesPerScavenge
Answer the average number of bytes that are promoted to oldspace during a scavenge.

timeBetweenGlobalGCs
Answer the average number of milliseconds between two global garbage collections.

timeBetweenGrowths
Answer the average number of milliseconds between decisions to grow the heap.

timeBetweenScavenges
Answer the average number of milliseconds between two scavenges (fast collections of the young generation).

timeToCollect
Answer the average number of milliseconds that a global garbage collection takes.

timeToCompact
Answer the average number of milliseconds that compacting the heap takes. This the same time that is taken by growing the heap.

timeToScavenge
Answer the average number of milliseconds that a scavenge takes (fast collections of the young generation).

1.126.6 ObjectMemory: builtins

update Update the values in the object to the current state of the VM.

1.126.7 ObjectMemory: derived information

scavengesBeforeTenuring
Answer the number of scavenges that an object must on average survive before being promoted to oldspace; this is however only an estimate because objects that are reachable from oldspace have a higher probability to be tenured soon,

while objects that are only reachable from thisContext have a lower probability to be tenured. Anyway, if this number falls below 2-3 you should definitely increment the size of eden and/or of survivor space, because you are tenuring too often and relying too much on global garbage collection to keep your heap clean; do the measurement on a restarted image, so that the extra tenuring incurred when creating long-lived objects such as classes or methods is not considered.

1.127 OrderedCollection

Defined in namespace Smalltalk
Superclass: SequenceableCollection
Category: Collections-Sequenceable

My instances represent ordered collections of arbitrary typed objects which are not directly accessible by an index. They can be accessed indirectly through an index, and can be manipulated by adding to the end or based on content (such as add:after:)

1.127.1 OrderedCollection class: instance creation

new Answer an OrderedCollection of default size

new: anInteger
Answer an OrderedCollection of size anInteger

1.127.2 OrderedCollection: accessing

at: anIndex
Answer the anIndex-th item of the receiver

at: anIndex put: anObject
Store anObject at the anIndex-th item of the receiver, answer anObject

first Answer the first item of the receiver

last Answer the last item of the receiver

size Return the number of objects in the receiver

1.127.3 OrderedCollection: adding

add: anObject
Add anObject in the receiver, answer it

add: newObject after: oldObject
Add newObject in the receiver just after oldObject, answer it. Fail if oldObject can't be found

add: newObject afterIndex: i
Add newObject in the receiver just after the i-th, answer it. Fail if i < 0 or i > self size

add: newObject before: oldObject
Add newObject in the receiver just before oldObject, answer it. Fail if oldObject can't be found

add: newObject beforeIndex: i
Add newObject in the receiver just before the i-th, answer it. Fail if i < 1 or i > self size + 1

addAll: aCollection
Add every item of aCollection to the receiver, answer it

addAll: newCollection after: oldObject
Add every item of newCollection to the receiver just after oldObject, answer it. Fail if oldObject is not found

addAll: newCollection afterIndex: i
Add every item of newCollection to the receiver just after the i-th, answer it. Fail if i < 0 or i > self size

addAll: newCollection before: oldObject
Add every item of newCollection to the receiver just before oldObject, answer it. Fail if oldObject is not found

addAll: newCollection beforeIndex: i
Add every item of newCollection to the receiver just before the i-th, answer it. Fail if i < 1 or i > self size + 1

addAllFirst: aCollection
Add every item of newCollection to the receiver right at the start of the receiver. Answer aCollection

addAllLast: aCollection
Add every item of newCollection to the receiver right at the end of the receiver. Answer aCollection

addFirst: newObject
Add newObject to the receiver right at the start of the receiver. Answer newObject

addLast: newObject
Add newObject to the receiver right at the end of the receiver. Answer newObject

1.127.4 OrderedCollection: built ins

primReplaceFrom: start to: stop with: byteArray startingAt: replaceStart
Replace the characters from start to stop with new characters whose ASCII codes are contained in byteArray, starting at the replaceStart location of byteArray

1.127.5 OrderedCollection: enumerating

do: aBlock
Evaluate aBlock for all the elements in the collection

1.127.6 OrderedCollection: removing

identityRemove: oldObject
Remove oldObject from the receiver. If absent, fail, else answer oldObject.

identityRemove: anObject ifAbsent: aBlock
Remove anObject from the receiver. If it can't be found, answer the result of evaluating aBlock

remove: anObject ifAbsent: aBlock
Remove anObject from the receiver. If it can't be found, answer the result of evaluating aBlock

removeAtIndex: anIndex
Remove the object at index anIndex from the receiver. Fail if the index is out of bounds.

removeFirst
Remove an object from the start of the receiver. Fail if the receiver is empty

removeLast
Remove an object from the end of the receiver. Fail if the receiver is empty

1.128 Package

Defined in namespace Smalltalk
Superclass: Kernel.PackageInfo
Category: Language-Packaging
I am not part of a standard Smalltalk system. I store internally the information on a Smalltalk package, and can output my description in XML.

1.128.1 Package class: accessing

tags Not commented.

1.128.2 Package class: instance creation

parse: file Answer a package from the XML description in file.

1.128.3 Package: accessing

addBuiltFile: aString
Not commented.

addCallout: aString
Not commented.

addFeature: aString
Not commented.

addFile: aString
Not commented.

addFileIn: aString
Not commented.

addLibrary: aString
Not commented.

addModule: aString
Not commented.

addPrerequisite: aString
Not commented.

addSunitScript: aString
Not commented.

baseDirectories
Answer 'baseDirectories'.

baseDirectories: aCollection
Check if it's possible to resolve the names in the package according to the base directories in baseDirectories, which depend on where the packages.xml is found: the three possible places are 1) the system kernel directory's parent directory, 2) the local kernel directory's parent directory, 3) the local image directory (in order of decreasing priority).

For a packages.xml found in the system kernel directory's parent directory, all three directories are searched. For a packages.xml found in the local kernel directory's parent directory, only directories 2 and 3 are searched. For a packages.xml directory in the local image directory, instead, only directory 3 is searched.

builtFiles
Answer a (modifiable) OrderedCollection of files that are part of the package but are not distributed.

callouts
Answer a (modifiable) Set of call-outs that are required to load the package. Their presence is checked after the libraries and modules are loaded so that you can do a kind of versioning.

directory
Answer the base directory from which to load the package.

features
Answer a (modifiable) Set of features provided by the package.

fileIns
Answer a (modifiable) OrderedCollections of files that are to be filed-in to load the package. This is usually a subset of 'files' and 'builtFiles'.

files
Answer a (modifiable) OrderedCollection of files that are part of the package.

fullPathOf: fileName
Try appending 'self directory' and fileName to each of the directory in baseDirectories, and return the path to the first tried filename that exists. Raise a PackageNotAvailable exception if no directory is found that contains the file.

libraries
Answer a (modifiable) Set of shared library names that are required to load the package.

modules
Answer a (modifiable) Set of modules that are required to load the package.

namespace
Answer the namespace in which the package is loaded.

namespace: aString
Set to aString the namespace in which the package is loaded.

prerequisites
Answer a (modifiable) Set of prerequisites.

primFileIn
Private - File in the given package without paying attention at dependencies and C callout availability

relativeDirectory
Answer the directory, relative to the packages file, from which to load the package.

relativeDirectory: dir
Set the directory, relative to the packages file, from which to load the package, to dir.

startScript
Answer the start script for the package.

startScript: aString
Set the start script for the package to aString.

stopScript Answer the start script for the package.

stopScript: aString
Set the stop script for the package to aString.

sunitScripts
Answer a (modifiable) OrderedCollection of SUnit scripts that compose the package's test suite.

test Answer the test sub-package.

test: aPackage
Set the test sub-package to be aPackage.

url Answer the URL at which the package repository can be found.

url: aString
Set to aString the URL at which the package repository can be found.

version Not commented.

version: aVersion
Not commented.

1.128.4 Package: still unclassified

checkTagIfInPath: aString
Not commented.

dir: file tag: aDictionary
Not commented.

isInPath Not commented.

parseAttributes: aString
Not commented.

path Not commented.

path: aString
Not commented.

1.128.5 Package: version parsing

parseVersion: aString
Not commented.

1.129 PackageLoader

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Packaging
I am not part of a standard Smalltalk system. I provide methods for retrieving package information from an XML file and to load packages into a Smalltalk image, correctly handling dependencies.

1.129.1 PackageLoader class: accessing

builtFilesFor: package
Answer a Set of Strings containing the filenames of the given package's machine-generated files (relative to the directory answered by #directoryFor:)

calloutsFor: package
Answer a Set of Strings containing the filenames of the given package's required callouts (relative to the directory answered by #directoryFor:)

directoryFor: package
Answer a Directory object to the given package's files

featuresFor: package
Answer a Set of Strings containing the features provided by the given package.

fileInsFor: package
Answer a Set of Strings containing the filenames of the given package's file-ins (relative to the directory answered by #directoryFor:)

filesFor: package
Answer a Set of Strings containing the filenames of the given package's files (relative to the directory answered by #directoryFor:)

flush Set to reload the 'packages.xml' file the next time it is needed.

ignoreCallouts
Answer whether unavailable C callouts must generate errors or not.

ignoreCallouts: aBoolean
Set whether unavailable C callouts must generate errors or not.

librariesFor: package
Answer a Set of Strings containing the filenames of the given package's libraries (relative to the directory answered by #directoryFor:)

modulesFor: package
Answer a Set of Strings containing the filenames of the given package's modules (relative to the directory answered by #directoryFor:)

packageAt: package
Answer a Package object for the given package

packageAt: package ifAbsent: aBlock
Answer a Package object for the given package

prerequisitesFor: package
Answer a Set of Strings containing the prerequisites for the given package

refresh
Reload the 'packages.xml' file in the image and kernel directories. The three possible places are 1) the kernel directory's parent directory, 2) the '.st' subdirectory of the user's home directory, 3) the local image directory (in order of decreasing priority).

For a packages.xml found in the kernel directory's parent directory, all three directories are searched. For a packages.xml found in the '.st' subdirectory, only directories 2 and 3 are searched. For a packages.xml directory in the local image directory, finally, only directory 3 is searched.

sunitScriptFor: package
Answer a Strings containing a SUnit script that describes the package's test suite.

1.129.2 PackageLoader class: loading

fileInPackage: package
File in the given package into GNU Smalltalk.

fileInPackages: packagesList
File in all the packages in packagesList into GNU Smalltalk.

1.129.3 PackageLoader class: testing

canLoad: package
Answer whether all the needed pre-requisites for package are available.

1.130 Permission

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Security
I am the basic class that represents whether operations that could harm the system's security are allowed or denied.

1.130.1 Permission class: testing

allowing: aSymbol target: aTarget action: action
Not commented.

allowing: aSymbol target: aTarget actions: actionsArray
Not commented.

denying: aSymbol target: aTarget action: action
Not commented.

denying: aSymbol target: aTarget actions: actionsArray
Not commented.

granting: aSymbol target: aTarget action: action
Not commented.

granting: aSymbol target: aTarget actions: actionsArray
Not commented.

name: aSymbol target: aTarget action: action
Not commented.

name: aSymbol target: aTarget actions: actionsArray
Not commented.

1.130.2 Permission: accessing

action: anObject
Not commented.

actions Answer 'actions'.

actions: anObject
Not commented.

allow Not commented.

allowing Not commented.

deny Not commented.

denying Not commented.

isAllowing Answer 'positive'.

name Answer 'name'.

name: anObject
Not commented.

target Answer 'target'.

target: anObject
Not commented.

1.130.3 Permission: testing

check: aPermission for: anObject
Not commented.

implies: aPermission
Not commented.

1.131 PluggableAdaptor

Defined in namespace Smalltalk
Superclass: ValueAdaptor
Category: Language-Data types

I mediate between complex get/set behavior and the #value/#value: protocol used by ValueAdaptors. The get/set behavior can be implemented by two blocks, or can be delegated to another object with messages such as #someProperty to get and #someProperty: to set.

1.131.1 PluggableAdaptor class: creating instances

getBlock: getBlock putBlock: putBlock
Answer a PluggableAdaptor using the given blocks to implement #value and #value:

on: anObject aspect: aSymbol
Answer a PluggableAdaptor using anObject's aSymbol message to implement #value, and anObject's aSymbol: message (aSymbol followed by a colon) to implement #value:

on: anObject getSelector: getSelector putSelector: putSelector
Answer a PluggableAdaptor using anObject's getSelector message to implement #value, and anObject's putSelector message to implement #value:

on: anObject index: anIndex
Answer a PluggableAdaptor using anObject's #at: and #at:put: message to implement #value and #value:; the first parameter of #at: and #at:put: is anIndex

on: aDictionary key: aKey
Same as #on:index:. Provided for clarity and completeness.

1.131.2 PluggableAdaptor: accessing

value Get the value of the receiver.

value: anObject
Set the value of the receiver.

1.132 PluggableProxy

Defined in namespace Smalltalk
Superclass: AlternativeObjectProxy
Category: Streams-Files

I am a proxy that stores a different object and, upon load, sends #reconstructOriginalObject to that object (which can be a DirectedMessage, in which case the message is sent). The object to be stored is retrieved by sending #binaryRepresentationObject to the object.

1.132.1 PluggableProxy class: accessing

on: anObject
Answer a proxy to be used to save anObject. The proxy stores a different object obtained by sending to anObject the #binaryRepresentationObject message (embedded between #preStore and #postStore as usual).

1.132.2 PluggableProxy: saving and restoring

object Reconstruct the object stored in the proxy and answer it; the binaryRepresentationObject is sent the #reconstructOriginalObject message, and the resulting object is sent the #postLoad message.

1.133 Point

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Data types
Beginning of a Point class for simple display manipulation. Has not been exhaustively tested but appears to work for the basic primitives and for the needs of the Rectangle class.

1.133.1 Point class: instance creation

new Create a new point with both coordinates set to 0

x: xInteger y: yInteger
Create a new point with the given coordinates

1.133.2 Point: accessing

x Answer the x coordinate

x: aNumber
Set the x coordinate to aNumber

x: anXNumber y: aYNumber
Set the x and y coordinate to anXNumber and aYNumber, respectively

y Answer the y coordinate

y: aNumber
Set the y coordinate to aNumber

1.133.3 Point: arithmetic

*** scale** Multiply the receiver by scale, which can be a Number or a Point

+ delta Sum the receiver and delta, which can be a Number or a Point

- delta Subtract delta, which can be a Number or a Point, from the receiver

/ scale Divide the receiver by scale, which can be a Number or a Point, with no loss of precision

// scale Divide the receiver by scale, which can be a Number or a Point, with truncation towards -infinity

abs Answer a new point whose coordinates are the absolute values of the receiver's

1.133.4 Point: comparing

< aPoint Answer whether the receiver is higher and to the left of aPoint

<= aPoint Answer whether aPoint is equal to the receiver, or the receiver is higher and to the left of aPoint

= aPoint Answer whether the receiver is equal to aPoint

> aPoint Answer whether the receiver is lower and to the right of aPoint

>= aPoint Answer whether aPoint is equal to the receiver, or the receiver is lower and to the right of aPoint

max: aPoint
Answer self if it is lower and to the right of aPoint, aPoint otherwise

min: aPoint
Answer self if it is higher and to the left of aPoint, aPoint otherwise

1.133.5 Point: converting

asPoint Answer the receiver.

asRectangle
Answer an empty rectangle whose origin is self

corner: aPoint
Answer a Rectangle whose origin is the receiver and whose corner is aPoint

extent: aPoint
Answer a Rectangle whose origin is the receiver and whose extent is aPoint

hash Answer an hash value for the receiver

1.133.6 Point: point functions

arcTan Answer the angle (measured counterclockwise) between the receiver and a ray starting in (0, 0) and moving towards (1, 0) - i.e. 3 o'clock

dist: aPoint
Answer the distance between the receiver and aPoint

dotProduct: aPoint
Answer the dot product between the receiver and aPoint

grid: aPoint
Answer a new point whose coordinates are rounded towards the nearest multiple of aPoint

normal Rotate the Point 90degrees clockwise and get the unit vector

transpose Answer a new point whose coordinates are the receiver's coordinates exchanged (x becomes y, y becomes x)

truncatedGrid: aPoint
Answer a new point whose coordinates are rounded towards -infinity, to a multiple of grid (which must be a Point)

1.133.7 Point: printing

printOn: aStream
Print a representation for the receiver on aStream

1.133.8 Point: storing

storeOn: aStream
Print Smalltalk code compiling to the receiver on aStream

1.133.9 Point: truncation and round off

rounded Answer a new point whose coordinates are rounded to the nearest integer

truncateTo: grid
Answer a new point whose coordinates are rounded towards -infinity, to a multiple of grid (which must be a Number)

1.134 PositionableStream

Defined in namespace Smalltalk
Superclass: Stream
Category: Streams-Collections
My instances represent streams where explicit positioning is permitted. Thus, my streams act in a manner to normal disk files: you can read or write sequentially, but also position the file to a particular place whenever you choose. Generally, you'll want to use ReadStream, WriteStream or ReadWriteStream instead of me to create and use streams.

1.134.1 PositionableStream class: instance creation

on: aCollection
Answer an instance of the receiver streaming on the whole contents of aCollection

on: aCollection from: firstIndex to: lastIndex
Answer an instance of the receiver streaming from the firstIndex-th item of aCollection to the lastIndex-th

1.134.2 PositionableStream: accessing-reading

close Disassociate a stream from its backing store.

contents Returns a collection of the same type that the stream accesses, up to and including the final element.

copyFrom: start to: end
Answer the data on which the receiver is streaming, from the start-th item to the end-th. Note that this method is 0-based, unlike the one in Collection, because a Stream's #position method returns 0-based values.

next Answer the next item of the receiver. Returns nil when at end of stream.

nextAvailable: anInteger into: aCollection startingAt: pos
Place up to anInteger objects from the receiver into aCollection, starting from position pos in the collection and stopping if no more data is available.

nextAvailable: anInteger putAllOn: aStream
Copy up to anInteger objects from the receiver into aStream, stopping if no more data is available.

peek Returns the next element of the stream without moving the pointer. Returns nil when at end of stream.

peekFor: anObject
Returns true and gobbles the next element from the stream of it is equal to anObject, returns false and doesn't gobble the next element if the next element is not equal to anObject.

readStream
Answer a ReadStream on the same contents as the receiver

reverseContents
Returns a collection of the same type that the stream accesses, up to and including the final element, but in reverse order.

upTo: anObject
Returns a collection of the same type that the stream accesses, up to but not including the object anObject. Returns the entire rest of the stream's contents if anObject is not present.

upToEnd Returns a collection of the same type that the stream accesses, containing the entire rest of the stream's contents.

1.134.3 PositionableStream: class type methods

isExternalStream
We stream on a collection residing in the image, so answer false

species Return the type of the collections returned by #upTo: etc., which are the same kind as those returned by the collection with methods such as #select:.

1.134.4 PositionableStream: compiling

name Answer a string that represents what the receiver is streaming on

segmentFrom: startPos to: endPos
Answer an object that, when sent #asString, will yield the result of sending 'copyFrom: startPos to: endPos' to the receiver

1.134.5 PositionableStream: positioning

basicPosition: anInteger
Move the stream pointer to the anInteger-th object

isPositionable
Answer true if the stream supports moving backwards with #skip:.

position Answer the current value of the stream pointer

position: anInteger
Move the stream pointer to the anInteger-th object

reset Move the stream back to its first element. For write-only streams, the stream is truncated there.

setToEnd Move the current position to the end of the stream.

size Answer the size of data on which we are streaming.

skip: anInteger
Move the current position by anInteger places, either forwards or backwards.

1.134.6 PositionableStream: still unclassified

nextPutAllOn: aStream
Write all the objects in the receiver to aStream.

1.134.7 PositionableStream: testing

atEnd Answer whether the objects in the stream have reached an end

basicAtEnd
Answer whether the objects in the stream have reached an end. This method must NOT be overridden.

isEmpty Answer whether the stream has no objects

1.134.8 PositionableStream: truncating

truncate Truncate the receiver to the current position - only valid for writing streams

1.135 Process

Defined in namespace Smalltalk
Superclass: Link
Category: Language-Processes
I represent a unit of computation. My instances are independantly executable blocks that have a priority associated with them, and they can suspend themselves and resume themselves however they wish.

1.135.1 Process: accessing

externalInterruptsEnabled
Answer whether the receiver is executed with interrupts enabled

name Answer the user-friendly name of the process.

name: aString
Give the name aString to the process

priority Answer the receiver's priority

priority: anInteger
Change the receiver's priority to anInteger

queueInterrupt: aBlock
Force the receiver to be interrupted and to evaluate aBlock as soon as it becomes the active process (this could mean NOW if the receiver is active). If the process is temporarily suspended or waiting on a semaphore, it is temporarily woken up so that the interrupt is processed as soon as the process priority allows to do. Answer the receiver.

suspendedContext
Answer the context that the process was executing at the time it was suspended.

suspendedContext: aContext
Modify the context that the process was executing at the time it was suspended.

valueWithoutInterrupts: aBlock
Evaluate aBlock and delay all interrupts that are requested during its execution to after aBlock returns.

1.135.2 Process: basic

context Return the execution context of the receiver.

debugger Return the object in charge of debugging the receiver. This always returns nil unless the DebugTools package is loaded.

finalize Terminate processes that are GCed while waiting on a dead semaphore.

lowerPriority
Lower a bit the priority of the receiver. A #lowerPriority will cancel a previous #raisePriority, and vice versa.

makeUntrusted: aBoolean
Set whether the receiver is trusted or not.

primTerminate
Terminate the receiver - This is nothing more than prohibiting to resume the process, then suspending it.

raisePriority
Raise a bit the priority of the receiver. A #lowerPriority will cancel a previous #raisePriority, and vice versa.

singleStep Execute a limited amount of code (usually a bytecode, or up to the next backward jump, or up to the next message send) of the receiver, which must in a ready-to-run state (neither executing nor terminating nor suspended), then restart running the current process. The current process should have higher priority than the receiver. For better performance, use the underlying primitive, Process>>#singleStepWaitingOn:.

terminate Terminate the receiver after having evaluated all the #ensure: and #ifCurtailed: blocks that are active in it. This is done by signalling a ProcessBeingTerminated notification.

terminateOnQuit
Mark the receiver so that it is terminated when ObjectMemory class>>#quit: is sent.

1.135.3 Process: builtins

resume Resume the receiver's execution

singleStepWaitingOn: aSemaphore
Execute a limited amount of code (usually a bytecode, or up to the next backward jump, or up to the next message send) of the receiver, which must in a ready-to-run state (neither executing nor terminating nor suspended), then restart running the current process. aSemaphore is used as a means to synchronize the execution of the current process and the receiver and should have no signals on it. The current process should have higher priority than the receiver.

suspend Do nothing if we're already suspended. Note that the blue book made suspend a primitive - but the real primitive is yielding control to another process. Suspending is nothing more than taking ourselves out of every scheduling list and THEN yielding control to another process

yield Yield control from the receiver to other processes

1.135.4 Process: debugging

detach Do nothing, instances of Process are already detached.

1.135.5 Process: printing

printOn: aStream
Print a representation of the receiver on aStream

1.136 ProcessEnvironment

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Processes
I represent a proxy for thread-local variables defined for Smalltalk processes. Associations requested to me retrieve the thread-local value for the current process. For now, I don't provide the full protocol of a Dictionary; in particular the iteration protocol is absent.

1.136.1 ProcessEnvironment class: disabled

new This method should not be called for instances of this class.

1.136.2 ProcessEnvironment class: singleton

uniqueInstance
Return the singleton instance of ProcessEnvironment.

1.136.3 ProcessEnvironment: accessing

add: newObject
Add the newObject association to the receiver

associationAt: key
Answer the value associated to the given key, or the result of evaluating aBlock if the key is not found

associationAt: key ifAbsent: aBlock
Answer the value associated to the given key, or the result of evaluating aBlock if the key is not found

at: key Answer the value associated to the given key. Return nil if the key is not found

at: key ifAbsent: aBlock
Answer the value associated to the given key, or the result of evaluating aBlock if the key is not found

at: key ifAbsentPut: aBlock
Answer the value associated to the given key, setting it to the result of evaluating aBlock if the key is not found.

at: key ifPresent: aBlock
Answer the value associated to the given key, or the result of evaluating aBlock if the key is not found

at: key put: value
Store value as associated to the given key

keys Answer a kind of Set containing the keys of the receiver

1.136.4 ProcessEnvironment: dictionary removing

remove: anAssociation
Remove anAssociation's key from the dictionary

remove: anAssociation ifAbsent: aBlock
Remove anAssociation's key from the dictionary

removeAllKeys: keys
Remove all the keys in keys, without raising any errors

removeAllKeys: keys ifAbsent: aBlock
Remove all the keys in keys, passing the missing keys as parameters to aBlock as they're encountered

removeKey: aSymbol
Remove the aSymbol key from the dictionary

removeKey: aSymbol ifAbsent: aBlock
Remove the aSymbol key from the dictionary

1.136.5 ProcessEnvironment: dictionary testing

includesKey: key
Answer whether the receiver contains the given key

1.137 ProcessorScheduler

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Processes

I provide methods that control the execution of processes.

1.137.1 ProcessorScheduler class: instance creation

new Error—new instances of ProcessorScheduler should not be created.

1.137.2 ProcessorScheduler: basic

activeDebugger
Answer the active process' debugger

activePriority
Answer the active process' priority

activeProcess
Answer the active process

processEnvironment
Answer another singleton object hosting thread-local variables for the Smalltalk processes. This acts like a normal Dictionary with a couple of differences: a) using #associationAt: will return special associations that retrieve a thread-local value; b) requesting missing keys will return nil, and removing them will be a nop.

processesAt: aPriority
Answer a linked list of processes at the given priority

terminateActive
Terminate the active process

timeSlice Answer the timeslice that is assigned to each Process before it is automatically preempted by the system (in milliseconds). An answer of zero means that preemptive multitasking is disabled. Note that the system by default is compiled without preemptive multitasking, and that even if it is enabled it will work only under BSD derivatives (or, in general, systems that support ITIMER_VIRTUAL).

timeSlice: milliSeconds
Set the timeslice that is assigned to each Process before it is automatically preempted by the system. Setting this to zero disables preemptive multitasking. Note that the system by default is compiled with preemptive multitasking disabled, and that even if it is enabled it will surely work only under BSD derivatives (or, in general, systems that support ITIMER_VIRTUAL).

yield Let the active process yield control to other processes

1.137.3 ProcessorScheduler: built ins

disableInterrupts
Disable interrupts caused by external events while the current process is executing. Note that interrupts are disabled on a per-process basis, and that calling #disableInterrupts twice requires calling #enableInterrupts twice as well to re-enable interrupts.

enableInterrupts
Re-enable interrupts caused by external events while the current process is executing. By default, interrupts are enabled.

1.137.4 ProcessorScheduler: idle tasks

idle Private - Call the next idle task. Return whether GNU Smalltalk should pause until the next OS signal.

idleAdd: aBlock
Register aBlock to be executed when things are idle

initialize Private - Start the finalization process.

pause: aBoolean
Private - Pause for some time if aBoolean is false, or until a signal if it is true.

startFinalizers
Private - Fire a low-priority process to finalize the objects

update: aSymbol
If we left some work behind when the image was saved, do it now.

1.137.5 ProcessorScheduler: printing

printOn: aStream
Store onto aStream a printed representation of the receiver

1.137.6 ProcessorScheduler: priorities

highIOPriority
Answer the priority for system high-priority I/O processes, such as a process handling input from a network.

highestPriority
Answer the highest valid priority

idlePriority
Answer the priority of idle processes.

lowIOPriority
Answer the priority for system low-priority I/O processes. Examples are the process handling input from the user (keyboard, pointing device, etc.) and the process distributing input from a network.

lowestPriority
Answer the lowest valid priority

priorityName: priority

Private - Answer a name for the given process priority

systemBackgroundPriority

Answer the priority for system background-priority processes. An incremental garbage collector could run at this level but now it runs at idlePriority instead.

timingPriority

Answer the priority for system real-time processes.

unpreemptedPriority

Answer the highest priority avilable in the system; never create a process with this priority, instead use BlockClosure>>#valueWithoutPreemption.

userBackgroundPriority

Answer the priority for user background-priority processes

userInterruptPriority

Answer the priority for user interrupt-priority processes. Processes run at this level will preempt the window scheduler and should, therefore, not consume the processor forever.

userSchedulingPriority

Answer the priority for user standard-priority processes

1.137.7 ProcessorScheduler: storing

storeOn: aStream

Store onto aStream a Smalltalk expression which evaluates to the receiver

1.137.8 ProcessorScheduler: timed invocation

isTimeoutProgrammed

Private - Answer whether there is a pending call to #signal:atMilliseconds:

signal: aSemaphore atNanosecondClockValue: ns

Private - signal 'aSemaphore' when the nanosecond clock reaches 'ns' nanoseconds.

signal: aSemaphore onInterrupt: anIntegerSignalNumber

Signal 'aSemaphore' when the given C signal occurs.

1.138 ProcessVariable

Defined in namespace Smalltalk
Superclass: LookupKey
Category: Language-Processes

I represent a proxy for a thread-local variable defined for a process. Requesting the value will return the thread-local setting for the current process.

1.138.1 ProcessVariable class: accessing

key: anObject

Return a new ProcessVariable with the given key. Not that the key need not be a symbol or string, for example you could use an array #(#{class name}

'name'). Setting the variable's value will automatically create it in the current process, while removal must be done by hand through the ProcessEnvironment singleton object.

new Return a new ProcessVariable with a new anonymous but unique key. It is suggested to use a descriptive name instead to ease debugging. Setting the variable's value will automatically create it in the current process, while removal must be done by hand through the ProcessEnvironment singleton object.

1.138.2 ProcessVariable: accessing

environment
Return the environment in which this ProcessVariable lives. This is the singleton instance of ProcessEnvironment for all variables.

use: anObject during: aBlock
Set the value of this variable to anObject during the execution of aBlock, then restore it.

value Return the value of this variable in the current process.

value: anObject
Set the value of the current process's copy of the variable to be anObject.

valueIfAbsent: aBlock
Return the value of this variable in the current process.

1.139 Promise

Defined in namespace Smalltalk
Superclass: ValueHolder
Category: Language-Data types
I store my value in a variable, and know whether I have been initialized or not. If you ask for my value and I have not been initialized, I suspend the process until a value has been assigned.

1.139.1 Promise class: creating instances

for: aBlock
Invoke aBlock at an indeterminate time in an indeterminate process before answering its value from #value sent to my result.

null This method should not be called for instances of this class.

1.139.2 Promise: accessing

hasError Answer whether calling #value will raise an exception.

hasValue Answer whether we already have a value (or calling #value will raise an error).

value Get the value of the receiver.

value: anObject
Set the value of the receiver.

1.139.3 Promise: initializing

initialize Private - set the initial state of the receiver

1.139.4 Promise: printing

printOn: aStream
Print a representation of the receiver

1.139.5 Promise: still unclassified

errorValue: anException
Private - Raise anException whenever #value is called.

1.140 Random

Defined in namespace Smalltalk
Superclass: Stream
Category: Streams
My instances are generator streams that produce random numbers, which are floating point values between 0 and 1.

1.140.1 Random class: instance creation

new Create a new random number generator whose seed is given by the current time on the millisecond clock

seed: aFloat
Create a new random number generator whose seed is aFloat

1.140.2 Random class: shortcuts

between: low and: high
Return a random integer between the given extrema

next Return a random number between 0 and 1 (excluded)

source Return a standard source of random numbers.

1.140.3 Random: basic

atEnd This stream never ends. Always answer false.

between: low and: high
Return a random integer between low and high.

next Return the next random number in the sequence.

nextPut: value
This method should not be called for instances of this class.

1.140.4 Random: testing

chiSquare Compute the chi-square of the random that this class generates.

chiSquare: n range: r
Return the chi-square deduced from calculating n random numbers in the 0..r range.

1.141 ReadStream

Defined in namespace Smalltalk
Superclass: PositionableStream
Category: Streams-Collections
I implement the set of read-only stream objects. You may read from my objects, but you may not write to them.

1.141.1 ReadStream class: instance creation

on: aCollection
Answer a new stream working on aCollection from its start.

on: aCollection from: firstIndex to: lastIndex
Answer an instance of the receiver streaming from the firstIndex-th item of aCollection to the lastIndex-th

1.142 ReadWriteStream

Defined in namespace Smalltalk
Superclass: WriteStream
Category: Streams-Collections
I am the class of streams that may be read and written from simultaneously. In some sense, I am the best of both ReadStream and WriteStream.

1.142.1 ReadWriteStream class: instance creation

on: aCollection
Answer a new stream working on aCollection from its start. The stream starts at the front of aCollection.

on: aCollection from: firstIndex to: lastIndex
Answer an instance of the receiver streaming from the firstIndex-th item of aCollection to the lastIndex-th

with: aCollection
Answer a new instance of the receiver which streams from the end of aCollection.

1.142.2 ReadWriteStream: positioning

contents Unlike WriteStreams, ReadWriteStreams return the whole contents of the underlying collection.

1.143 Rectangle

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Data types

Beginning of the Rectangle class for simple display manipulation. Rectangles require the Point class to be available. An extension to the Point class is made here that since it requires Rectangles to be defined (see converting)

1.143.1 Rectangle class: instance creation

left: leftNumber right: rightNumber top: topNumber bottom: bottomNumber
Answer a rectangle with the given coordinates

left: leftNumber top: topNumber right: rightNumber bottom: bottomNumber
Answer a rectangle with the given coordinates

new Answer the (0 @ 0 corner: 0 @ 0) rectangle

origin: originPoint corner: cornerPoint
Answer a rectangle with the given corners

origin: originPoint extent: extentPoint
Answer a rectangle with the given origin and size

1.143.2 Rectangle: accessing

bottom Answer the corner's y of the receiver

bottom: aNumber
Set the corner's y of the receiver

bottomCenter
Answer the center of the receiver's bottom side

bottomLeft
Answer the bottom-left corner of the receiver

bottomLeft: aPoint
Answer the receiver with the bottom-left changed to aPoint

bottomRight
Answer the bottom-right corner of the receiver

bottomRight: aPoint
Change the bottom-right corner of the receiver

center Answer the center of the receiver

corner Answer the corner of the receiver

corner: aPoint
Set the corner of the receiver

extent Answer the extent of the receiver

extent: aPoint
Change the size of the receiver, keeping the origin the same

height Answer the height of the receiver

height: aNumber
Set the height of the receiver

left Answer the x of the left edge of the receiver

left: aValue
Set the x of the left edge of the receiver

left: l top: t right: r bottom: b
Change all four the coordinates of the receiver's corners

leftCenter Answer the center of the receiver's left side

origin Answer the top-left corner of the receiver

origin: aPoint
Change the top-left corner of the receiver to aPoint

origin: pnt1 corner: pnt2
Change both the origin (top-left corner) and the corner (bottom-right corner) of the receiver

origin: pnt1 extent: pnt2
Change the top-left corner and the size of the receiver

right Answer the x of the bottom-right corner of the receiver

right: aNumber
Change the x of the bottom-right corner of the receiver

rightCenter
Answer the center of the receiver's right side

top Answer the y of the receiver's top-left corner

top: aValue
Change the y of the receiver's top-left corner

topCenter Answer the center of the receiver's top side

topLeft Answer the receiver's top-left corner

topLeft: aPoint
Change the receiver's top-left corner's coordinates to aPoint

topRight Answer the receiver's top-right corner

topRight: aPoint
Change the receiver's top-right corner to aPoint

width Answer the receiver's width

width: aNumber
Change the receiver's width to aNumber

1.143.3 Rectangle: copying

copy Return a deep copy of the receiver for safety.

1.143.4 Rectangle: printing

printOn: aStream
Print a representation of the receiver on aStream

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.143.5 Rectangle: rectangle functions

amountToTranslateWithin: aRectangle
Answer a Point so that if aRectangle is translated by that point, its origin lies within the receiver's.

area Answer the receiver's area. The area is the width times the height, so it is possible for it to be negative if the rectangle is not normalized.

areasOutside: aRectangle
Answer a collection of rectangles containing the parts of the receiver outside of aRectangle. For all points in the receiver, but outside aRectangle, exactly one rectangle in the collection will contain that point.

expandBy: delta
Answer a new rectangle that is the receiver expanded by aValue: if aValue is a rectangle, calculate origin=origin-aValue origin, corner=corner+aValue corner; else calculate origin=origin-aValue, corner=corner+aValue.

insetBy: delta
Answer a new rectangle that is the receiver inset by aValue: if aValue is a rectangle, calculate origin=origin+aValue origin, corner=corner-aValue corner; else calculate origin=origin+aValue, corner=corner-aValue.

insetOriginBy: originDelta corner: cornerDelta
Answer a new rectangle that is the receiver inset so that origin=origin+originDelta, corner=corner-cornerDelta. The deltas can be points or numbers

intersect: aRectangle
Answers the rectangle (if any) created by the overlap of rectangles A and B. Answers nil if the rectangles do not overlap

merge: aRectangle
Answer a new rectangle which is the smallest rectangle containing both the receiver and aRectangle.

translatedToBeWithin: aRectangle
Answer a copy of the receiver that does not extend beyond aRectangle.

1.143.6 Rectangle: testing

= aRectangle
Answer whether the receiver is equal to aRectangle

contains: aRectangle
: Answer true if the receiver contains (see containsPoint:) both aRectangle's origin and aRectangle's corner

containsPoint: aPoint
: Answer true if aPoint is equal to, or below and to the right of, the receiver's origin; and aPoint is above and to the left of the receiver's corner

hash Answer an hash value for the receiver

intersects: aRectangle
: Answer true if the receiver intersect aRectangle, i.e. if it contains (see containsPoint:) any of aRectangle corners or if aRectangle contains the receiver

1.143.7 Rectangle: transforming

moveBy: aPoint
: Change the receiver so that the origin and corner are shifted by aPoint

moveTo: aPoint
: Change the receiver so that the origin moves to aPoint and the size remains unchanged

scaleBy: scale
: Answer a copy of the receiver in which the origin and corner are multiplied by scale

translateBy: factor
: Answer a copy of the receiver in which the origin and corner are shifted by aPoint

1.143.8 Rectangle: truncation and round off

rounded Answer a copy of the receiver with the coordinates rounded to the nearest integers

1.144 RecursionLock

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Processes

1.144.1 RecursionLock class: instance creation

new Answer a new semaphore

1.144.2 RecursionLock: accessing

isOwnerProcess
: Answer whether the receiver is the owner of the lock.

name Answer a user-defined name for the lock.

name: aString
Set to aString the user-defined name for the lock.

waitingProcesses
Answer the set of processes that are waiting on the semaphore.

wouldBlock
Answer whether sending #wait to the receiver would suspend the active process.

1.144.3 RecursionLock: mutual exclusion

critical: aBlock
Wait for the receiver to be free, execute aBlock and signal the receiver again. Return the result of evaluating aBlock.

1.144.4 RecursionLock: printing

printOn: aStream
Print a human-readable represention of the receiver on aStream.

1.145 Regex

Defined in namespace Smalltalk
Superclass: Object
Category: Collections-Text

A Regex is a read-only string for which the regular expression matcher can cache a compiled representation, thus speeding up matching. Regex objects are constructed automatically by methods that expect to match many times the same regular expression, but can also be constructed explicitly sending #asRegex to a String or Symbol.

Creation of Regex objects inside a loop is of course slower than creating them outside the loop, but special care is taken so that the same Regex object is used whenever possible (when converting Strings to Regex, the cache is sought for an equivalent, already constructed Regex).

1.145.1 Regex class: instance creation

fromString: aString
Like 'aString asRegex'.

new Do not send this message.

1.145.2 Regex: basic

at: anIndex put: anObject
Fail. Regex objects are read-only.

copy Answer the receiver; instances of Regex are identity objects because their only purpose is to ease caching, and we obtain better caching if we avoid copying Regex objects

1.145.3 Regex: conversion

asRegex Answer the receiver, which *is* a Regex!

asString Answer the receiver, converted back to a String

species Answer 'String'.

1.145.4 Regex: printing

displayOn: aStream
Print a represention of the receiver on aStream. For most objects this is simply its #printOn: representation, but for strings and characters, superfluous dollars or extra pairs of quotes are stripped.

displayString
Answer a String representing the receiver. For most objects this is simply its #printString, but for strings and characters, superfluous dollars or extra pair of quotes are stripped.

printOn: aStream
Print a represention of the receiver on aStream.

1.146 RegexResults

Defined in namespace Smalltalk
Superclass: Object
Category: Collections-Text
I hold the results of a regular expression match, and I can reconstruct which parts of the matched string were assigned to each subexpression. Methods such as #=~ return RegexResults objects, while others transform the string directly without passing the results object back to the caller.

1.146.1 RegexResults: accessing

asArray If the regular expression was matched, return an Array with the subexpressions that were present in the regular expression.

at: anIndex
If the regular expression was matched, return the text of the anIndex-th subexpression in the successful match.

from If the regular expression was matched, return the index of the first character in the successful match.

fromAt: anIndex
If the regular expression was matched, return the index of the first character of the anIndex-th subexpression in the successful match.

intervalAt: anIndex
If the regular expression was matched, return an Interval for the range of indices in the anIndex-th subexpression of the successful match.

match If the regular expression was matched, return the text of the successful match.

matchInterval
If the regular expression was matched, return an Interval for the range of indices of the successful match.

size If the regular expression was matched, return the number of subexpressions that were present in the regular expression.

subject If the regular expression was matched, return the text that was matched against it.

to If the regular expression was matched, return the index of the last character in the successful match.

toAt: anIndex
If the regular expression was matched, return the index of the last character of the anIndex-th subexpression in the successful match.

1.146.2 RegexResults: testing

ifMatched: oneArgBlock
If the regular expression was matched, pass the receiver to oneArgBlock and return its result. Otherwise, return nil.

ifMatched: oneArgBlock ifNotMatched: zeroArgBlock
If the regular expression was matched, evaluate oneArgBlock with the receiver as the argument. If it was not, evaluate zeroArgBlock. Answer the result of the block's evaluation.

ifNotMatched: zeroArgBlock
If the regular expression was matched, return the receiver. If it was not, evaluate zeroArgBlock and return its result.

ifNotMatched: zeroArgBlock ifMatched: oneArgBlock
If the regular expression was matched, evaluate oneArgBlock with the receiver as the argument. If it was not, evaluate zeroArgBlock. Answer the result of the block's evaluation.

matched Answer whether the regular expression was matched

1.147 RootNamespace

Defined in namespace Smalltalk
Superclass: AbstractNamespace
Category: Language-Implementation
I am a special form of dictionary. Classes hold on an instance of me; it is called their 'environment'.

1.147.1 RootNamespace class: instance creation

new: spaceName
Create a new root namespace with the given name, and add to Smalltalk a key that references it.

1.147.2 RootNamespace: namespace hierarchy

siblings Answer all the other root namespaces

siblingsDo: aBlock
Evaluate aBlock once for each of the other root namespaces, passing the namespace as a parameter.

1.147.3 RootNamespace: overrides for superspaces

inheritedKeys
Answer a Set of all the keys in the receiver and its superspaces

set: key to: newValue ifAbsent: aBlock
Assign newValue to the variable named as specified by 'key'. This method won't define a new variable; instead if the key is not found it will search in superspaces and evaluate aBlock if it is not found. Answer newValue.

1.147.4 RootNamespace: printing

nameIn: aNamespace
Answer Smalltalk code compiling to the receiver when the current namespace is aNamespace

printOn: aStream in: aNamespace
Print on aStream some Smalltalk code compiling to the receiver when the current namespace is aNamespace

storeOn: aStream
Store Smalltalk code compiling to the receiver

1.148 RunArray

Defined in namespace Smalltalk
Superclass: OrderedCollection
Category: Collections-Sequenceable
My instances are OrderedCollections that automatically apply Run Length Encoding compression to the things they store. Be careful when using me: I can provide great space savings, but my instances don't grant linear access time. RunArray's behavior currently is similar to that of OrderedCollection (you can add elements to RunArrays); maybe it should behave like an ArrayedCollection.

1.148.1 RunArray class: instance creation

new Answer an empty RunArray

new: aSize
Answer a RunArray with space for aSize runs

1.148.2 RunArray: accessing

at: anIndex
Answer the element at index anIndex

at: anIndex put: anObject
Replace the element at index anIndex with anObject and answer anObject

1.148.3 RunArray: adding

add: anObject afterIndex: anIndex
Add anObject after the element at index anIndex

addAll: aCollection afterIndex: anIndex
Add all the elements of aCollection after the one at index anIndex. If aCollection is unordered, its elements could be added in an order which is not the #do: order

addAllFirst: aCollection
Add all the elements of aCollection at the beginning of the receiver. If aCollection is unordered, its elements could be added in an order which is not the #do: order

addAllLast: aCollection
Add all the elements of aCollection at the end of the receiver. If aCollection is unordered, its elements could be added in an order which is not the #do: order

addFirst: anObject
Add anObject at the beginning of the receiver. Watch out: this operation can cause serious performance pitfalls

addLast: anObject
Add anObject at the end of the receiver

1.148.4 RunArray: basic

first Answer the first element in the receiver

last Answer the last element of the receiver

size Answer the number of elements in the receiver

1.148.5 RunArray: copying

deepCopy Answer a copy of the receiver containing copies of the receiver's elements (-#copy is used to obtain them)

shallowCopy
Answer a copy of the receiver. The elements are not copied

1.148.6 RunArray: enumerating

do: aBlock
Enumerate all the objects in the receiver, passing each one to aBlock

objectsAndRunLengthsDo: aBlock
Enumerate all the runs in the receiver, passing to aBlock two parameters for every run: the first is the repeated object, the second is the number of copies

1.148.7 RunArray: removing

removeAtIndex: anIndex
Remove the object at index anIndex from the receiver and answer the removed object

removeFirst
Remove the first object from the receiver and answer the removed object

removeLast
Remove the last object from the receiver and answer the removed object

1.148.8 RunArray: searching

indexOf: anObject startingAt: anIndex ifAbsent: aBlock
Answer the index of the first copy of anObject in the receiver, starting the search at the element at index anIndex. If no equal object is found, answer the result of evaluating aBlock

1.148.9 RunArray: testing

= anObject
Answer true if the receiver is equal to anObject

hash Answer an hash value for the receiver

1.149 ScaledDecimal

Defined in namespace Smalltalk
Superclass: Number
Category: Language-Data types
ScaledDecimal provides a numeric representation of fixed point decimal numbers able to accurately represent decimal fractions. It supports unbounded precision, with no limit to the number of digits before and after the decimal point.

1.149.1 ScaledDecimal class: instance creation

newFromNumber: aNumber scale: scale
Answer a new instance of ScaledDecimal, representing a decimal fraction with a decimal representation considered valid up to the scale-th digit.

1.149.2 ScaledDecimal: arithmetic

*** aNumber**
Multiply two numbers and answer the result.

+ aNumber
Sum two numbers and answer the result.

- aNumber
Subtract aNumber from the receiver and answer the result.

/ aNumber
Divide two numbers and answer the result.

// aNumber
Answer the integer quotient after dividing the receiver by aNumber with truncation towards negative infinity.

\\ aNumber
Answer the remainder after integer division the receiver by aNumber with truncation towards negative infinity.

1.149.3 ScaledDecimal: coercion

asCNumber
Convert the receiver to a kind of number that is understood by the C call-out mechanism.

asFloatD Answer the receiver, converted to a FloatD

asFloatE Answer the receiver, converted to a FloatE

asFloatQ Answer the receiver, converted to a FloatQ

asFraction Answer the receiver, converted to a Fraction

ceiling Answer the receiver, converted to an Integer and truncated towards +infinity.

coerce: aNumber
Answer aNumber, converted to a ScaledDecimal with the same scale as the receiver.

fractionPart
Answer the fractional part of the receiver.

generality Return the receiver's generality

integerPart
Answer the fractional part of the receiver.

truncated Answer the receiver, converted to an Integer and truncated towards -infinity.

1.149.4 ScaledDecimal: comparing

< aNumber
Answer whether the receiver is less than arg.

<= aNumber
Answer whether the receiver is less than or equal to arg.

= arg Answer whether the receiver is equal to arg.

> aNumber
Answer whether the receiver is greater than arg.

>= aNumber
Answer whether the receiver is greater than or equal to arg.

hash Answer an hash value for the receiver.

~= arg Answer whether the receiver is not equal arg.

1.149.5 ScaledDecimal: constants

one Answer the receiver's representation of one.

zero Answer the receiver's representation of zero.

1.149.6 ScaledDecimal: printing

displayOn: aStream
Print a representation of the receiver on aStream, intended to be directed to a user. In this particular case, the 'scale' part of the #printString is not emitted.

printOn: aStream
Print a representation of the receiver on aStream.

1.149.7 ScaledDecimal: storing

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

storeLiteralOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

storeOn: aStream
Print Smalltalk code that compiles to the receiver on aStream.

1.150 SecurityPolicy

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Security
I am the class that represents which operations that could harm the system's security are allowed or denied to a particular class. If a class does not have a policy, it is allowed everything if it is trusted, and denied everything if it is untrusted

1.150.1 SecurityPolicy: modifying

addPermission: aPermission
Not commented.

owner: aClass
Not commented.

removePermission: aPermission
Not commented.

withOwner: aClass
Not commented.

1.150.2 SecurityPolicy: querying

check: aPermission
Not commented.

implies: aPermission
Not commented.

1.151 Semaphore

Defined in namespace Smalltalk
Superclass: LinkedList
Category: Language-Processes
My instances represent counting semaphores. I provide methods for signalling the semaphore's availability, and methods for waiting for its availability. I also provide some methods for implementing critical sections.

1.151.1 Semaphore class: instance creation

forMutualExclusion
Answer a new semaphore with a signal on it. These semaphores are a useful shortcut when you use semaphores as critical sections.

new Answer a new semaphore

1.151.2 Semaphore: accessing

name Answer a user-friendly name for the receiver

name: aString
Answer a user-friendly name for the receiver

waitingProcesses
Answer an Array of processes currently waiting on the receiver.

wouldBlock
Answer whether waiting on the receiver would suspend the current process.

1.151.3 Semaphore: builtins

lock Without putting the receiver to sleep, force processes that try to wait on the semaphore to block. Answer whether this was the case even before.

notify Resume one of the processes that were waiting on the semaphore if there were any. Do not leave a signal on the semaphore if no process is waiting.

notifyAll Resume all the processes that were waiting on the semaphore if there were any. Do not leave a signal on the semaphore if no process is waiting.

signal Signal the receiver, resuming a waiting process' if there is one

wait Wait for the receiver to be signalled, suspending the executing process if it is not yet. Return nil if the wait was interrupted, the receiver otherwise.

waitAfterSignalling: aSemaphore
Signal aSemaphore then, atomically, wait for the receiver to be signalled, suspending the executing process if it is not yet. This is needed to avoid race conditions when the #notify and #notifyAll are used before waiting on receiver: otherwise, if a process sends any of the two between the time aSemaphore is signaled and the time the process starts waiting on the receiver, the notification is lost.

1.151.4 Semaphore: mutual exclusion

critical: aBlock
Wait for the receiver to be free, execute aBlock and signal the receiver again. Return the result of evaluating aBlock.

1.151.5 Semaphore: printing

printOn: aStream
Print a human-readable represention of the receiver on aStream.

1.152 SequenceableCollection

Defined in namespace Smalltalk
Superclass: Collection
Category: Collections-Sequenceable
My instances represent collections of objects that are ordered. I provide some access and manipulation methods.

1.152.1 SequenceableCollection class: instance creation

join: aCollection separatedBy: sepCollection
Where aCollection is a collection of SequenceableCollections, answer a new instance with all the elements therein, in order, each separated by an occurrence of sepCollection.

1.152.2 SequenceableCollection: basic

after: oldObject
Return the element after oldObject. Error if oldObject not found or if no following object is available

allButFirst
Answer a copy of the receiver without the first object.

allButFirst: n
Answer a copy of the receiver without the first n objects.

allButLast Answer a copy of the receiver without the last object.

allButLast: n
Answer a copy of the receiver without the last n objects.

at: anIndex ifAbsent: aBlock
Answer the anIndex-th item of the collection, or evaluate aBlock and answer the result if the index is out of range

atAll: keyCollection
Answer a collection of the same kind returned by #collect:, that only includes the values at the given indices. Fail if any of the values in keyCollection is out of bounds for the receiver.

atAll: aCollection put: anObject
Put anObject at every index contained in aCollection

atAllPut: anObject
Put anObject at every index in the receiver

atRandom Return a random item of the receiver.

before: oldObject
Return the element before oldObject. Error if oldObject not found or if no preceding object is available

first Answer the first item in the receiver

first: n Answer the first n items in the receiver

fourth Answer the fourth item in the receiver

identityIncludes: anObject
Answer whether we include the anObject object

identityIndexOf: anElement
Answer the index of the first occurrence of an object identical to anElement in the receiver. Answer 0 if no item is found

identityIndexOf: anElement ifAbsent: exceptionBlock
Answer the index of the first occurrence of an object identical to anElement in the receiver. Invoke exceptionBlock and answer its result if no item is found

identityIndexOf: anElement startingAt: anIndex
Answer the first index > anIndex which contains an object identical to anElement. Answer 0 if no item is found

identityIndexOf: anObject startingAt: anIndex ifAbsent: exceptionBlock
Answer the first index > anIndex which contains an object exactly identical to anObject. Invoke exceptionBlock and answer its result if no item is found

identityIndexOfLast: anElement ifAbsent: exceptionBlock
Answer the last index which contains an object identical to anElement. Invoke exceptionBlock and answer its result if no item is found

includes: anObject
Answer whether we include anObject

indexOf: anElement
Answer the index of the first occurrence of anElement in the receiver. Answer 0 if no item is found

indexOf: anElement ifAbsent: exceptionBlock
Answer the index of the first occurrence of anElement in the receiver. Invoke exceptionBlock and answer its result if no item is found

indexOf: anElement startingAt: anIndex
Answer the first index > anIndex which contains anElement. Answer 0 if no item is found

indexOf: anElement startingAt: anIndex ifAbsent: exceptionBlock
Answer the first index > anIndex which contains anElement. Invoke exceptionBlock and answer its result if no item is found

indexOfLast: anElement ifAbsent: exceptionBlock
Answer the last index which contains anElement. Invoke exceptionBlock and answer its result if no item is found

indexOfSubCollection: aSubCollection
Answer the first index > anIndex at which starts a sequence of items matching aSubCollection. Answer 0 if no such sequence is found.

indexOfSubCollection: aSubCollection ifAbsent: exceptionBlock
Answer the first index > anIndex at which starts a sequence of items matching aSubCollection. Answer 0 if no such sequence is found.

indexOfSubCollection: aSubCollection startingAt: anIndex
Answer the first index > anIndex at which starts a sequence of items matching aSubCollection. Answer 0 if no such sequence is found.

indexOfSubCollection: aSubCollection startingAt: anIndex ifAbsent: exceptionBlock
Answer the first index > anIndex at which starts a sequence of items matching aSubCollection. Invoke exceptionBlock and answer its result if no such sequence is found

last Answer the last item in the receiver

last: n Answer the last n items in the receiver

second Answer the second item in the receiver

third Answer the third item in the receiver

1.152.3 SequenceableCollection: comparing

endsWith: aSequenceableCollection
Returns true if the receiver ends with the same characters as aSequenceableCollection.

startsWith: aSequenceableCollection
Returns true if the receiver starts with the same characters as aSequenceableCollection.

1.152.4 SequenceableCollection: concatenating

join: sepCollection
Answer a new collection like my first element, with all the elements (in order) of all my elements (which should be collections) separated by sepCollection.

I use my first element instead of myself as a prototype because my elements are more likely to share the desired properties than I am, such as in:

#('hello,' 'world') join: ' ' => 'hello, world'

with: aSequenceableCollection
Return an Array with the same size as the receiver and aSequenceableCollection, each element of which is a 2-element Arrays including one element from the receiver and one from aSequenceableCollection.

with: seqColl1 with: seqColl2
Return an Array with the same size as the receiver and the arguments, each element of which is a 3-element Arrays including one element from the receiver and one from each argument.

with: seqColl1 with: seqColl2 with: seqColl3
Return an Array with the same size as the receiver and the arguments, each element of which is a 4-element Arrays including one element from the receiver and one from each argument.

1.152.5 SequenceableCollection: copying SequenceableCollections

copyAfter: anObject
Answer a new collection holding all the elements of the receiver after the first occurrence of anObject, up to the last.

copyAfterLast: anObject
Answer a new collection holding all the elements of the receiver after the last occurrence of anObject, up to the last.

copyFrom: start
Answer a new collection containing all the items in the receiver from the start-th.

copyFrom: start to: stop
Answer a new collection containing all the items in the receiver from the start-th and to the stop-th

copyReplaceAll: oldSubCollection with: newSubCollection
Answer a new collection in which all the sequences matching oldSubCollection are replaced with newSubCollection

copyReplaceFrom: start to: stop with: replacementCollection
Answer a new collection of the same class as the receiver that contains the same elements as the receiver, in the same order, except for elements from index 'start' to index 'stop'.

If start < stop, these are replaced by the contents of the replacementCollection. Instead, If start = (stop + 1), like in 'copyReplaceFrom: 4 to: 3 with: anArray', then every element of the receiver will be present in the answered copy; the operation will be an append if stop is equal to the size of the receiver or, if it is not, an insert before index 'start'.

copyReplaceFrom: start to: stop withObject: anObject
Answer a new collection of the same class as the receiver that contains the same elements as the receiver, in the same order, except for elements from index 'start' to index 'stop'.

If start < stop, these are replaced by stop-start+1 copies of anObject. Instead, If start = (stop + 1), then every element of the receiver will be present in the answered copy; the operation will be an append if stop is equal to the size of the receiver or, if it is not, an insert before index 'start'.

copyUpTo: anObject
Answer a new collection holding all the elements of the receiver from the first up to the first occurrence of anObject, excluded.

copyUpToLast: anObject
Answer a new collection holding all the elements of the receiver from the first up to the last occurrence of anObject, excluded.

copyWithFirst: anObject
Answer a new collection holding all the elements of the receiver after the first occurrence of anObject, up to the last.

1.152.6 SequenceableCollection: enumerating

anyOne Answer an unspecified element of the collection.

do: aBlock
Evaluate aBlock for all the elements in the sequenceable collection

do: aBlock separatedBy: sepBlock
Evaluate aBlock for all the elements in the sequenceable collection. Between each element, evaluate sepBlock without parameters.

doWithIndex: aBlock
Evaluate aBlock for all the elements in the sequenceable collection, passing the index of each element as the second parameter. This method is mantained for backwards compatibility and is not mandated by the ANSI standard; use #keysAndValuesDo:

findFirst: aBlock
Returns the index of the first element of the sequenceable collection for which aBlock returns true, or 0 if none

findLast: aBlock
Returns the index of the last element of the sequenceable collection for which aBlock returns true, or 0 if none does

fold: binaryBlock
First, pass to binaryBlock the first and second elements of the receiver; for each subsequent element, pass the result of the previous evaluation and an element. Answer the result of the last invocation, or the first element if the collection has size 1. Fail if the collection is empty.

from: startIndex to: stopIndex do: aBlock
Evaluate aBlock for all the elements in the sequenceable collection whose indices are in the range index to stopIndex

from: startIndex to: stopIndex doWithIndex: aBlock
Evaluate aBlock for all the elements in the sequenceable collection whose indices are in the range index to stopIndex, passing the index of each element as the second parameter. This method is mantained for backwards compatibility and is not mandated by the ANSI standard; use #from:to:keysAndValuesDo:

from: startIndex to: stopIndex keysAndValuesDo: aBlock
Evaluate aBlock for all the elements in the sequenceable collection whose indices are in the range index to stopIndex, passing the index of each element as the first parameter and the element as the second.

keys Return an Interval corresponding to the valid indices in the receiver.

keysAndValuesDo: aBlock
Evaluate aBlock for all the elements in the sequenceable collection, passing the index of each element as the first parameter and the element as the second.

readStream
Answer a ReadStream streaming on the receiver

readWriteStream
Answer a ReadWriteStream which streams on the receiver

reverse Answer the receivers' contents in reverse order

reverseDo: aBlock
Evaluate aBlock for all elements in the sequenceable collection, from the last to the first.

with: aSequenceableCollection collect: aBlock
Evaluate aBlock for each pair of elements took respectively from the receiver and from aSequenceableCollection; answer a collection of the same kind of the receiver, made with the block's return values. Fail if the receiver has not the same size as aSequenceableCollection.

with: aSequenceableCollection do: aBlock
Evaluate aBlock for each pair of elements took respectively from the receiver and from aSequenceableCollection. Fail if the receiver has not the same size as aSequenceableCollection.

1.152.7 SequenceableCollection: manipulation

swap: anIndex with: anotherIndex
Swap the item at index anIndex with the item at index another index

1.152.8 SequenceableCollection: replacing items

replaceAll: anObject with: anotherObject
In the receiver, replace every occurrence of anObject with anotherObject.

replaceFrom: start to: stop with: replacementCollection
Replace the items from start to stop with replacementCollection's items from 1 to stop-start+1 (in unexpected order if the collection is not sequenceable).

replaceFrom: start to: stop with: replacementCollection startingAt: repStart
Replace the items from start to stop with replacementCollection's items from repStart to repStart+stop-start

replaceFrom: anIndex to: stopIndex withObject: replacementObject
Replace every item from start to stop with replacementObject.

1.152.9 SequenceableCollection: sorting

sort Sort the contents of the receiver according to the default sort block, which uses #<= to compare items.

sort: sortBlock
Sort the contents of the receiver according to the given sort block, which accepts pair of items and returns true if the first item is less than the second one.

sorted Return a copy of the receiver sorted according to the default sort block, which uses #<= to compare items.

sorted: sortBlock
Return a copy of the receiver sorted according to the given sort block, which accepts pair of items and returns true if the first item is less than the second one.

1.152.10 SequenceableCollection: still unclassified

nextPutAllOn: aStream
Write all the objects in the receiver to aStream

1.152.11 SequenceableCollection: testing

= aCollection
Answer whether the receiver's items match those in aCollection

examineOn: aStream
Print all the instance variables and context of the receiver on aStream

hash Answer an hash value for the receiver

isSequenceable
Answer whether the receiver can be accessed by a numeric index with #at:/-#at:put:.

1.152.12 SequenceableCollection: testing collections

size Answer a dummy size of 0, so that SequenceableCollection>>#do: works.

1.153 Set

Defined in namespace Smalltalk
Superclass: HashedCollection
Category: Collections-Unordered
I am the typical set object; I also known how to do arithmetic on my instances.

1.153.1 Set: arithmetic

& aSet Compute the set intersection of the receiver and aSet.

+ aSet Compute the set union of the receiver and aSet.

- aSet Compute the set difference of the receiver and aSet.

1.153.2 Set: awful ST-80 compatibility hacks

findObjectIndex: object
: Tries to see if anObject exists as an indexed variable. As soon as nil or anObject is found, the index of that slot is answered

1.153.3 Set: comparing

< aSet Answer whether the receiver is a strict subset of aSet

<= aSet Answer whether the receiver is a subset of aSet

> aSet Answer whether the receiver is a strict superset of aSet

>= aSet Answer whether the receiver is a superset of aSet

1.154 SharedQueue

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Processes

My instances provide a guaranteed safe mechanism to allow for communication between processes. All access to the underlying data structures is controlled with critical sections so that things proceed smoothly.

1.154.1 SharedQueue class: instance creation

new Create a new instance of the receiver

sortBlock: sortBlock
: Create a new instance of the receiver which implements a priority queue with the given sort block

1.154.2 SharedQueue: accessing

isEmpty Answer whether there is an object on the queue

next Wait for an object to be on the queue, then remove it and answer it

nextPut: value
: Put value on the queue and answer it

peek Wait for an object to be on the queue if necessary, then answer the same object that #next would answer without removing it.

1.155 SingletonProxy

Defined in namespace Smalltalk
Superclass: AlternativeObjectProxy
Category: Streams-Files

I am a proxy that stores the class of an object rather than the object itself, and pretends that a registered instance (which most likely is a singleton instance of the stored class) was stored instead.

1.155.1 SingletonProxy class: accessing

acceptUsageForClass: aClass
The receiver was asked to be used as a proxy for the class aClass. The registration is fine if the class is actually a singleton.

1.155.2 SingletonProxy class: instance creation

on: anObject
Answer a proxy to be used to save anObject. The proxy stores the class and restores the object by looking into a dictionary of class -> singleton objects.

1.155.3 SingletonProxy: saving and restoring

object Reconstruct the object stored in the proxy and answer it; the binaryRepresentationObject is sent the #reconstructOriginalObject message, and the resulting object is sent the #postLoad message.

1.156 SmallInteger

Defined in namespace Smalltalk
Superclass: Integer
Category: Language-Data types
I am the integer class of the GNU Smalltalk system. My instances can represent signed 30 bit integers and are as efficient as possible.

1.156.1 SmallInteger class: getting limits

bits Answer the number of bits (excluding the sign) that can be represented directly in an object pointer

largest Answer the largest integer represented directly in an object pointer

smallest Answer the smallest integer represented directly in an object pointer

1.156.2 SmallInteger class: testing

isIdentity Answer whether x = y implies x == y for instances of the receiver

1.156.3 SmallInteger: bit arithmetic

highBit Return the index of the highest order 1 bit of the receiver

lowBit Return the index of the lowest order 1 bit of the receiver.

1.156.4 SmallInteger: built ins

*** arg** Multiply the receiver and arg and answer another Number

+ arg Sum the receiver and arg and answer another Number

- arg Subtract arg from the receiver and answer another Number

/ arg Divide the receiver by arg and answer another Integer or Fraction

// arg Dividing receiver by arg (with truncation towards -infinity) and answer the result

< arg Answer whether the receiver is less than arg

<= arg Answer whether the receiver is less than or equal to arg

= arg Answer whether the receiver is equal to arg

== arg Answer whether the receiver is the same object as arg

> arg Answer whether the receiver is greater than arg

>= arg Answer whether the receiver is greater than or equal to arg

\\ arg Calculate the remainder of dividing receiver by arg (with truncation towards -infinity) and answer it

asFloatD Convert the receiver to a FloatD, answer the result

asFloatE Convert the receiver to a FloatE, answer the result

asFloatQ Convert the receiver to a FloatQ, answer the result

asObject Answer the object whose index is in the receiver, nil if there is a free object, fail if index is out of bounds

asObjectNoFail
Answer the object whose index is in the receiver, or nil if no object is found at that index

bitAnd: arg
Do a bitwise AND between the receiver and arg, answer the result

bitOr: arg Do a bitwise OR between the receiver and arg, answer the result

bitShift: arg
Shift the receiver by arg places to the left if arg > 0, by arg places to the right if arg < 0, answer another Number

bitXor: arg
Do a bitwise XOR between the receiver and arg, answer the result

divExact: arg
Dividing receiver by arg assuming that the remainder is zero, and answer the result

nextValidOop
Answer the index of the first non-free OOP after the receiver. This is used internally; it is placed here to avoid polluting Object.

quo: arg Dividing receiver by arg (with truncation towards zero) and answer the result

~= arg Answer whether the receiver is not equal to arg

~~ arg Answer whether the receiver is not the same object as arg

1.156.5 SmallInteger: builtins

at: anIndex
Answer the index-th indexed instance variable of the receiver. This method always fails.

at: anIndex put: value
Store value in the index-th indexed instance variable of the receiver This method always fails.

basicAt: anIndex
Answer the index-th indexed instance variable of the receiver. This method always fails.

basicAt: anIndex put: value
Store value in the index-th indexed instance variable of the receiver This method always fails.

scramble Answer the receiver with its bits mixed and matched.

1.156.6 SmallInteger: coercion

asCNumber
Convert the receiver to a kind of number that is understood by the C call-out mechanism.

1.156.7 SmallInteger: coercion methods

generality Return the receiver's generality

unity Coerce 1 to the receiver's class

zero Coerce 0 to the receiver's class

1.156.8 SmallInteger: testing functionality

isSmallInteger
Answer 'true'.

1.157 SortedCollection

Defined in namespace Smalltalk
Superclass: OrderedCollection
Category: Collections-Sequenceable
I am a collection of objects, stored and accessed according to some sorting criteria. I store things using heap sort and quick sort. My instances have a comparison block associated with them; this block takes two arguments and is a predicate which returns true if the first argument should be sorted earlier than the second. The default block is [:a :b | a <= b], but I will accept any block that conforms to the above criteria – actually any object which responds to #value:value:.

1.157.1 SortedCollection class: hacking

defaultSortBlock
Answer a default sort block for the receiver.

1.157.2 SortedCollection class: instance creation

new Answer a new collection with a default size and sort block

new: aSize
Answer a new collection with a default sort block and the given size

sortBlock: aSortBlock
Answer a new collection with a default size and the given sort block

1.157.3 SortedCollection: basic

last Answer the last item of the receiver

removeLast
Remove an object from the end of the receiver. Fail if the receiver is empty

sortBlock Answer the receiver's sort criteria

sortBlock: aSortBlock
Change the sort criteria for a sorted collection, resort the elements of the collection, and return it.

1.157.4 SortedCollection: copying

copyEmpty: newSize
Answer an empty copy of the receiver, with the same sort block as the receiver

1.157.5 SortedCollection: disabled

add: anObject afterIndex: i
This method should not be called for instances of this class.

addAll: aCollection afterIndex: i
This method should not be called for instances of this class.

addAllFirst: aCollection
This method should not be called for instances of this class.

addAllLast: aCollection
This method should not be called for instances of this class.

addFirst: anObject
This method should not be called for instances of this class.

addLast: anObject
This method should not be called for instances of this class.

at: index put: anObject
This method should not be called for instances of this class.

1.157.6 SortedCollection: enumerating

beConsistent
Prepare the receiver to be walked through with #do: or another enumeration method.

1.157.7 SortedCollection: saving and loading

postLoad Restore the default sortBlock if it is nil

preStore Store the default sortBlock as nil

1.157.8 SortedCollection: searching

includes: anObject
Private - Answer whether the receiver includes an item which is equal to anObject

indexOf: anObject startingAt: index ifAbsent: aBlock
Answer the first index > anIndex which contains anElement. Invoke exceptionBlock and answer its result if no item is found

occurrencesOf: anObject
Answer how many occurrences of anObject can be found in the receiver

1.157.9 SortedCollection: sorting

sort Sort the contents of the receiver according to the given sort block, which accepts pair of items and returns true if the first item is less than the second one. Fails if the collections's sort block is not the same as the default sort block.

sort: sortBlock
Sort the contents of the receiver according to the given sort block, which accepts pair of items and returns true if the first item is less than the second one. Fails if the sort block is not the same as the collection's sort block.

1.158 Stream

Defined in namespace Smalltalk
Superclass: Iterable
Category: Streams
I am an abstract class that provides interruptable sequential access to objects. I can return successive objects from a source, or accept successive objects and store them sequentially on a sink. I provide some simple iteration over the contents of one of my instances, and provide for writing collections sequentially.

1.158.1 Stream: accessing-reading

contents Answer the whole contents of the receiver, from the next object to the last

file Return nil by default; not all streams have a file.

name Return nil by default; not all streams have a name.

next Return the next object in the receiver

next: anInteger
Return the next anInteger objects in the receiver

nextAvailable: anInteger
Return up to anInteger objects in the receiver. Besides stopping if the end of the stream is reached, this may return less than this number of bytes for various reasons. For example, on files and sockets this operation could be non-blocking, or could do at most one I/O operation.

nextAvailable: anInteger into: aCollection startingAt: pos
Place the next anInteger objects from the receiver into aCollection, starting at position pos. Return the number of items stored. Besides stopping if the end of the stream is reached, this may return less than this number of bytes for various reasons. For example, on files and sockets this operation could be non-blocking, or could do at most one I/O operation.

nextAvailable: anInteger putAllOn: aStream
Copy up to anInteger objects in the receiver to aStream. Besides stopping if the end of the stream is reached, this may return less than this number of bytes for various reasons. For example, on files and sockets this operation could be non-blocking, or could do at most one I/O operation.

nextLine Returns a collection of the same type that the stream accesses, containing the next line up to the next new-line character. Returns the entire rest of the stream's contents if no new-line character is found.

nextMatchFor: anObject
Answer whether the next object is equal to anObject. Even if it does not, anObject is lost

splitAt: anObject
Answer an OrderedCollection of parts of the receiver. A new (possibly empty) part starts at the start of the receiver, or after every occurrence of an object which is equal to anObject (as compared by #=).

upTo: anObject
Returns a collection of the same type that the stream accesses, up to but not including the object anObject. Returns the entire rest of the stream's contents if anObject is not present.

upToAll: aCollection
If there is a sequence of objects remaining in the stream that is equal to the sequence in aCollection, set the stream position just past that sequence and answer the elements up to, but not including, the sequence. Else, set the stream position to its end and answer all the remaining elements.

upToEnd Answer every item in the collection on which the receiver is streaming, from the next one to the last

1.158.2 Stream: accessing-writing

next: anInteger put: anObject
Write anInteger copies of anObject to the receiver

next: n putAll: aCollection startingAt: start
Write n objects to the stream, reading them from aCollection and starting at the start-th item.

nextPut: anObject
Write anObject to the receiver

nextPutAll: aCollection
Write all the objects in aCollection to the receiver

nextPutAllFlush: aCollection
Put all the elements of aCollection in the stream, then flush the buffers if supported by the stream.

1.158.3 Stream: basic

species Answer ‘Array’.

1.158.4 Stream: buffering

next: anInteger into: answer startingAt: pos
Read up to anInteger bytes from the stream and store them into answer. Return the number of bytes that were read, raising an exception if we could not read the full amount of data.

next: anInteger putAllOn: aStream
Read up to anInteger bytes from the stream and store them into aStream. Return the number of bytes that were read, raising an exception if we could not read the full amount of data.

1.158.5 Stream: built ins

fileIn File in the contents of the receiver. During a file in operation, global variables (starting with an uppercase letter) that are not declared don’t yield an ‘unknown variable’ error. Instead, they are defined as nil in the ‘Undeclared’ dictionary (a global variable residing in Smalltalk). As soon as you add the variable to a namespace (for example by creating a class) the Association will be removed from Undeclared and reused in the namespace, so that the old references will automagically point to the new value.

fileInLine: lineNum file: aFile at: charPosInt
Private - Much like a preprocessor #line directive; it is used internally by #fileIn, and explicitly by the Emacs Smalltalk mode.

fileInLine: lineNum fileName: aString at: charPosInt
Private - Much like a preprocessor #line directive; it is used internally by #fileIn, and explicitly by the Emacs Smalltalk mode.

1.158.6 Stream: character writing

cr Store a cr on the receiver

crTab Store a cr and a tab on the receiver

encoding Answer the encoding to be used when storing Unicode characters.

isUnicode Answer whether the receiver is able to store Unicode characters. Note that if this method returns true, the stream may or may not be able to store Characters (as opposed to UnicodeCharacters) whose value is above 127.

nl Store a new line on the receiver

nlTab Store a new line and a tab on the receiver

space Store a space on the receiver

space: n Store n spaces on the receiver

tab Store a tab on the receiver

tab: n Store n tabs on the receiver

1.158.7 Stream: compiling

segmentFrom: startPos to: endPos
Answer an object that, when sent #asString, will yield the result of sending 'copyFrom: startPos to: endPos' to the receiver

1.158.8 Stream: concatenating

with: aStream
Return a new Stream whose elements are 2-element Arrays, including one element from the receiver and one from aStream.

with: stream1 with: stream2
Return a new Stream whose elements are 3-element Arrays, including one element from the receiver and one from each argument.

with: stream1 with: stream2 with: stream3
Return a new Stream whose elements are 3-element Arrays, including one element from the receiver and one from each argument.

1.158.9 Stream: enumerating

do: aBlock
Evaluate aBlock once for every object in the receiver

linesDo: aBlock
Evaluate aBlock once for every line in the receiver (assuming the receiver is streaming on Characters).

1.158.10 Stream: filing out

fileOut: aClass
File out aClass on the receiver. If aClass is not a metaclass, file out class and instance methods; if aClass is a metaclass, file out only the class methods

1.158.11 Stream: filtering

, anIterable
: Answer a new stream that concatenates the data in the receiver with the data in aStream. Both the receiver and aStream should be readable.

collect: aBlock
: Answer a new stream that will pass the returned objects through aBlock, and return whatever object is returned by aBlock instead. Note that when peeking in the returned stream, the block will be invoked multiple times, with possibly surprising results.

lines
: Answer a new stream that answers lines from the receiver.

peek
: Returns the next element of the stream without moving the pointer. Returns nil when at end of stream. Lookahead is implemented automatically for streams that are not positionable but can be copied.

peekFor: aCharacter
: Returns true and gobbles the next element from the stream of it is equal to anObject, returns false and doesn't gobble the next element if the next element is not equal to anObject. Lookahead is implemented automatically for streams that are not positionable but can be copied.

reject: aBlock
: Answer a new stream that only returns those objects for which aBlock returns false. Note that the returned stream will not be positionable.

select: aBlock
: Answer a new stream that only returns those objects for which aBlock returns true. Note that the returned stream will not be positionable.

1.158.12 Stream: polymorphism

close
: Do nothing. This is provided for consistency with file streams

flush
: Do nothing. This is provided for consistency with file streams

pastEnd
: The end of the stream has been reached. Signal a Notification.

1.158.13 Stream: positioning

isPositionable
: Answer true if the stream supports moving backwards with #skip:.

skip: anInteger
: Move the position forwards by anInteger places

skipSeparators
: Advance the receiver until we find a character that is not a separator. Answer false if we reach the end of the stream, else answer true; in this case, sending #next will return the first non-separator character (possibly the same to which the stream pointed before #skipSeparators was sent).

skipTo: anObject
Move the current position to after the next occurrence of anObject and return true if anObject was found. If anObject doesn't exist, the pointer is atEnd, and false is returned.

skipToAll: aCollection
If there is a sequence of objects remaining in the stream that is equal to the sequence in aCollection, set the stream position just past that sequence and answer true. Else, set the stream position to its end and answer false.

1.158.14 Stream: printing

<< anObject
This method is a short-cut for #display:; it prints anObject on the receiver by sending displayOn: to anObject. This method is provided so that you can use cascading and obtain better-looking code

display: anObject
Print anObject on the receiver by sending displayOn: to anObject. This method is provided so that you can use cascading and obtain better-looking code

print: anObject
Print anObject on the receiver by sending printOn: to anObject. This method is provided so that you can use cascading and obtain better-looking code

1.158.15 Stream: still unclassified

nextPutAllOn: aStream
Write all the objects in the receiver to aStream

1.158.16 Stream: storing

store: anObject
Print Smalltalk code compiling to anObject on the receiver, by sending storeOn: to anObject. This method is provided so that you can use cascading and obtain better-looking code

1.158.17 Stream: streaming protocol

nextAvailablePutAllOn: aStream
Copy to aStream a more-or-less arbitrary amount of data. When used on files, this does at most one I/O operation. For other kinds of stream, the definition may vary. This method is used to do stream-to-stream copies.

1.158.18 Stream: testing

atEnd Answer whether the stream has got to an end

isExternalStream
Answer whether the receiver streams on a file or socket. By default, answer false.

isSequenceable
Answer whether the receiver can be accessed by a numeric index with #at:/-#at:put:.

readStream
As a wild guess, return the receiver. WriteStreams should override this method.

1.159 String

Defined in namespace Smalltalk
Superclass: CharacterArray
Category: Collections-Text

My instances represent 8-bit character strings. Being a very common case, they are particularly optimized.

Note that, if you care about multilingualization, you should treat String only as an encoded representation of a UnicodeString. The I18N package adds more Unicode-friendliness to the system so that encoding and decoding is performed automatically in more cases. In that case, String represents a case when the encoding is either unknown, irrelevant, or assumed to be the system default.

1.159.1 String class: instance creation

fromCData: aCObject
Answer a String containing the bytes starting at the location pointed to by aCObject, up to the first NUL character.

fromCData: aCObject size: anInteger
Answer a String containing anInteger bytes starting at the location pointed to by aCObject

1.159.2 String class: multibyte encodings

isUnicode Answer false; the receiver stores bytes (i.e. an encoded form), not characters.

1.159.3 String: accessing

byteAt: index
Answer the ascii value of index-th character variable of the receiver

byteAt: index put: value
Store (Character value: value) in the index-th indexed instance variable of the receiver

1.159.4 String: basic

, aString Answer a new instance of an ArrayedCollection containing all the elements in the receiver, followed by all the elements in aSequenceableCollection

= aCollection
Answer whether the receiver's items match those in aCollection

indexOf: anElement startingAt: anIndex
Answer the first index > anIndex which contains anElement. Invoke exceptionBlock and answer its result if no item is found

indexOf: anElement startingAt: anIndex ifAbsent: exceptionBlock
Answer the first index > anIndex which contains anElement. Invoke exceptionBlock and answer its result if no item is found

1.159.5 String: built ins

asCData: aCType
Allocate memory with malloc for a NULL-terminated copy of the receiver, and return a pointer to it as a CObject of the given type.

at: anIndex
Answer the index-th indexed instance variable of the receiver

at: anIndex ifAbsent: aBlock
Answer the index-th indexed instance variable of the receiver

at: anIndex put: value
Store value in the index-th indexed instance variable of the receiver

basicAt: anIndex
Answer the index-th indexed instance variable of the receiver. This method must not be overridden, override at: instead

basicAt: anIndex put: value
Store value in the index-th indexed instance variable of the receiver This method must not be overridden, override at:put: instead

hash Answer an hash value for the receiver

replaceFrom: start to: stop with: aString startingAt: replaceStart
Replace the characters from start to stop with new characters whose ASCII codes are contained in aString, starting at the replaceStart location of aString

replaceFrom: start to: stop withByteArray: byteArray startingAt: replaceStart
Replace the characters from start to stop with new characters whose ASCII codes are contained in byteArray, starting at the replaceStart location of byteArray

similarityTo: aString
Answer a number that denotes the similarity between aString and the receiver. 0 indicates equality, negative numbers indicate some difference. Implemented as a primitive for speed.

size Answer the size of the receiver

1.159.6 String: CObject

asCData Allocate memory with malloc for a NULL-terminated copy of the receiver, and return a pointer to it as a CChar.

1.159.7 String: converting

asByteArray
Return the receiver, converted to a ByteArray of ASCII values

asString But I already am a String! Really!

asSymbol Returns the symbol corresponding to the receiver

encoding Answer the encoding of the receiver. This is not implemented unless you load the Iconv package.

1.159.8 String: filesystem

/ aName Answer a File object as appropriate for a file named 'aName' in the directory represented by the receiver.

asFile Answer a File object for the file whose name is in the receiver.

1.159.9 String: printing

displayOn: aStream
Print a representation of the receiver on aStream. Unlike #printOn:, this method strips extra quotes.

displayString
Answer a String representing the receiver. For most objects this is simply its #printString, but for CharacterArrays and characters, superfluous dollars or extra pair of quotes are stripped.

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

printOn: aStream
Print a representation of the receiver on aStream

storeLiteralOn: aStream
Store a Smalltalk literal compiling to the receiver on aStream

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream

1.159.10 String: regex

=~ pattern
Answer a RegexResults object for matching the receiver against the Regex or String object pattern.

allOccurrencesOfRegex: pattern
Find all the matches of pattern within the receiver and collect them into an OrderedCollection.

allOccurrencesOfRegex: pattern do: aBlock
Find all the matches of pattern within the receiver and pass the RegexResults objects to aBlock.

allOccurrencesOfRegex: pattern from: from to: to
Find all the matches of pattern within the receiver and within the given range of indices. Collect them into an OrderedCollection, which is then returned.

allOccurrencesOfRegex: pattern from: from to: to do: aBlock
Find all the matches of pattern within the receiver and within the given range of indices. For each match, pass the RegexResults object to aBlock.

asRegex Answer the receiver, converted to a Regex object.

copyFrom: from to: to replacingAllRegex: pattern with: aStringOrBlock
Returns the substring of the receiver between from and to. Any match of pattern in that part of the string is replaced using aStringOrBlock as follows: if it is a block, a RegexResults object is passed, while if it is a string, %n sequences are replaced with the captured subexpressions of the match (as in #%).

copyFrom: from to: to replacingRegex: pattern with: aStringOrBlock
Returns the substring of the receiver between from and to. If pattern has a match in that part of the string, the match is replaced using aStringOrBlock as follows: if it is a block, a RegexResults object is passed, while if it is a string, %n sequences are replaced with the captured subexpressions of the match (as in #%).

copyReplacingAllRegex: pattern with: aStringOrBlock
Returns the receiver after replacing all the matches of pattern (if any) using aStringOrBlock as follows: if it is a block, a RegexResults object is passed, while if it is a string, %n sequences are replaced with the captured subexpressions of the match (as in #%).

copyReplacingRegex: pattern with: aStringOrBlock
Returns the receiver after replacing the first match of pattern (if any) using aStringOrBlock as follows: if it is a block, a RegexResults object is passed, while if it is a string, %n sequences are replaced with the captured subexpressions of the match (as in #%).

indexOfRegex: regexString
If an occurrence of the regex is present in the receiver, return the Interval corresponding to the leftmost-longest match. Otherwise return nil.

indexOfRegex: regexString from: from to: to
If an occurrence of the regex is present in the receiver, return the Interval corresponding to the leftmost-longest match occurring within the given range of indices. Otherwise return nil.

indexOfRegex: regexString from: from to: to ifAbsent: excBlock
If an occurrence of the regex is present in the receiver, return the Interval corresponding to the leftmost-longest match occurring within the given indices. Otherwise, evaluate excBlock and return the result.

indexOfRegex: regexString ifAbsent: excBlock
If an occurrence of the regex is present in the receiver, return the Interval corresponding to the leftmost-longest match. Otherwise, evaluate excBlock and return the result.

indexOfRegex: regexString startingAt: index
If an occurrence of the regex is present in the receiver, return the Interval corresponding to the leftmost-longest match starting after the given index. Otherwise return nil.

indexOfRegex: regexString startingAt: index ifAbsent: excBlock
If an occurrence of the regex is present in the receiver, return the Interval corresponding to the leftmost-longest match starting after the given index. Otherwise, evaluate excBlock and return the result.

matchRegex: pattern
Answer whether the receiver is an exact match for the pattern. This means that the pattern is implicitly anchored at the beginning and the end.

matchRegex: pattern from: from to: to
Answer whether the given range of indices is an exact match for the pattern. This means that there is a match starting at from and ending at to (which is not necessarily the longest match starting at from).

occurrencesOfRegex: pattern
Returns count of how many times pattern repeats in the receiver.

occurrencesOfRegex: pattern from: from to: to
Return a count of how many times pattern repeats in the receiver within the given range of index.

occurrencesOfRegex: pattern startingAt: index
Returns count of how many times pattern repeats in the receiver, starting the search at the given index.

onOccurrencesOfRegex: pattern do: body
Find all the matches of pattern within the receiver and, for each match, pass the RegexResults object to aBlock.

onOccurrencesOfRegex: pattern from: from to: to do: aBlock
Find all the matches of pattern within the receiver and within the given range of indices. For each match, pass the RegexResults object to aBlock.

replacingAllRegex: pattern with: aStringOrBlock
Returns the receiver if the pattern has no match in it. Otherwise, any match of pattern in that part of the string is replaced using aStringOrBlock as follows: if it is a block, a RegexResults object is passed, while if it is a string, %n sequences are replaced with the captured subexpressions of the match (as in #%).

replacingRegex: pattern with: aStringOrBlock
Returns the receiver if the pattern has no match in it. If it has a match, it is replaced using aStringOrBlock as follows: if it is a block, a RegexResults object is passed, while if it is a string, %n sequences are replaced with the captured subexpressions of the match (as in #%).

searchRegex: pattern
A synonym for #=~. Answer a RegexResults object for matching the receiver against the Regex or String object pattern.

searchRegex: pattern from: from to: to
Answer a RegexResults object for matching the receiver against the Regex or String object pattern, restricting the match to the specified range of indices.

searchRegex: pattern startingAt: anIndex
Answer a RegexResults object for matching the receiver against the Regex or String object pattern, starting the match at index anIndex.

tokenize: pattern
Split the receiver at every occurrence of pattern. All parts that do not match pattern are separated and stored into an Array of Strings that is returned.

tokenize: pattern from: from to: to
Split the receiver at every occurrence of pattern (considering only the indices between from and to). All parts that do not match pattern are separated and stored into an Array of Strings that is returned.

~ pattern Answer whether the receiver matched against the Regex or String object pattern.

1.159.11 String: still unclassified

escapeRegex
Answer the receiver with all regex special characters escaped by a backslash.

1.159.12 String: testing functionality

isString Answer 'true'.

1.160 Symbol

Defined in namespace Smalltalk
Superclass: String
Category: Language-Implementation
My instances are unique throughout the Smalltalk system. My instances behave for the most part like strings, except that they print differently, and I guarantee that any two instances that have the same printed representation are in fact the same instance.

1.160.1 Symbol class: built ins

intern: aString
Private - Same as 'aString asSymbol'

1.160.2 Symbol class: instance creation

internCharacter: aCharacter
Answer the one-character symbol associated to the given character.

new This method should not be called for instances of this class.

new: size This method should not be called for instances of this class.

with: element1
Answer a collection whose only element is element1

with: element1 with: element2
Answer a collection whose only elements are the parameters in the order they were passed

with: element1 with: element2 with: element3
Answer a collection whose only elements are the parameters in the order they were passed

with: element1 with: element2 with: element3 with: element4
Answer a collection whose only elements are the parameters in the order they were passed

with: element1 with: element2 with: element3 with: element4 with: element5
Answer a collection whose only elements are the parameters in the order they were passed

1.160.3 Symbol class: symbol table

hasInterned: aString ifTrue: aBlock
If aString has not been interned yet, answer false. Else, pass the interned version to aBlock and answer true. Note that this works because String>>-#hash calculates the same hash value used by the VM when interning strings into the SymbolTable. Changing one of the hashing methods without changing the other will break this method.

isSymbolString: aString
Answer whether aString has already been interned. Note that this works because String>>#hash calculates the same hash value used by the VM when interning strings into the SymbolTable. Changing one of the hashing methods without changing the other will break this method.

rebuildTable
Rebuild the SymbolTable, thereby garbage-collecting unreferenced Symbols. While this process is done, preemption is disabled because it is not acceptable to leave the SymbolTable in a partially updated state. Note that this works because String>>#hash calculates the same hash value used by the VM when interning strings into the SymbolTable. Changing one of the hashing methods without changing the other will break this method.

1.160.4 Symbol: accessing the method dictionary

implementors
Answer a Set of all the compiled method associated with selector named by the receiver, which is supposed to be a valid message name.

1.160.5 Symbol: basic

deepCopy Returns a deep copy of the receiver. As Symbols are identity objects, we actually return the receiver itself.

keywords Answer an array of keywords that compose the receiver, which is supposed to be a valid message name (#+, #not, #printOn:, #ifTrue:ifFalse:, etc.)

numArgs Answer the number of arguments supported by the receiver, which is supposed to be a valid message name (#+, #not, #printOn:, #ifTrue:ifFalse:, etc.)

shallowCopy
Returns a deep copy of the receiver. As Symbols are identity objects, we actually return the receiver itself.

1.160.6 Symbol: built ins

= aSymbol
Answer whether the receiver and aSymbol are the same object

hash Answer an hash value for the receiver. Symbols are optimized for speed

1.160.7 Symbol: converting

asString Answer a String with the same characters as the receiver

asSymbol But we are already a Symbol, and furthermore, Symbols are identity objects! So answer the receiver.

1.160.8 Symbol: misc

species Answer 'String'.

1.160.9 Symbol: storing

displayOn: aStream
Print a represention of the receiver on aStream. For most objects this is simply its #printOn: representation, but for strings and characters, superfluous dollars or extra pairs of quotes are stripped.

displayString
Answer a String representing the receiver. For most objects this is simply its #printString, but for strings and characters, superfluous dollars or extra pair of quotes are stripped.

printOn: aStream
Print a represention of the receiver on aStream.

storeLiteralOn: aStream
Print Smalltalk code on aStream that compiles to the same symbol as the receiver.

storeOn: aStream
Print Smalltalk code on aStream that compiles to the same symbol as the receiver.

1.160.10 Symbol: testing

isSimpleSymbol
Answer whether the receiver must be represented in quoted-string (e.g. #'abc-def') form.

1.160.11 Symbol: testing functionality

isString Answer 'false'.

isSymbol Answer 'true'.

1.161 SymLink

Defined in namespace Smalltalk
Superclass: Link
Category: Language-Implementation

I am used to implement the Smalltalk symbol table. My instances are links that contain symbols, and the symbol table basically a hash table that points to chains of my instances.

1.161.1 SymLink class: instance creation

symbol: aSymbol nextLink: aSymLink
Answer a new SymLink, which refers to aSymbol and points to aSymLink as the next SymLink in the chain.

1.161.2 SymLink: accessing

symbol Answer the Symbol that the receiver refers to in the symbol table.

symbol: aSymbol
Set the Symbol that the receiver refers to in the symbol table.

1.161.3 SymLink: iteration

do: aBlock
Evaluate aBlock for each symbol in the list

1.161.4 SymLink: printing

printOn: aStream
Print a representation of the receiver on aStream.

1.162 SystemDictionary

Defined in namespace Smalltalk
Superclass: RootNamespace
Category: Language-Implementation

I am a special namespace. I only have one instance, called "Smalltalk", which is known to the Smalltalk interpreter. I define several methods that are "system" related, such as #quitPrimitive. My instance also helps keep track of dependencies between objects.

1.162.1 SystemDictionary class: initialization

initialize Create the kernel's private namespace.

1.162.2 SystemDictionary: basic

halt Interrupt interpreter

hash Smalltalk usually contains a reference to itself, avoid infinite loops

1.162.3 SystemDictionary: builtins

basicBacktrace
Prints the method invocation stack backtrace, as an aid to debugging

byteCodeCounter
Answer the number of bytecodes executed by the VM

debug This methods provides a way to break in the VM code. Set a breakpoint in _gst_debug and call this method near the point where you think the bug happens.

declarationTrace
Answer whether compiled bytecodes are printed on stdout

declarationTrace: aBoolean
Set whether compiled bytecodes are printed on stdout

executionTrace
Answer whether executed bytecodes are printed on stdout

executionTrace: aBoolean
Set whether executed bytecodes are printed on stdout

getTraceFlag: anIndex
Private - Returns a boolean value which is one of the interpreter's tracing flags

setTraceFlag: anIndex to: aBoolean
Private - Sets the value of one of the interpreter's tracing flags (indicated by 'anIndex') to the value aBoolean.

verboseTrace
Answer whether execution tracing prints the object on the stack top

verboseTrace: aBoolean
Set whether execution tracing prints the object on the stack top

1.162.4 SystemDictionary: c call-outs

environ Not commented.

getArgc Not commented.

getArgv: index
Not commented.

getenv: aString
Not commented.

putenv: aString
Not commented.

system: aString
Not commented.

system: aString withArguments: args
Not commented.

1.162.5 SystemDictionary: command-line

arguments: pattern do: actionBlock
Parse the command-line arguments according to the syntax specified in pattern. For every command-line option found, the two-argument block actionBlock is evaluated passing the option name and the argument. For file names (or in general, other command-line arguments than options) the block's first argument will be nil. For options without arguments, or with unspecified optional arguments, the block's second argument will be nil. The option name will be passed as a character object for short options, and as a string for long options.

If an error is found, nil is returned. For more information on the syntax of pattern, see #arguments:do:ifError:.

arguments: pattern do: actionBlock ifError: errorBlock
Parse the command-line arguments according to the syntax specified in pattern. For every command-line option found, the two-argument block actionBlock is evaluated passing the option name and the argument. For file names (or in general, other command-line arguments than options) the block's first argument will be nil. For options without arguments, or with unspecified optional arguments, the block's second argument will be nil. The option name will be passed as a character object for short options, and as a string for long options.

If an error is found, the parsing is interrupted, errorBlock is evaluated, and the returned value is answered.

Every whitespace-separated part ('word') of pattern specifies a command-line option. If a word ends with a colon, the option will have a mandatory argument. If a word ends with two colons, the option will have an optional argument. Before the colons, multiple option names (either short names like '-l' or long names like '–long') can be specified. Before passing the option to actionBlock, the name will be canonicalized to the last one.

Prefixes of long options are accepted as long as they're unique, and they are canonicalized to the full name before passing it to actionBlock. Additionally, the full name of an option is accepted even if it is the prefix of a longer option.

Mandatory arguments can appear in the next argument, or in the same argument (separated by an = for arguments to long options). Optional arguments must appear in the same argument.

1.162.6 SystemDictionary: miscellaneous

arguments Return the command line arguments after the -a switch

backtrace Print a backtrace on the Transcript.

hostSystem
Answer the triplet corresponding to the system for which GNU Smalltalk was built.

1.162.7 SystemDictionary: printing

nameIn: aNamespace
Answer ''Smalltalk''.

printOn: aStream in: aNamespace
Store Smalltalk code compiling to the receiver

storeOn: aStream
Store Smalltalk code compiling to the receiver

1.162.8 SystemDictionary: profiling

rawProfile: anIdentityDictionary
Set the raw profile to be anIdentityDictionary and return the old one.

1.162.9 SystemDictionary: special accessing

addFeature: aFeature
Add the aFeature feature to the Features set

hasFeatures: features
Returns true if the feature or features in 'features' is one of the implementation dependent features present

removeFeature: aFeature
Remove the aFeature feature to the Features set

version Answer the current version of the GNU Smalltalk environment

1.162.10 SystemDictionary: testing

imageLocal
Answer whether the kernel directory is a subdirectory of the image directory (non-local image) or not.

isSmalltalk
Answer 'true'.

1.163 SystemExceptions.AlreadyDefined

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidArgument
Category: Language-Exceptions
I am raised when one tries to define a symbol (class or pool variable) that is already defined.

1.163.1 SystemExceptions.AlreadyDefined: accessing

description
Answer a description for the error

1.164 SystemExceptions.ArgumentOutOfRange

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidArgument
Category: Language-Exceptions

I am raised when one invokes a method with an argument outside of its valid range.

1.164.1 SystemExceptions.ArgumentOutOfRange class: signaling

signalOn: value mustBeBetween: low and: high
Raise the exception. The given value was not between low and high.

1.164.2 SystemExceptions.ArgumentOutOfRange: accessing

description
Answer a textual description of the exception.

high Answer the highest value that was permitted.

high: aMagnitude
Set the highest value that was permitted.

low Answer the lowest value that was permitted.

low: aMagnitude
Set the lowest value that was permitted.

1.165 SystemExceptions.BadReturn

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.VMError
Category: Language-Exceptions

I am raised when one tries to return from an already-terminated method.

1.165.1 SystemExceptions.BadReturn: accessing

description
Answer a textual description of the exception.

1.166 SystemExceptions.CInterfaceError

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.PrimitiveFailed
Category: Language-Exceptions

I am raised when an error happens that is related to the C interface.

1.166.1 SystemExceptions.CInterfaceError: accessing

description
Answer a textual description of the exception.

1.167 SystemExceptions.EmptyCollection

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidValue
Category: Language-Exceptions

I am raised when one invokes a method on an empty collection.

1.167.1 SystemExceptions.EmptyCollection: accessing

description
Answer a textual description of the exception.

1.168 SystemExceptions.EndOfStream

Defined in namespace Smalltalk.SystemExceptions
Superclass: Notification
Category: Language-Exceptions

I am raised when a stream reaches its end.

1.168.1 SystemExceptions.EndOfStream class: signaling

signalOn: stream
Answer an exception reporting the parameter has reached its end.

1.168.2 SystemExceptions.EndOfStream: accessing

description
Answer a textual description of the exception.

stream Answer the stream whose end was reached.

stream: anObject
Set the stream whose end was reached.

1.169 SystemExceptions.FileError

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.PrimitiveFailed
Category: Language-Exceptions

I am raised when an error happens that is related to the file system.

1.169.1 SystemExceptions.FileError: accessing

description
Answer a textual description of the exception.

1.170 SystemExceptions.IndexOutOfRange

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.ArgumentOutOfRange
Category: Language-Exceptions

I am raised when one invokes am accessor method with an index outside of its valid range.

1.170.1 SystemExceptions.IndexOutOfRange class: signaling

signalOn: aCollection withIndex: value
The given index was out of range in aCollection.

1.170.2 SystemExceptions.IndexOutOfRange: accessing

collection Answer the collection that triggered the error

collection: anObject
Set the collection that triggered the error

description
Answer a textual description of the exception.

messageText
Answer an exception's message text.

1.171 SystemExceptions.InvalidArgument

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidValue
Category: Language-Exceptions
I am raised when one invokes a method with an invalid argument.

1.171.1 SystemExceptions.InvalidArgument: accessing

messageText
Answer an exception's message text.

1.172 SystemExceptions.InvalidProcessState

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidValue
Category: Language-Exceptions
I am an error raised when trying to resume a terminated process, or stuff like that.

1.172.1 SystemExceptions.InvalidProcessState: accessing

description
Answer a textual description of the exception.

1.173 SystemExceptions.InvalidSize

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidArgument
Category: Language-Exceptions
I am raised when an argument has an invalid size.

1.173.1 SystemExceptions.InvalidSize: accessing

description
Answer a textual description of the exception.

1.174 SystemExceptions.InvalidState

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidValue
Category: Language-Exceptions
I am raised when one invokes a method and the receiver or an argument are in an invalid state for the method.

1.174.1 SystemExceptions.InvalidState: accessing

messageText
Answer an exception's message text.

1.175 SystemExceptions.InvalidValue

Defined in namespace Smalltalk.SystemExceptions
Superclass: Error
Category: Language-Exceptions
I am raised when one invokes a method with an invalid receiver or argument.

1.175.1 SystemExceptions.InvalidValue class: signaling

signalOn: value
Answer an exception reporting the parameter as invalid.

signalOn: value reason: reason
Answer an exception reporting 'value' as invalid, for the given reason.

1.175.2 SystemExceptions.InvalidValue: accessing

description
Answer a textual description of the exception.

messageText
Answer an exception's message text.

value Answer the object that was found to be invalid.

value: anObject
Set the object that was found to be invalid.

1.176 SystemExceptions.MustBeBoolean

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.WrongClass
Category: Language-Exceptions
I am raised when one invokes a boolean method on a non-boolean.

1.176.1 SystemExceptions.MustBeBoolean class: signaling

signalOn: anObject
Signal a new exception, with the bad value in question being anObject.

1.177 SystemExceptions.MutationError

Defined in namespace Smalltalk.SystemExceptions
Superclass: Error
Category: Language-Exceptions
I am an error raised when a class is mutated in an invalid way.

1.177.1 SystemExceptions.MutationError class: instance creation

new Create an instance of the receiver, which you will be able to signal later.

1.177.2 SystemExceptions.MutationError: accessing

description
Answer a textual description of the exception.

1.178 SystemExceptions.NoRunnableProcess

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.VMError
Category: Language-Exceptions
I am raised when no runnable process can be found in the image.

1.178.1 SystemExceptions.NoRunnableProcess: accessing

description
Answer a textual description of the exception.

1.179 SystemExceptions.NotEnoughElements

Defined in namespace Smalltalk.SystemExceptions
Superclass: Error
Category: Language-Exceptions
I am raised when one invokes #next: but not enough items remain in the stream.

1.179.1 SystemExceptions.NotEnoughElements class: signaling

signalOn: remainingCount
Answer an exception reporting the parameter as invalid.

1.179.2 SystemExceptions.NotEnoughElements: accessing

description
Answer a textual description of the exception.

messageText
Answer an exception's message text.

remainingCount
Answer the number of items that were to be read.

remainingCount: anObject
Set the number of items that were to be read.

1.180 SystemExceptions.NotFound

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidArgument
Category: Language-Exceptions
I am raised when something is searched without success.

1.180.1 SystemExceptions.NotFound class: accessing

signalOn: value reason: aString
Raise an exception: reason specifies the reason of the exception.

signalOn: value what: aString
Raise an exception; aString specifies what was not found (a key, an object, a class, and so on).

1.180.2 SystemExceptions.NotFound: accessing

description
Answer a textual description of the exception.

1.181 SystemExceptions.NotImplemented

Defined in namespace Smalltalk.SystemExceptions
Superclass: Error
Category: Language-Exceptions
I am raised when a method is called that has not been implemented.

1.181.1 SystemExceptions.NotImplemented: accessing

description
Answer a textual description of the exception.

1.182 SystemExceptions.NotIndexable

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidValue
Category: Language-Exceptions
I am raised when an object is not indexable.

1.182.1 SystemExceptions.NotIndexable: accessing

description
Answer a textual description of the exception.

1.183 SystemExceptions.NotYetImplemented

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.NotImplemented
Category: Language-Exceptions

I am raised when a method is called that has not been implemented yet.

1.183.1 SystemExceptions.NotYetImplemented: accessing

description

Answer a textual description of the exception.

1.184 SystemExceptions.PackageNotAvailable

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.NotFound
Category: Language-Packaging

1.184.1 SystemExceptions.PackageNotAvailable class: still unclassified

signal: aString

Signal an exception saying that the package named aString can't be found.

signal: package reason: reason

Signal an exception saying that be package named package can't be found because the reason named reason.

1.184.2 SystemExceptions.PackageNotAvailable: description

isResumable

Answer true. Package unavailability is resumable, because the package files might just lie elsewhere.

1.185 SystemExceptions.PrimitiveFailed

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.VMError
Category: Language-Exceptions

I am raised when a primitive fails for some reason.

1.185.1 SystemExceptions.PrimitiveFailed: accessing

description

Answer a textual description of the exception.

1.186 SystemExceptions.ProcessBeingTerminated

Defined in namespace Smalltalk.SystemExceptions
Superclass: Notification
Category: Language-Exceptions

I am raised when a process is terminated.

1.186.1 SystemExceptions.ProcessBeingTerminated class: still unclassified

initialize Not commented.

1.186.2 SystemExceptions.ProcessBeingTerminated: accessing

description
Answer a textual description of the exception.

semaphore
If the process was waiting on a semaphore, answer it.

semaphore: aSemaphore
If the process was waiting on a semaphore, answer it.

1.187 SystemExceptions.ProcessTerminated

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidValue
Category: Language-Exceptions
I am raised when somebody tries to resume or interrupt a terminated process.

1.187.1 SystemExceptions.ProcessTerminated: accessing

description
Answer a textual description of the exception.

1.188 SystemExceptions.ReadOnlyObject

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidValue
Category: Language-Exceptions
I am raised when one writes to a read-only object.

1.188.1 SystemExceptions.ReadOnlyObject: accessing

description
Answer a textual description of the exception.

1.189 SystemExceptions.SecurityError

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.VMError
Category: Language-Exceptions
I am an error raised when an untrusted object tries to do an insecure operation.

1.189.1 SystemExceptions.SecurityError class: accessing

signal: aPermission
Raise the exception, setting to aPermission the permission that was tested and failed.

1.189.2 SystemExceptions.SecurityError: accessing

description

Answer a textual description of the exception.

failedPermission

Answer the permission that was tested and that failed.

failedPermission: anObject

Set which permission was tested and failed.

1.190 SystemExceptions.ShouldNotImplement

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.NotImplemented
Category: Language-Exceptions

I am raised when a method is called that a class wishes that is not called.

1.190.1 SystemExceptions.ShouldNotImplement: accessing

description

Answer a textual description of the exception.

1.191 SystemExceptions.SubclassResponsibility

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.ShouldNotImplement
Category: Language-Exceptions

I am raised when a method is called whose implementation is the responsibility of concrete subclass.

1.191.1 SystemExceptions.SubclassResponsibility: accessing

description

Answer a textual description of the exception.

1.192 SystemExceptions.UnhandledException

Defined in namespace Smalltalk.SystemExceptions
Superclass: Exception
Category: Language-Exception

I am raised when a backtrace is shown to terminate the current process.

1.192.1 SystemExceptions.UnhandledException: accessing

defaultAction

Terminate the current process.

description

Answer a textual description of the exception.

originalException

Answer the uncaught exception.

originalException: anObject
Set the uncaught exception to anObject.

1.193 SystemExceptions.UserInterrupt

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.VMError
Category: Language-Exceptions
I am raised when one presses Ctrl-C.

1.193.1 SystemExceptions.UserInterrupt: accessing

description
Answer a textual description of the exception.

1.194 SystemExceptions.VerificationError

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.VMError
Category: Language-Exceptions
I am raised when the verification of a method fails.

1.194.1 SystemExceptions.VerificationError: accessing

description
Answer a textual description of the exception.

1.195 SystemExceptions.VMError

Defined in namespace Smalltalk.SystemExceptions
Superclass: Error
Category: Language-Exceptions
I am an error related to the innards of the system.

1.195.1 SystemExceptions.VMError: accessing

description
Answer a textual description of the exception.

1.196 SystemExceptions.WrongArgumentCount

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.PrimitiveFailed
Category: Language-Exceptions
I am raised when one tries to evaluate a method (via #perform:...) or a block but passes the wrong number of arguments.

1.196.1 SystemExceptions.WrongArgumentCount: accessing

description
Answer a textual description of the exception.

1.197 SystemExceptions.WrongClass

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.InvalidValue
Category: Language-Exceptions

I am raised when an argument is constrained to be an instance of a determinate class, and this constraint is not respected by the caller.

1.197.1 SystemExceptions.WrongClass class: signaling

signalOn: anObject mustBe: aClassOrArray

Raise an exception. The given object should have been an instance of one of the classes indicated by aClassOrArray (which should be a single class or an array of classes). Whether instances of subclasses are allowed should be clear from the context, though in general (i.e. with the exception of a few system messages) they should be.

1.197.2 SystemExceptions.WrongClass: accessing

description

Answer a textual description of the exception.

messageText

Answer an exception's message text.

validClasses

Answer the list of classes whose instances would have been valid.

validClasses: aCollection

Set the list of classes whose instances would have been valid.

validClassesString

Answer the list of classes whose instances would have been valid, formatted as a string.

1.198 SystemExceptions.WrongMessageSent

Defined in namespace Smalltalk.SystemExceptions
Superclass: SystemExceptions.ShouldNotImplement
Category: Language-Exceptions

I am raised when a method is called that a class wishes that is not called. This exception also includes a suggestion on which message should be sent instead

1.198.1 SystemExceptions.WrongMessageSent class: signaling

signalOn: selector useInstead: aSymbol

Raise an exception, signaling which selector was sent and suggesting a valid alternative.

1.198.2 SystemExceptions.WrongMessageSent: accessing

messageText

Answer an exception's message text.

selector Answer which selector was sent.

selector: aSymbol
Set which selector was sent.

suggestedSelector
Answer a valid alternative to the selector that was used.

suggestedSelector: aSymbol
Set a valid alternative to the selector that was used.

1.199 TextCollector

Defined in namespace Smalltalk
Superclass: Stream
Category: Streams
I am a thread-safe class that maps between standard Stream protocol and a single message to another object (its selector is pluggable and should roughly correspond to #nextPutAll:). I am, in fact, the class that implements the global Transcript object.

1.199.1 TextCollector class: accessing

message: receiverToSelectorAssociation
Answer a new instance of the receiver, that uses the message identified by anAssociation to perform write operations. anAssociation's key is the receiver, while its value is the selector.

new This method should not be called for instances of this class.

1.199.2 TextCollector: accessing

cr Emit a new-line (carriage return) to the Transcript

critical: aBlock
Evaluate aBlock while holding the Transcript lock

endEntry Emit two new-lines. This method is present for compatibility with VisualWorks.

next: anInteger put: anObject
Write anInteger copies of anObject to the Transcript

next: n putAll: aString startingAt: pos
Write aString to the Transcript

nextPut: aCharacter
Emit aCharacter to the Transcript

show: aString
Write aString to the Transcript

showCr: aString
Write aString to the Transcript, followed by a new-line character

showOnNewLine: aString
Write aString to the Transcript, preceded by a new-line character

1.199.3 TextCollector: printing

print: anObject
Print anObject's representation to the Transcript

printOn: aStream
Print a representation of the receiver onto aStream

1.199.4 TextCollector: set up

message
Answer an association representing the message to be sent to perform write operations. The key is the receiver, the value is the selector

message: receiverToSelectorAssociation
Set the message to be sent to perform write operations to the one represented by anAssociation. anAssociation's key is the receiver, while its value is the selector

1.199.5 TextCollector: storing

store: anObject
Print Smalltalk code which evaluates to anObject on the Transcript

storeOn: aStream
Print Smalltalk code which evaluates to the receiver onto aStream

1.200 Time

Defined in namespace Smalltalk
Superclass: Magnitude
Category: Language-Data types
My instances represent times of the day. I provide methods for instance creation, methods that access components (hours, minutes, and seconds) of a time value, and a block execution timing facility.

1.200.1 Time class: basic (UTC)

midnight
Answer a time representing midnight in Coordinated Universal Time (UTC)

utcNow
Answer a time representing the current time of day in Coordinated Universal Time (UTC)

utcSecondClock
Answer the number of seconds since the midnight of 1/1/1901 (unlike #secondClock, the reference time is here expressed as UTC, that is as Coordinated Universal Time).

1.200.2 Time class: builtins

primNanosecondClock
Returns the number of milliseconds since midnight.

primSecondClock
Returns the number of seconds to/from 1/1/2000.

timezone Answer a String associated with the current timezone (either standard or daylight-saving) on this operating system. For example, the answer could be 'EST' to indicate Eastern Standard Time; the answer can be empty and can't be assumed to be a three-character code such as 'EST'.

timezoneBias
Specifies the current bias, in seconds, for local time translation for the current time. The bias is the difference, in seconds, between Coordinated Universal Time (UTC) and local time; a positive bias indicates that the local timezone is to the east of Greenwich (e.g. Europe, Asia), while a negative bias indicates that it is to the west (e.g. America)

timezoneBias: seconds
Specifies the bias, in seconds, for local time translation for the given second clock value (0 being midnight of 1/1/1901). The bias is the difference, in seconds, between Coordinated Universal Time (UTC) and local time; a positive bias indicates that the local timezone is to the east of Greenwich (e.g. Europe, Asia), while a negative bias indicates that it is to the west (e.g. America)

1.200.3 Time class: clocks

millisecondClock
Answer the number of milliseconds since startup.

millisecondClockValue
Answer the number of milliseconds since startup

millisecondsPerDay
Answer the number of milliseconds in a day

millisecondsToRun: timedBlock
Answer the number of milliseconds which timedBlock took to run

nanosecondClock
Answer the number of nanoseconds since startup.

nanosecondClockValue
Answer the number of milliseconds since startup

secondClock
Answer the number of seconds since the midnight of 1/1/1901

1.200.4 Time class: initialization

initialize Initialize the Time class after the image has been bootstrapped

update: aspect
Private - Initialize the receiver's instance variables

1.200.5 Time class: instance creation

fromSeconds: secondCount
Answer a Time representing secondCount seconds past midnight

hour: h Answer a Time that is the given number of hours past midnight

hour: h minute: m second: s
Answer a Time that is the given number of hours, minutes and seconds past midnight

hours: h Answer a Time that is the given number of hours past midnight

hours: h minutes: m seconds: s
Answer a Time that is the given number of hours, minutes and seconds past midnight

minute: m Answer a Time that is the given number of minutes past midnight

minutes: m
Answer a Time that is the given number of minutes past midnight

new Answer a Time representing midnight

now Answer a time representing the current time of day

readFrom: aStream
Parse an instance of the receiver (hours/minutes/seconds) from aStream

second: s Answer a Time that is the given number of seconds past midnight

seconds: s Answer a Time that is the given number of seconds past midnight

1.200.6 Time: accessing (ANSI for DateAndTimes)

hour Answer the number of hours in the receiver

hour12 Answer the hour in a 12-hour clock

hour24 Answer the hour in a 24-hour clock

minute Answer the number of minutes in the receiver

second Answer the number of seconds in the receiver

1.200.7 Time: accessing (non ANSI & for Durations)

asMilliseconds
Not commented.

asNanoseconds
Not commented.

asSeconds Answer ‘seconds’.

hours Answer the number of hours in the receiver

minutes Answer the number of minutes in the receiver

seconds Answer the number of seconds in the receiver

1.200.8 Time: arithmetic

addSeconds: timeAmount
Answer a new Time that is timeAmount seconds after the receiver

addTime: timeAmount
Answer a new Time that is timeAmount seconds after the receiver; timeAmount is a Time.

printOn: aStream
Print a representation of the receiver on aStream

subtractTime: timeAmount
Answer a new Time that is timeAmount seconds before the receiver; timeAmount is a Time.

1.200.9 Time: comparing

< aTime Answer whether the receiver is less than aTime

= aTime Answer whether the receiver is equal to aTime

hash Answer an hash value for the receiver

1.201 True

Defined in namespace Smalltalk
Superclass: Boolean
Category: Language-Data types
I represent truth and justice in the world. My motto is "semper veritatis".

1.201.1 True: basic

& aBoolean
We are true – anded with anything, we always answer the other operand

and: aBlock
We are true – anded with anything, we always answer the other operand, so evaluate aBlock

eqv: aBoolean
Answer whether the receiver and aBoolean represent the same boolean value

ifFalse: falseBlock
We are true – answer nil

ifFalse: falseBlock ifTrue: trueBlock
We are true – evaluate trueBlock

ifTrue: trueBlock
We are true – evaluate trueBlock

ifTrue: trueBlock ifFalse: falseBlock
We are true – evaluate trueBlock

not We are true – answer false

or: aBlock We are true – ored with anything, we always answer true

xor: aBoolean
Answer whether the receiver and aBoolean represent different boolean values

| aBoolean
We are true – ored with anything, we always answer true

1.201.2 True: C hacks

asCBooleanValue
Answer '1'.

1.201.3 True: printing

printOn: aStream
Print a representation of the receiver on aStream

1.202 UndefinedObject

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Implementation
I have the questionable distinction of being a class with only one instance, which is the object "nil".

1.202.1 UndefinedObject: basic

copy Answer the receiver.

deepCopy Answer the receiver.

shallowCopy
Answer the receiver.

1.202.2 UndefinedObject: class creation - alternative

subclass: classNameString classInstanceVariableNames: stringClassInstVarNames instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

subclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableByteSubclass: classNameString classInstanceVariableNames: stringClassInstVarNames instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableByteSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableLongSubclass: classNameString classInstanceVariableNames: stringClassInstVarNames instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableLongSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableSubclass: classNameString classInstanceVariableNames: stringClassInstVarNames instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

variableSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames
Don't use this, it is only present to file in from IBM Smalltalk

1.202.3 UndefinedObject: class polymorphism

allSubclasses
Return all the classes in the system.

instSize Answer '0'.

metaclassFor: classNameString
Create a Metaclass object for the given class name. The metaclass is a subclass of Class

methodDictionary
Answer 'nil'.

removeSubclass: aClass
Ignored – necessary to support disjoint class hierarchies

subclass: classNameString
Define a subclass of the receiver with the given name. If the class is already defined, don't modify its instance or class variables but still, if necessary, recompile everything needed.

subclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString
Define a fixed subclass of the receiver with the given name, instance variables, class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed.

variable: shape subclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString
Define a variable subclass of the receiver with the given name, shape, instance variables, class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed. The shape can be one of

#byte #int8 #character #short #ushort #int #uint #int64 #uint64 #utf32 #float #double or #pointer.

variableByteSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString
Define a byte variable subclass of the receiver with the given name, instance variables, class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed.

variableSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString
Define a variable pointer subclass of the receiver with the given name, instance variables, class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed.

variableWordSubclass: classNameString instanceVariableNames: stringInstVarNames classVariableNames: stringOfClassVarNames poolDictionaries: stringOfPoolNames category: categoryNameString
Define a word variable subclass of the receiver with the given name, instance variables, class variables, pool dictionaries and category. If the class is already defined, if necessary, recompile everything needed.

1.202.4 UndefinedObject: CObject interoperability

free Do nothing, a NULL pointer can be safely freed.

narrow Return the receiver: a NULL pointer is always nil, whatever its type.

1.202.5 UndefinedObject: dependents access

addDependent: ignored
Fail, nil does not support dependents.

release Ignore this call, nil does not support dependents.

1.202.6 UndefinedObject: iteration

ifNil: nilBlock ifNotNilDo: iterableBlock
Evaluate nilBlock if the receiver is nil, else evaluate iterableBlock with each element of the receiver (which should be an Iterable).

ifNotNilDo: iterableBlock
Evaluate iterableBlock with each element of the receiver (which should be an Iterable) if not nil. Else answer nil

ifNotNilDo: iterableBlock ifNil: nilBlock
Evaluate nilBlock if the receiver is nil, else evaluate iterableBlock, passing each element of the receiver (which should be an Iterable).

1.202.7 UndefinedObject: printing

printOn: aStream
Print a representation of the receiver on aStream.

printOn: aStream in: aNamespace
Print on aStream a representation of the receiver as it would be accessed from aNamespace: nil is the same everywhere, so print the same as #printOn:

1.202.8 UndefinedObject: still unclassified

inheritsFrom: aClass
Always return false, as nil inherits from nothing.

1.202.9 UndefinedObject: storing

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

storeLiteralOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

storeOn: aStream
Store Smalltalk code compiling to the receiver on aStream.

1.202.10 UndefinedObject: testing

ifNil: nilBlock
Evaluate nilBlock if the receiver is nil, else answer nil

ifNil: nilBlock ifNotNil: notNilBlock
Evaluate nilBlock if the receiver is nil, else evaluate notNilBlock, passing the receiver.

ifNotNil: notNilBlock
Evaluate notNilBlock if the receiver is not nil, passing the receiver. Else answer nil

ifNotNil: notNilBlock ifNil: nilBlock
Evaluate nilBlock if the receiver is nil, else evaluate notNilBlock, passing the receiver.

isNil Answer whether the receiver is the undefined object nil. Always answer true.

isNull Answer whether the receiver represents a NULL C pointer. Always answer true.

notNil Answer whether the receiver is not the undefined object nil. Always answer false.

1.203 UnicodeCharacter

Defined in namespace Smalltalk
Superclass: Character
Category: Language-Data types
My instances represent the over one million characters of the Unicode character set. It provides messages to translate between integers and character objects.

UnicodeCharacter objects are created when accessing UnicodeStrings, or with Character class>>#codePoint:.

1.203.1 UnicodeCharacter class: built ins

value: anInteger
Returns the character object, possibly a Character, corresponding to anInteger. Error if anInteger is not an integer, or not in 0..16r10FFFF.

This is only a primitive for speed. UnicodeCharacter's #value: method is equivalent to #codePoint: (which is the same for Character and UnicodeCharacter).

1.203.2 UnicodeCharacter: coercion methods

*** aNumber**
Returns a String with aNumber occurrences of the receiver.

1.204 UnicodeString

Defined in namespace Smalltalk
Superclass: CharacterArray
Category: Collections-Text
My instances represent Unicode string data types. Data is stored as 4-byte UTF-32 characters

1.204.1 UnicodeString class: converting

fromString: aString
Return the String, aString, converted to its Unicode representation. Unless the I18N package is loaded, this is not implemented.

1.204.2 UnicodeString class: multibyte encodings

defaultEncoding
Answer the encoding used by the receiver. Conventionally, we answer 'Unicode' to ensure that two UnicodeStrings always have the same encoding.

isUnicode Answer true; the receiver stores characters.

1.204.3 UnicodeString: built ins

at: anIndex ifAbsent: aBlock
Answer the index-th indexed instance variable of the receiver

1.204.4 UnicodeString: built-ins

hash Answer an hash value for the receiver

1.204.5 UnicodeString: converting

asString Returns the string corresponding to the receiver. Without the Iconv package, unrecognized Unicode characters become $? characters. When it is loaded, an appropriate single- or multi-byte encoding could be used.

asSymbol Returns the symbol corresponding to the receiver

asUnicodeString
But I already am a UnicodeString! Really!

displayOn: aStream
Print a representation of the receiver on aStream

printOn: aStream
Print a representation of the receiver on aStream

1.204.6 UnicodeString: multibyte encodings

encoding Answer the encoding used by the receiver. Conventionally, we answer 'Unicode' to ensure that two UnicodeStrings always have the same encoding.

numberOfCharacters
Answer the number of Unicode characters in the receiver. This is the same as #size for UnicodeString.

1.205 ValueAdaptor

Defined in namespace Smalltalk
Superclass: Object
Category: Language-Data types
My subclasses are used to access data from different objects with a consistent protocol. However, I'm an abstract class.

1.205.1 ValueAdaptor class: creating instances

new We don't know enough of subclasses to have a shared implementation of new

1.205.2 ValueAdaptor: accessing

value Retrive the value of the receiver. Must be implemented by ValueAdaptor's subclasses

value: anObject
Set the value of the receiver. Must be implemented by ValueAdaptor's subclasses

1.205.3 ValueAdaptor: printing

printOn: aStream
Print a representation of the receiver

1.206 ValueHolder

Defined in namespace Smalltalk
Superclass: ValueAdaptor
Category: Language-Data types
I store my value in a variable. For example, you can use me to pass numbers by reference. Just instance me before calling a method and ask for my value after that method. There are a lot of other creative uses for my intances, though.

1.206.1 ValueHolder class: creating instances

new Create a ValueHolder whose starting value is nil

null Answer the sole instance of NullValueHolder

with: anObject
Create a ValueHolder whose starting value is anObject

1.206.2 ValueHolder: accessing

value Get the value of the receiver.

value: anObject
Set the value of the receiver.

1.206.3 ValueHolder: initializing

initialize Private - set the initial value of the receiver

1.207 VariableBinding

Defined in namespace Smalltalk
Superclass: HomedAssociation
Category: Language-Data types
My instances represent a mapping between a key in a namespace and its value. I print different than a normal Association, and know about my parent namespace, otherwise my behavior is the same.

1.207.1 VariableBinding: compiler

literalEquals: anObject
Not commented.

literalHash
Not commented.

1.207.2 VariableBinding: printing

path Print a dotted path that compiles to the receiver's value

printOn: aStream
Put on aStream a representation of the receiver

1.207.3 VariableBinding: saving and loading

binaryRepresentationObject
This method is implemented to allow for a PluggableProxy to be used with VariableBindings. Answer a DirectedMessage which sends #at: to the environment that holds the receiver.

1.207.4 VariableBinding: storing

isLiteralObject
Answer whether the receiver is expressible as a Smalltalk literal.

storeLiteralOn: aStream
Store on aStream some Smalltalk code which compiles to the receiver

storeOn: aStream
Put on aStream some Smalltalk code compiling to the receiver

1.207.5 VariableBinding: testing

isDefined Answer true if this VariableBinding lives outside the Undeclared dictionary

1.208 VersionableObjectProxy

Defined in namespace Smalltalk
Superclass: NullProxy
Category: Streams-Files
I am a proxy that stores additional information to allow different versions of an object's representations to be handled by the program. VersionableObjectProxies are backwards compatible, that is you can support versioning even if you did not use a VersionableObjectProxy for that class when the object was originarily dumped. VersionableObjectProxy does not support classes that changed shape across different versions. See the method comments for more information.

1.208.1 VersionableObjectProxy class: saving and restoring

loadFrom: anObjectDumper
Retrieve the object. If the version number doesn't match the #binaryRepresentationVersion answered by the class, call the class' #convertFromVersion:-withFixedVariables:instanceVariables:for: method. The stored version number will be the first parameter to that method (or nil if the stored object did not employ a VersionableObjectProxy), the remaining parameters will be respectively the fixed instance variables, the indexed instance variables (or nil if the class is fixed), and the ObjectDumper itself. If no VersionableObjectProxy, the class is sent #nonVersionedInstSize to retrieve the number of fixed instance variables stored for the non-versioned object.

1.208.2 VersionableObjectProxy: saving and restoring

dumpTo: anObjectDumper
Save the object with extra versioning information.

1.209 VFS.ArchiveFile

Defined in namespace Smalltalk.VFS
Superclass: VFS.FileWrapper
Category: Streams-Files
ArchiveFile handles virtual filesystems that have a directory structure of their own. The directories and files in the archive are instances of ArchiveMember,

but the functionality resides entirely in ArchiveFile because the members will still ask the archive to get directory information on them, to extract them to a real file, and so on.

1.209.1 VFS.ArchiveFile: ArchiveMember protocol

fillMember: anArchiveMember
Extract the information on anArchiveMember. Answer false if it actually does not exist in the archive; otherwise, answer true after having told anArchiveMember about them by sending #size:stCtime:stMtime:stAtime:isDirectory: to it.

member: anArchiveMember do: aBlock
Evaluate aBlock once for each file in the directory represented by anArchiveMember, passing its name.

member: anArchiveMember mode: bits
Set the permission bits for the file in anArchiveMember.

refresh Extract the directory listing from the archive

removeMember: anArchiveMember
Remove the member represented by anArchiveMember.

updateMember: anArchiveMember
Update the member represented by anArchiveMember by copying the file into which it was extracted back to the archive.

1.209.2 VFS.ArchiveFile: directory operations

at: aName
Answer a FilePath for a file named 'aName' residing in the directory represented by the receiver.

nameAt: aString
Answer a FilePath for a file named 'aName' residing in the directory represented by the receiver.

namesDo: aBlock
Evaluate aBlock once for each file in the directory represented by the receiver, passing its name.

release Release the resources used by the receiver that don't survive when reloading a snapshot.

1.209.3 VFS.ArchiveFile: querying

isAccessible
Answer whether a directory with the name contained in the receiver does exist and can be accessed

isDirectory
Answer true. The archive can always be considered as a directory.

1.209.4 VFS.ArchiveFile: still unclassified

displayOn: aStream
Print a representation of the file identified by the receiver.

1.209.5 VFS.ArchiveFile: TmpFileArchiveMember protocol

extractMember: anArchiveMember
Extract the contents of anArchiveMember into a file that resides on disk, and answer the name of the file.

extractMember: anArchiveMember into: file
Extract the contents of anArchiveMember into a file that resides on disk, and answer the name of the file.

1.210 VFS.ArchiveMember

Defined in namespace Smalltalk.VFS
Superclass: FilePath
Category: Streams-Files
TmpFileArchiveMember is a handler class for members of archive files that creates temporary files when extracting files from an archive.

1.210.1 VFS.ArchiveMember: accessing

archive Answer the archive of which the receiver is a member.

asString Answer the name of the file identified by the receiver as answered by File>>-#name.

creationTime
Answer the creation time of the file identified by the receiver. On some operating systems, this could actually be the last change time (the ‘last change time’ has to do with permissions, ownership and the like).

lastAccessTime
Answer the last access time of the file identified by the receiver

lastChangeTime
Answer the last change time of the file identified by the receiver (the ‘last change time’ has to do with permissions, ownership and the like). On some operating systems, this could actually be the file creation time.

lastModifyTime
Answer the last modify time of the file identified by the receiver (the ‘last modify time’ has to do with the actual file contents).

name Answer the receiver’s file name.

name: aName
Set the receiver’s file name to aName.

refresh Refresh the statistics for the receiver

size Answer the size of the file identified by the receiver

1.210.2 VFS.ArchiveMember: basic

= aFile Answer whether the receiver represents the same file as the receiver.

hash Answer a hash value for the receiver.

1.210.3 VFS.ArchiveMember: delegation

full Answer the size of the file identified by the receiver

1.210.4 VFS.ArchiveMember: directory operations

at: aName
Answer a FilePath for a file named 'aName' residing in the directory represented by the receiver.

createDirectory: dirName
Create a subdirectory of the receiver, naming it dirName.

namesDo: aBlock
Evaluate aBlock once for each file in the directory represented by the receiver, passing its name.

1.210.5 VFS.ArchiveMember: file operations

open: class mode: mode ifFail: aBlock
Open the receiver in the given mode (as answered by FileStream's class constant methods)

remove Remove the file with the given path name

renameTo: newFileName
Rename the file with the given path name oldFileName to newFileName

update: aspect
Private - Update the in-archive version of the file before closing.

1.210.6 VFS.ArchiveMember: initializing

archive: anArchiveFile
Set the archive of which the receiver is a member.

fillFrom: data
Called back by the receiver's archive when the ArchiveMember asks for file information.

size: bytes stCtime: ctime stMtime: mtime stAtime: atime mode: modeBits
Set the file information for the receiver.

size: bytes stMtime: mtime mode: modeBits
Set the file information for the receiver.

1.210.7 VFS.ArchiveMember: still unclassified

, aName Answer an object of the same kind as the receiver, whose name is suffixed with aName.

displayOn: aStream
Print a representation of the file identified by the receiver.

isAbsolute Answer whether the receiver identifies an absolute path.

1.210.8 VFS.ArchiveMember: testing

exists Answer whether a file with the name contained in the receiver does exist.

isAccessible
Answer whether a directory with the name contained in the receiver does exist and is accessible

isDirectory
Answer whether a file with the name contained in the receiver does exist and identifies a directory.

isExecutable
Answer whether a file with the name contained in the receiver does exist and is executable

isReadable
Answer whether a file with the name contained in the receiver does exist and is readable

isSymbolicLink
Answer whether a file with the name contained in the receiver does exist and identifies a symbolic link.

isWriteable
Answer whether a file with the name contained in the receiver does exist and is writeable

mode Answer the octal permissions for the file.

mode: mode
Set the octal permissions for the file to be 'mode'.

1.211 VFS.FileWrapper

Defined in namespace Smalltalk.VFS
Superclass: FilePath
Category: Streams-Files
FileWrapper gives information for virtual files that refer to a real file on disk.

1.211.1 VFS.FileWrapper class: initializing

initialize Register the receiver with ObjectMemory

update: aspect
Private - Remove the files before quitting, and register the virtual filesystems specified by the subclasses upon image load.

1.211.2 VFS.FileWrapper class: instance creation

on: file Create an instance of this class representing the contents of the given file, under the virtual filesystem fsName.

1.211.3 VFS.FileWrapper: accessing

asString Answer the string representation of the receiver's path.

at: aName
Answer a File or Directory object as appropriate for a file named 'aName' in the directory represented by the receiver.

lastAccessTime: accessDateTime lastModifyTime: modifyDateTime
Update the timestamps of the file corresponding to the receiver, to be accessDateTime and modifyDateTime.

name Answer the full path to the receiver.

owner: ownerString group: groupString
Set the receiver's owner and group to be ownerString and groupString.

pathTo: destName
Compute the relative path from the receiver to destName.

1.211.4 VFS.FileWrapper: basic

= aFile Answer whether the receiver represents the same file as the receiver.

hash Answer a hash value for the receiver.

1.211.5 VFS.FileWrapper: delegation

creationTime
Answer the creation time of the file identified by the receiver. On some operating systems, this could actually be the last change time (the 'last change time' has to do with permissions, ownership and the like).

full Answer the size of the file identified by the receiver

isExecutable
Answer whether a file with the name contained in the receiver does exist and is executable

isReadable
Answer whether a file with the name contained in the receiver does exist and is readable

isWriteable
Answer whether a file with the name contained in the receiver does exist and is writeable

lastAccessTime
Answer the last access time of the file identified by the receiver

lastChangeTime
Answer the last change time of the file identified by the receiver (the 'last change time' has to do with permissions, ownership and the like). On some operating systems, this could actually be the file creation time.

lastModifyTime
Answer the last modify time of the file identified by the receiver (the 'last modify time' has to do with the actual file contents).

mode Answer the permission bits for the file identified by the receiver

mode: anInteger
Answer the permission bits for the file identified by the receiver

open: class mode: mode ifFail: aBlock
Open the receiver in the given mode (as answered by FileStream's class constant methods)

remove Remove the file with the given path name

size Answer the size of the file identified by the receiver

1.211.6 VFS.FileWrapper: enumerating

namesDo: aBlock
Evaluate aBlock once for each file in the directory represented by the receiver, passing its name.

1.211.7 VFS.FileWrapper: file operations

pathFrom: dirName
Compute the relative path from the directory dirName to the receiver

renameTo: newName
Rename the file identified by the receiver to newName

symlinkAs: destName
Create destName as a symbolic link of the receiver. The appropriate relative path is computed automatically.

symlinkFrom: srcName
Create the receiver as a symbolic link from srcName (relative to the path of the receiver).

1.211.8 VFS.FileWrapper: testing

exists Answer whether a file with the name contained in the receiver does exist.

isAbsolute Answer whether the receiver identifies an absolute path.

isAccessible
Answer whether a directory with the name contained in the receiver does exist and can be accessed

isDirectory
Answer whether a file with the name contained in the receiver does exist identifies a directory.

isSymbolicLink
: Answer whether a file with the name contained in the receiver does exist and identifies a symbolic link.

1.212 VFS.StoredZipMember

Defined in namespace Smalltalk.VFS
Superclass: VFS.TmpFileArchiveMember
Category: Streams-Files
: ArchiveMember is the handler class for stored ZIP archive members, which are optimized.

1.212.1 VFS.StoredZipMember: accessing

offset
: Answer 'offset'.

offset: anInteger
: Not commented.

1.212.2 VFS.StoredZipMember: opening

open: class mode: mode ifFail: aBlock
: Not commented.

1.213 VFS.TmpFileArchiveMember

Defined in namespace Smalltalk.VFS
Superclass: VFS.ArchiveMember
Category: Streams-Files

1.213.1 VFS.TmpFileArchiveMember: directory operations

file
: Answer the real file name which holds the file contents, or nil if it does not apply.

open: class mode: mode ifFail: aBlock
: Open the receiver in the given mode (as answered by FileStream's class constant methods)

1.213.2 VFS.TmpFileArchiveMember: finalization

release
: Release the resources used by the receiver that don't survive when reloading a snapshot.

1.213.3 VFS.TmpFileArchiveMember: still unclassified

extracted
: Answer whether the file has already been extracted to disk.

1.214 VFS.ZipFile

Defined in namespace Smalltalk.VFS
Superclass: VFS.ArchiveFile
Category: Streams-Files

ZipFile transparently extracts files from a ZIP archive.

1.214.1 VFS.ZipFile: members

centralDirectoryRangeIn: f
Not commented.

createDirectory: dirName
Create a subdirectory of the receiver, naming it dirName.

extractMember: anArchiveMember into: temp
Extract the contents of anArchiveMember into a file that resides on disk, and answer the name of the file.

fileData
Extract the directory listing from the archive

member: anArchiveMember mode: bits
Set the permission bits for the file in anArchiveMember.

removeMember: anArchiveMember
Remove the member represented by anArchiveMember.

updateMember: anArchiveMember
Update the member represented by anArchiveMember by copying the file into which it was extracted back to the archive.

1.215 Warning

Defined in namespace Smalltalk
Superclass: Notification
Category: Language-Exceptions

Warning represents an ‘important’ but resumable error.

1.215.1 Warning: exception description

description
Answer a textual description of the exception.

1.216 WeakArray

Defined in namespace Smalltalk
Superclass: Array
Category: Collections-Weak

I am similar to a plain array, but my items are stored in a weak object, so I track which of them are garbage collected.

1.216.1 WeakArray class: instance creation

new Create a new WeakArray of size 0.

new: size Create a new WeakArray of the given size.

1.216.2 WeakArray: accessing

aliveObjectsDo: aBlock
Evaluate aBlock for all the elements in the array, excluding the garbage collected ones. Note: a finalized object stays alive until the next collection (the collector has no means to see whether it was resuscitated by the finalizer), so an object being alive does not mean that it is usable.

at: index Answer the index-th item of the receiver, or nil if it has been garbage collected.

at: index put: object
Store the value associated to the given index; plus, store in nilValues whether the object is nil. nil objects whose associated item of nilValues is 1 were touched by the garbage collector.

atAll: indices put: object
Put object at every index contained in the indices collection

atAllPut: object
Put object at every index in the receiver

clearGCFlag: index
Clear the 'object has been garbage collected' flag for the item at the given index

do: aBlock
Evaluate aBlock for all the elements in the array, including the garbage collected ones (pass nil for those).

isAlive: index
Answer whether the item at the given index is still alive or has been garbage collected. Note: a finalized object stays alive until the next collection (the collector has no means to see whether it was resuscitated by the finalizer), so an object being alive does not mean that it is usable.

size Answer the number of items in the receiver

1.216.3 WeakArray: conversion

asArray Answer a non-weak version of the receiver

deepCopy Returns a deep copy of the receiver (the instance variables are copies of the receiver's instance variables)

shallowCopy
Returns a shallow copy of the receiver (the instance variables are not copied)

species Answer Array; this method is used in the #copyEmpty: message, which in turn is used by all collection-returning methods (collect:, select:, reject:, etc.).

1.216.4 WeakArray: loading

postLoad Called after loading an object; must restore it to the state before 'preStore' was called. Make it weak again

1.217 WeakIdentitySet

Defined in namespace Smalltalk
Superclass: WeakSet
Category: Collections-Weak

I am similar to a plain identity set, but my keys are stored in a weak array; I track which of them are garbage collected and, as soon as I encounter one of them, I swiftly remove all the garbage collected keys

1.217.1 WeakIdentitySet: accessing

identityIncludes: anObject
Answer whether I include anObject exactly. As I am an identity-set, this is the same as #includes:.

1.218 WeakKeyDictionary

Defined in namespace Smalltalk
Superclass: Dictionary
Category: Collections-Weak

I am similar to a plain Dictionary, but my keys are stored in a weak array; I track which of them are garbage collected and, as soon as I encounter one of them, I swiftly remove all the associations for the garbage collected keys

1.218.1 WeakKeyDictionary class: hacks

postLoad Called after loading an object; must restore it to the state before 'preStore' was called. Make it weak again

1.218.2 WeakKeyDictionary: accessing

add: anAssociation
Store value as associated to the given key.

at: key put: value
Store value as associated to the given key.

1.219 WeakKeyIdentityDictionary

Defined in namespace Smalltalk
Superclass: WeakKeyDictionary
Category: Collections-Weak

I am similar to a plain identity dictionary, but my keys are stored in a weak array; I track which of them are garbage collected and, as soon as I encounter one of them, I swiftly remove all the associations for the garbage collected keys

1.220 WeakSet

Defined in namespace Smalltalk
Superclass: Set
Category: Collections-Weak

I am similar to a plain set, but my items are stored in a weak array; I track which of them are garbage collected and, as soon as I encounter one of them, I swiftly remove all.

1.220.1 WeakSet: accessing

add: newObject
Add newObject to the set, if and only if the set doesn't already contain an occurrence of it. Don't fail if a duplicate is found. Answer newObject

do: aBlock
Enumerate all the non-nil members of the set

1.220.2 WeakSet: copying

deepCopy Returns a deep copy of the receiver (the instance variables are copies of the receiver's instance variables)

shallowCopy
Returns a shallow copy of the receiver (the instance variables are not copied)

1.220.3 WeakSet: loading

postLoad Called after loading an object; must restore it to the state before 'preStore' was called. Make it weak again

1.221 WeakValueIdentityDictionary

Defined in namespace Smalltalk
Superclass: WeakValueLookupTable
Category: Collections-Weak

I am similar to a plain identity dictionary, but my values are stored in a weak array; I track which of the values are garbage collected and, as soon as one of them is accessed, I swiftly remove the associations for the garbage collected values

1.222 WeakValueLookupTable

Defined in namespace Smalltalk
Superclass: LookupTable
Category: Collections-Weak

I am similar to a plain LookupTable, but my values are stored in a weak array; I track which of the values are garbage collected and, as soon as one of them is accessed, I swiftly remove the associations for the garbage collected values

1.222.1 WeakValueLookupTable class: hacks

primNew: realSize
Answer a new, uninitialized instance of the receiver with the given size

1.222.2 WeakValueLookupTable: hacks

at: key ifAbsent: aBlock
Answer the value associated to the given key, or the result of evaluating aBlock if the key is not found

at: key ifPresent: aBlock
If aKey is absent, answer nil. Else, evaluate aBlock passing the associated value and answer the result of the invocation

includesKey: key
Answer whether the receiver contains the given key.

1.222.3 WeakValueLookupTable: rehashing

rehash Rehash the receiver

1.223 WordArray

Defined in namespace Smalltalk
Superclass: ArrayedCollection
Category: Collections-Sequenceable
I am similar to a plain array, but my items are 32-bit integers.

1.223.1 WordArray: built ins

at: anIndex ifAbsent: aBlock
Answer the index-th indexed instance variable of the receiver

1.224 WriteStream

Defined in namespace Smalltalk
Superclass: PositionableStream
Category: Streams-Collections
I am the class of writeable streams. I only allow write operations to my instances; reading is strictly forbidden.

1.224.1 WriteStream class: instance creation

on: aCollection
Answer a new instance of the receiver which streams on aCollection. Every item of aCollection is discarded.

with: aCollection
Answer a new instance of the receiver which streams from the end of aCollection.

with: aCollection from: firstIndex to: lastIndex
Answer a new instance of the receiver which streams from the firstIndex-th item of aCollection to the lastIndex-th. The pointer is moved to the last item in that range.

1.224.2 WriteStream: accessing-writing

contents Returns a collection of the same type that the stream accesses, up to and including the final element.

next: n putAll: aCollection startingAt: pos
Put n characters or bytes of aCollection, starting at the pos-th, in the collection buffer.

nextPut: anObject
Store anObject as the next item in the receiver. Grow the collection if necessary

readStream
Answer a ReadStream on the same contents as the receiver

reverseContents
Returns a collection of the same type that the stream accesses, up to and including the final element, but in reverse order.

1.224.3 WriteStream: positioning

emptyStream
Extension - Reset the stream

1.225 ZeroDivide

Defined in namespace Smalltalk
Superclass: ArithmeticError
Category: Language-Exceptions
A ZeroDivide exception is raised by numeric classes when a program tries to divide by zero. Information on the dividend is available to the handler.

1.225.1 ZeroDivide class: instance creation

dividend: aNumber
Create a new ZeroDivide object remembering that the dividend was aNumber.

new Create a new ZeroDivide object; the dividend is conventionally set to zero.

1.225.2 ZeroDivide: accessing

dividend Answer the number that was being divided by zero

1.225.3 ZeroDivide: description

description
Answer a textual description of the exception.

Class index

Method index

Selector cross-reference

Table of Contents