

# Package ‘R.oo’

November 2, 2009

**Version** 1.6.5

**Date** 2009-10-30

**Title** R object-oriented programming with or without references

**Author** Henrik Bengtsson <henrikb@braju.com>

**Maintainer** Henrik Bengtsson <henrikb@braju.com>

**Depends** R (>= 2.2.0), R.methodsS3, utils

**Suggests** tools

**Imports** methods

**Description** Methods and classes for object-oriented programming in R with or without references. Large effort has been made on making definition of methods as simple as possible with a minimum of maintenance for package developers. The package has been developed since 2001 and is now considered very stable. This is a cross-platform package implemented in pure R that defines standard S3 classes without any tricks.

**License** LGPL (>= 2.1)

**URL** <http://www.braju.com/R/>

**DevelURL** <http://www.braju.com/R/>

**LazyLoad** TRUE

**Repository** CRAN

**Date/Publication** 2009-11-02 07:46:28

**R topics documented:**

R.oo-package . . . . .	2
charToInt . . . . .	4
Class . . . . .	5
dimension . . . . .	6
equals . . . . .	7
Exception . . . . .	8
extend . . . . .	11
getConstructorS3 . . . . .	13
getName.environment . . . . .	13
hashCode . . . . .	14
InternalErrorException . . . . .	15
intToChar . . . . .	16
ll . . . . .	17
Object . . . . .	19
objectSize . . . . .	24
objectSize.environment . . . . .	25
Package . . . . .	25
RccViolationException . . . . .	28
Rdoc . . . . .	30
RdocException . . . . .	31
setConstructorS3 . . . . .	33
throw . . . . .	35
throw.error . . . . .	36
trim . . . . .	36
typeofClass . . . . .	37
<b>Index</b>	<b>38</b>

---

R.oo-package

*Package R.oo*


---

**Description**

Methods and classes for object-oriented programming in R with or without references. Large effort has been made on making definition of methods as simple as possible with a minimum of maintenance for package developers. The package has been developed since 2001 and is now considered very stable. This is a cross-platform package implemented in pure R that defines standard S3 classes without any tricks.

Please note that the Rdoc syntax/grammar used to convert Rdoc comments in code into Rd files is not strictly defined and is modified by the need of the author. Ideally, there will be a well defined Rdoc language one day.

## Installation and updates

To install this package do

```
install.packages("R.oo")
```

To get the "devel" version, see <http://www.braju.com/R/>.

## Dependencies and other requirements

This package requires a standard R installation and the **R.methodsS3** package.

## To get started

To get started, see:

1. [Object](#) - Root class providing support for reference variables. Any class inheriting from this class supports reference variables.

## Further readings

For a detailed introduction to the package see [1]. To define static fields, see help on [Object](#).

## How to cite this package

Whenever using this package, please cite [1] as

```
@INPROCEEDINGS{BengtssonH_2003,  
  author      = {Henrik Bengtsson},  
  title       = {The {R.oo} package - Object-Oriented Programming  
                with References Using Standard {R} Code},  
  booktitle   = {Proceedings of the 3rd International Workshop on  
                Distributed Statistical Computing (DSC 2003)},  
  year        = {2003},  
  editor      = {Kurt Hornik and Friedrich Leisch and Achim Zeileis},  
  address     = {Vienna, Austria},  
  month       = {March},  
  issn        = {1609-395X},  
  howpublished = {http://www.ci.tuwien.ac.at/Conferences/DSC-2003/},  
}
```

## License

The releases of this package is licensed under LGPL version 2.1 or newer.

## Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

## References

[1] H. Bengtsson, *The R.oo package - Object-Oriented Programming with References Using Standard R Code*, In Kurt Hornik, Friedrich Leisch and Achim Zeileis, editors, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20-22, Vienna, Austria. <http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/>

---

charToInt

*Converts a vector of ASCII characters into a vector of integers*

---

## Description

Converts a `vector` of ASCII `characters` to a equal length vector of ASCII `integers`.

## Usage

```
## Default S3 method:  
charToInt(ch, ...)
```

## Arguments

ch	A <code>character vector</code> .
...	Not used.

## Value

Returns an ASCII `character vector`.

## Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

## See Also

`utf8Conversion.intToChar()`

## Examples

```
i <- charToInt(unlist(strsplit("Hello world!", split=NULL)))  
# Gives: 72 101 108 108 111 32 119 111 114 108 100 33  
ch <- intToChar(c(72,101,108,108,111,32,119,111,114,108,100,33))  
# Gives: "H" "e" "l" "l" "o" " " "w" "o" "r" "l" "d" "!"
```



<code>argsToString</code>	Gets the arguments of a function as a character string.
<code>as.character</code>	Returns a short string describing the class.
<code>forName</code>	Gets a Class object by a name of a class.
<code>getDetails</code>	Lists the fields and methods of a class.
<code>getFields</code>	Returns the field names of a class.
<code>getKnownSubclasses</code>	Gets all subclasses that are currently loaded.
<code>getMethods</code>	Returns the method names of class and its super classes.
<code>getName</code>	Gets the name of the class.
<code>getPackage</code>	Gets the package to which the class belongs.
<code>getRdDeclaration</code>	Gets the class declaration in Rd format.
<code>getRdHierarchy</code>	Gets the class hierarchy in Rd format.
<code>getRdMethods</code>	Gets the methods of a class in Rd format.
<code>getStaticInstance</code>	Gets the static instance of this class.
<code>getSuperclasses</code>	Gets the super classes of this class.
<code>isAbstract</code>	Checks if a class is abstract or not.
<code>isBeingCreated</code>	Checks if a class is currently being initiated.
<code>isDeprecated</code>	Checks if a class is deprecated or not.
<code>isPrivate</code>	Checks if a class is defined private or not.
<code>isProtected</code>	Checks if a class is defined protected or not.
<code>isPublic</code>	Checks if a class is defined public or not.
<code>isStatic</code>	Checks if a class is static or not.
<code>newInstance</code>	Creates a new instance of this class.
<code>print</code>	Prints detailed information about the class and its fields and methods.

#### Methods inherited from Object:

,<, [[, [[<, as.character, attach, attachLocally, clearCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

#### Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

---

dimension

*Gets the dimension of the object*

---

#### Description

Gets the dimension of the object similar to what `dim()` does, but instead of `NULL` it will return the length of a vector. If a function is passed, `NULL` is returned.

#### Usage

```
## Default S3 method:
dimension(object, ...)
```

**Arguments**

object            The object for which the dimension should be obtained.  
 ...                Not used.

**Value**

Returns an `integer vector` or `NULL`.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

`ll.default().dim()` and `length()`.

**Examples**

```
dimension(matrix(1:100, ncol=10))    # 10 10
dimension(1:14)                      # 14
dimension(data.frame(a=1:10, b=10:1)) # 10 2
dimension(print)                      # NULL
```

---

equals

*Compares an object with another*

---

**Description**

Compares an object with another and returns `TRUE` if they are equal. The equal property must be

1) *reflexive*, i.e. `equals(o1, o1)` should be `TRUE`.

2) *symmetric*, i.e. `equals(o1, o2)` is `TRUE` if and only if `equals(o2, o1)` is `TRUE`.

3) *transitive*, i.e. `equals(o1, o2)` is `TRUE` and `equals(o2, o3)` is `TRUE`, then `equals(o1, o3)` should be `TRUE`.

5) *consistent*, i.e. `equals(o1, o2)` should return the same result on multiple invocations as long as nothing has changed.

6) `equals(o1, NULL)` should return `FALSE`.

By default `identical()` is used.

**Usage**

```
## Default S3 method:
equals(object, other, ...)
```

**Arguments**

`object, other`  
 Objects to be compared.  
`...` Not used.

**Value**

Returns `TRUE` if the objects are equal, otherwise `FALSE`.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

`identical()`.

---

Exception

*The Exception class to be thrown and caught*

---

**Description**

Package: R.oo

**Class Exception****Object**

```

~~|
~~+--try-error
~~~~~|
~~~~~+--condition
~~~~~|
~~~~~+--error
~~~~~|
~~~~~+--simpleError
~~~~~|
~~~~~+--Exception

```

**Directly known subclasses:**

[InternalErrorException](#), [RccViolationException](#), [RdocException](#)

```

public static class Exception
  extends simpleError

```

Creates an Exception that can be thrown and caught. The `Exception` class is the root class of all other Exception classes.

**Usage**

```
Exception(..., sep="", collapse="", )
```

**Arguments**

...	One or several strings, which will be concatenated and contain informative message about the exception.
sep	The string to used for concatenating several strings.
collapse	The string to used collapse vectors together.

**Fields and Methods****Methods:**

<code>as.character</code>	Gets a character string representing of the Exception.
<code>getCall</code>	-
<code>getLastException</code>	Static method to get the last Exception thrown.
<code>getMessage</code>	Gets the message of the Exception.
<code>getStackTrace</code>	Gets the stack trace saved when the exception was created.
<code>getStackTraceString</code>	Gets the stack trace as a string.
<code>getWhen</code>	Gets the time when the Exception was created.
<code>print</code>	Prints the Exception.
<code>printStackTrace</code>	Prints the stack trace saved when the exception was created.
<code>throw</code>	Throws an Exception that can be caught.

**Methods inherited from error:**

as.character, throw

**Methods inherited from condition:**

as.character, conditionCall, conditionMessage, print

**Methods inherited from Object:**

,<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

See also `tryCatch()` (and `try()`).

**Examples**

```
#####
# 1. To catch a regular "error" exception thrown by e.g. stop().
#####
```

```

x <- NA
y <- NA
tryCatch({
  x <- log(123);
  y <- log("a");
}, error = function(ex) {
  print(ex);
})
print(x)
print(y)

#####
# 2. Always run a "final" expression regardless or error or not.
#####
filename <- tempfile("R.methodsS3.example")
con <- file(filename)
tryCatch({
  open(con, "r");
}, error = function(ex) {
  cat("Could not open ", filename, " for reading.\n", sep="")
}, finally = {
  close(con)
  cat("The id of the connection is ",
      ifelse(is.null(con), "NULL", con), ".\n", sep="")
})

#####
# 3. Creating your own Exception class
#####
setConstructorS3("NegativeLogValueException", function(
  msg="Trying to calculate the logarithm of a negative value", value=NULL) {
  extend(Exception(msg=msg), "NegativeLogValueException",
    .value = value
  )
})

setMethodS3("as.character", "NegativeLogValueException", function(this, ...) {
  paste(as.character.Exception(this), ": ", getValue(this), sep="");
})

setMethodS3("getValue", "NegativeLogValueException", function(this, ...) {
  this$.value;
})

mylog <- function(x, base=exp(1)) {
  if (x < 0)
    throw(NegativeLogValueException(value=x))
  else
    log(x, base=base)
}

# Note that the order of the catch list is important:

```

```

l <- NA;
x <- 123;
tryCatch({
  l <- mylog(x);
}, NegativeLogValueException = function(ex) {
  cat(as.character(ex), "\n")
}, "try-error" = function(ex) {
  cat("try-error: Could not calculate the logarithm of ", x, ".\n", sep="")
}, error = function(ex) {
  cat("error: Could not calculate the logarithm of ", x, ".\n", sep="")
})
cat("The logarithm of ", x, " is ", l, ".\n\n", sep="")

```

---

 extend

*Extends a object*


---

### Description

Simply speaking this method "extends" the class of an object. What is actually happening is that it creates an instance of class name `...className`, by taking another object and add `...className` to the class list and also add all the named values in `...` as attributes.

The method should be used by the constructor of a class and nowhere else.

### Usage

```

## Default S3 method:
extend(this, ...className, ...)

```

### Arguments

```

this          Object to be extended.
...className  The name of new class.
...          Attribute fields of the new class.

```

### Value

Returns an object of class `...className`.

### Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

**Examples**

```

setConstructorS3("MyDouble", function(value=0, ...) {
  extend(as.double(value), "MyDouble", ...)
})

setMethodS3("as.character", "MyDouble", function(object, ...) {
  fmtstr <- attr(object, "fmtstr")
  if (is.null(fmtstr))
    fmtstr <- "%.6f"
  sprintf(fmtstr, object)
})

setMethodS3("print", "MyDouble", function(object, ...) {
  print(as.character(object), ...)
})

x <- MyDouble(3.1415926)
print(x)

x <- MyDouble(3.1415926, fmtstr="%3.2f")
print(x)
attr(x, "fmtstr") <- "%e"
print(x)

setConstructorS3("MyList", function(value=0, ...) {
  extend(list(value=value, ...), "MyList")
})

setMethodS3("as.character", "MyList", function(object, ...) {
  fmtstr <- object$fmtstr
  if (is.null(fmtstr))
    fmtstr <- "%.6f"
  sprintf(fmtstr, object$value)
})

setMethodS3("print", "MyList", function(object, ...) {
  print(as.character(object), ...)
})

x <- MyList(3.1415926)
print(x)
x <- MyList(3.1415926, fmtstr="%3.2f")
print(x)
x$fmtstr <- "%e"
print(x)

```

---

getConstructorS3 *Get a constructor method*

---

### Description

Get a constructor method.

### Usage

```
## Default S3 method:  
getConstructorS3(name, ...)
```

### Arguments

name	The name of the constructor function.
...	Not used.

### Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

### See Also

[setConstructorS3\(\)](#), [getMethodS3](#), [isGenericS3](#).

---

getName.environment  
*Gets the name of an environment*

---

### Description

Gets the name of an environment, e.g. "R\_GlobalEnv" or "0x01ddd060".

### Usage

```
## S3 method for class 'environment':  
getName(env, ...)
```

### Arguments

env	An <a href="#">environment</a> .
...	Not used.

### Value

Returns a [character](#) string.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

`environmentName()`.

**Examples**

```
name <- getName(globalenv())
print(name)
stopifnot(identical(name, "R_GlobalEnv"))

getName(new.env())
```

---

hashCode

*Gets an integer hashcoded for R objects*

---

**Description**

Gets an integer hashcoded for R objects.

**Usage**

```
## Default S3 method:
hashCode(object, ...)
```

**Arguments**

<code>object</code>	A <a href="#">vector</a> or <a href="#">list</a> of R objects.
<code>...</code>	Not used.

**Details**

A [character](#) string is converted into a hashcode following Java conventions by  $s[1] * 31^{(n-1)} + s[2] * 31^{(n-2)} + \dots + s[n]$  using integer arithmetic, where  $s[i]$  is the  $i$ :th character of the string,  $n$  is the length of the string. The hash value of the empty string is zero.

For all other objects, by default `as.integer()` is called.

**Value**

Returns a [vector](#) of [integer](#)'s.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

---

InternalErrorException

*InternalErrorException represents internal errors*


---

## Description

Package: R.oo

### Class InternalErrorException

Object

```

~~|
~~+--try-error
~~~~~|
~~~~~+--condition
~~~~~|
~~~~~+--error
~~~~~|
~~~~~+--simpleError
~~~~~|
~~~~~+--Exception
~~~~~|
~~~~~+--InternalErrorException

```

### Directly known subclasses:

```

public static class InternalErrorException
extends Exception

```

InternalErrorException represents internal errors that are likely to be due to implementation errors done by the author of a specific package and not because the user made an error. Errors that are due to unexpected input to functions etc falls under this error type.

## Usage

```
InternalErrorException(..., package=NULL)
```

## Arguments

...	Any arguments accepted by <a href="#">Exception</a>
package	The name ( <a href="#">character</a> string) of the package where the error exists. Can also be a <a href="#">Package</a> object. If <a href="#">NULL</a> , the source of the error is assumed to be unknown.

**Fields and Methods****Methods:**

`getMessage` Gets the message of the exception.  
`getPackage` Gets the suspicious package likely to contain an error.

**Methods inherited from Exception:**

`as.character`, `getCall`, `getLastException`, `getMessage`, `getStackTrace`, `getWhen`, `print`, `printStackTrace`, `throw`

**Methods inherited from error:**

`as.character`, `throw`

**Methods inherited from condition:**

`as.character`, `conditionCall`, `conditionMessage`, `print`

**Methods inherited from Object:**

`<-`, `[]`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

For detailed information about exceptions see [Exception](#).

---

`intToChar`

*Converts a vector of integers into a vector of ASCII characters*

---

**Description**

Converts a vector of ASCII integers to a equal length vector of ASCII characters. To make sure that all values in the input vector are in the range [0,255], the input vector is taken modulo 256.

**Usage**

```
## Default S3 method:
intToChar(i, ...)
```

**Arguments**

`i` An *integer vector*.  
`...` Not used.

**Value**

Returns a ASCII `integer vector`.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

`utf8Conversion.charToInt()`

**Examples**

```
i <- charToInt(unlist(strsplit("Hello world!", split=NULL)))
# Gives: 72 101 108 108 111 32 119 111 114 108 100 33
ch <- intToChar(c(72,101,108,108,111,32,119,111,114,108,100,33))
# Gives: "H" "e" "l" "l" "o" " " "w" "o" "r" "l" "d" "!"
```

---

11	<i>Generates a list of informative properties of all members of an environment</i>
----	--

---

**Description**

Generates a list of informative properties of all members of an environment.

**Usage**

```
## Default S3 method:
ll(pattern=".*", ..., private=FALSE, properties=getOption("R.oo::ll/properties"), s
```

**Arguments**

<code>pattern</code>	Regular expression pattern specifying which members to return. If <code>".*"</code> , all names are matched.
<code>...</code>	A named <code>vector</code> of format <code>functionName=value</code> , where <code>functionName()</code> will be called on each member found. If the result matches the <code>value</code> , the member is returned, otherwise not.
<code>private</code>	If <code>TRUE</code> , also private members, i.e. members with a name starting with a <code>.</code> (period), will be listed, otherwise not.
<code>properties</code>	Names of properties to be returned. There must exist a <code>function</code> with the same name, because it will be called. This way one can extract any type of property by defining new methods.
<code>sortBy</code>	Name or index of column (property) to be sorted by. If <code>NULL</code> , the objects are listed in the order they are found.
<code>envir</code>	An <code>environment</code> , a search path index or a name of a package to be scanned.

**Value**

Returns a `data.frame` containing information about all the members.

**Default properties returned**

It is possible to set the default value of argument `properties` by setting option `"R.oo:ll/properties"`, e.g. `options("R.oo:ll/properties"=c("data.class", "dimension"))`. If this option is not set when the package is loaded, it is set to `c("data.class", "dimension", "objectSize")`.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

`ll.Object()`.

**Examples**

```
## Not run:
To list all objects in .GlobalEnv:
> ll()
      member data.class dimension objectSize
1      *tmp*   Person          1          428
2 as.character.Person function      NULL      1208
3      country character          1          44
4 equals.Person function      NULL      2324
5      filename character          1          84
6      getAge  function      NULL      372
7 getAge.Person function      NULL      612
8 getName.Person function      NULL      628
9 hashCode.Person function      NULL      1196
10 last.warning list            1          192
11      obj     Person          1          428
12      Person Class           NULL      2292
13      setAge function      NULL      372
14 setAge.Person function      NULL      2088
15      setName function      NULL      372
16 setName.Person function      NULL      760
17 staticCode.Person function      NULL      2372

To list all functions in the methods package:
ll(mode="function", envir="methods")

To list all numeric and character object in the base package:
ll(mode=c("numeric", "character"), envir="base")

To list all objects in the base package greater than 40kb:
subset(ll(envir="base"), objectSize > 40000)
## End(Not run)
```

Object

*The root class that every class must inherit from***Description**

R.00

**Class Object**public class **Object**

`Object` is the root class of all other classes. All classes *must* extend this class, directly or indirectly, which means that they all will inherit the methods in this class.

**Usage**`Object (core=NA)`**Arguments**

`core` The core value of each *reference* referring to the `Object`. By default, this is just the smallest possible R object, but there are situations where it is useful to have another kind of core, which is the case with the `Class` class. *Note that this value belongs to the reference variable and not to the `Object`, which means it can not be referenced.*

**Fields and Methods****Methods:**

<code>\$</code>	-
<code>\$&lt;-</code>	-
<code>[[</code>	-
<code>[[&lt;-</code>	-
<code>as.character</code>	Gets a character string representing the object.
<code>attach</code>	Attaches an <code>Object</code> to the R search path.
<code>attachLocally</code>	Attaches an <code>Object</code> locally to an environment.
<code>clearCache</code>	Clear fields that are defined to have cached values.
<code>clone</code>	Clones an <code>Object</code> .
<code>detach</code>	Detach an <code>Object</code> from the R search path.
<code>equals</code>	Compares an object with another.
<code>extend</code>	Extends another class.
<code>finalize</code>	Finalizer methods called when object is clean out.
<code>gc</code>	Clear cached fields and calls the garbage collector.
<code>getEnvironment</code>	Gets the environment of this object.
<code>getFields</code>	Returns the field names of an <code>Object</code> .
<code>getInstantiationTime</code>	Gets the time when the object was instantiated.

<code>getInternalAddress</code>	Gets the memory location where the Object resides.
<code>getStaticInstance</code>	Gets the static instance of this objects class.
<code>hasField</code>	Checks if a field exists or not.
<code>hashCode</code>	Gets a hash code for the Object.
<code>isReferable</code>	Checks if the object is referable or not.
<code>ll</code>	Generates a list of informative properties of all members of an Object.
<code>load</code>	Static method to load an Object from a file or a connection.
<code>names</code>	-
<code>newInstance</code>	Creates a new instance of the same class as this object.
<code>novirtual</code>	Returns a reference to the same Object with virtual fields turned off.
<code>objectSize</code>	Gets the size of the Object in bytes.
<code>print</code>	Prints an Object.
<code>registerFinalizer</code>	Registers a finalizer hook for the object.
<code>save</code>	Saves an Object to a file or a connection.
<code>staticCode</code>	Method that will be call each time a new instance of a class is created.

### Defining static fields

To define a static field of an Object class, use a private field `<.field>` and then create a virtual field `<field>` by defining methods `get<Field>()` and `set<Field>()`. These methods should retrieve and assign the value of the field `<.field>` of the *static* instance of the class. The second example below shows how to do this. The example modifies also the static field already in the constructor, which is something that otherwise may be tricky.

### Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

### References

[1] H. Bengtsson, *The R.oo package - Object-Oriented Programming with References Using Standard R Code*, In Kurt Hornik, Friedrich Leisch and Achim Zeileis, editors, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20-22, Vienna, Austria. <http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/>

### Examples

```
#####
# Defines the class Person with private fields .name and .age, and
# with methods print(), getName(), setName(), getAge() and setAge().
#####
setConstructorS3("Person", function(name, age) {
  if (missing(name)) name <- NA;
  if (missing(age)) age <- NA;

  extend(Object(), "Person",
    .name=name,
    .age=age
  )
})
```

```

    )
  })

  setMethodS3("as.character", "Person", function(this, ...) {
    paste(this$.name, "is", as.integer(this$.age), "years old.");
  })

  setMethodS3("equals", "Person", function(this, obj, ...) {
    ( identical(data.class(this), data.class(obj)) &&
      identical(this$.getName(), obj$.getName()) &&
      identical(this$.getAge(), obj$.getAge()) );
  })

  setMethodS3("hashCode", "Person", function(this, ...) {
    # Get the hashCode() of the '.name' and the '.age' fields
    # using hashCode.default().
    hashCode(this$.name) * hashCode(this$.age);
  })

  setMethodS3("getName", "Person", function(this, ...) {
    this$.name;
  })

  setMethodS3("setName", "Person", function(this, newName, ...) {
    throw("It is not possible to change the name of a Person.");
  })

  setMethodS3("getAge", "Person", function(this, ...) {
    this$.age;
  })

  setMethodS3("setAge", "Person", function(this, newAge, ...) {
    if (!is.numeric(newAge))
      throw("Age must be numeric: ", newAge);
    if (newAge < 0)
      throw("Trying to set a negative age: ", newAge);
    this$.age <- newAge;
  })

#####
# Code demonstrating different properties of the Object class using
# the example class Person.
#####

# Create an object (instance of) the class Person.
p1 <- Person("Dalai Lama", 67)

# 'p1' is an Object of class Person.
print(data.class(p1)) # "Person"

# Prints information about the Person object.

```

```

print(p1)          # "Dalai Lama is 67 years old."

# or equivalent (except that no generic method has to exist):

p1$print()        # "Dalai Lama is 67 years old."

# Shows that no generic method is required if the \ $ operator is used:
print(p1$name()) # "Dalai Lama"

# The following will call p1$name() since there exists a get-()
# method for the 'name' property.
print(p1$name)    # "Dalai Lama"

# and equivalent when using the [[ operator.
print(p1[["name"]]) # "Dalai Lama"

# The following shows that p1$setName(68) is called, simply because
# there exists a set-() method for the 'name' property.
p1$age <- 68      # Will call p1$setAge(68)

# Shows that the age of the Person has been updated:
print(p1)        # "Dalai Lama is 68 years old."

# If there would not exist such a set-() method or field a new
# field would be created:
p1$country <- "Tibet"

# Lists all (non-private) members of the Person object:
print(ll(p1))

# which gives
#   member class      mode   typeof length dim bytes
#   1 country  NULL character character      1 NULL   44

# The following will call p1$setName("Lalai Dama") which will
# throw an exception saying one can not change the name of
# a Person.
tryCatch(p1$name <- "Lalai Dama", error=print)

# The following will call p1$setAge(-4) which will throw an
# exception saying that the age must be a non-negative number.
tryCatch(p1$age <- -100, error=print)

# Attaches Object 'p1' to the search path.
attach(p1)

# Accesses the newly created field 'country'.
print(country)    # "Tibet"

# Detaches Object 'p1' from the search path. Note that all
# modifications to 'country' are lost.
country <- "Sweden"
detach(p1)

```

```

print(pl$country)    # "Tibet"

# Saves the Person object to a temporary file.
filename <- tempfile("R.methodsS3.example")
save(pl, filename)

# Deletes the object
rm(pl)

# Loads an Object (of "unknown" class) from file using the
# static method load() of class Object.
obj <- Object$load(filename)

# Prints information about the new Object.
print(obj)

# Lists all (non-private) members of the new Object.
print(ll(obj))

#####
# Example illustrating how to "emulate" static fields using virtual
# fields, i.e. get- and set-methods. Here we use a private static
# field '.count' of the static class instance 'MyClass', i.e.
# MyClass$.count. Then we define a virtual field 'count' via method
# getCount() to access this static field. This will make all queries
# for 'count' of any object to use the static field instead. In the
# same way is assignment controlled via the setCount() method. A
# side effect of this way of coding is that all MyClass instances will
# also have the private field '.count' (set to zero except for the
# static field that is).
#####
setConstructorS3("MyClass", function(...) {
  # Create an instance (the static class instance included)
  this <- extend(Object(), "MyClass",
    .count = 0
  )

  # In order for a static field to be updated in the
  # constructor it has to be done after extend().
  this$count <- this$count + 1;

  # Return the object
  this;
})

setMethodS3("as.character", "MyClass", function(this, ...) {
  paste(class(this)[1], ": Number of instances: ", this$count, sep="");
})

# Get virtual field 'count', e.g. obj$count.
setMethodS3("getCount", "MyClass", function(this, ...) {
  MyClass$.count;
})

```

```
  })

  # Set virtual field 'count', e.g. obj$count <- value.
  setMethodS3("setCount", "MyClass", function(this, value, ...) {
    MyClass$count <- value;
  })

  # Create four instances of class 'MyClass'
  obj <- lapply(1:4, MyClass)
  print(obj)
  print(MyClass$count)
  print(obj[[1]]$count)

  stopifnot(obj[[1]]$count == length(obj))
  stopifnot(MyClass$count == length(obj))
```

---

objectSize

*Gets the size of the object in bytes*

---

### Description

Gets the size of the object in bytes. This method is just a wrapper for `object.size`.

### Usage

```
## Default S3 method:
objectSize(...)
```

### Arguments

... Arguments passed to `object.size`.

### Value

Returns an `integer`.

### Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

### See Also

Internally `object.size`.

---

```
objectSize.environment
```

*Gets the size of an environment in bytes*

---

**Description**

Gets the size of an environment in bytes.

**Usage**

```
## S3 method for class 'environment':  
objectSize(envir, ...)
```

**Arguments**

<code>envir</code>	An <code>environment()</code> .
<code>...</code>	Arguments passed to <code>ls()</code> .

**Value**

Returns an `integer`.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

Internally `object.size`.

---

Package	<i>The Package class provides methods for accessing package information</i>
---------	---

---

**Description**

Package: R.oo  
**Class Package**

```
Object  
~~|  
~~+--Package
```

**Directly known subclasses:**

```
public class Package
extends Object
```

Creates a Package that can be thrown and caught. The Package class is the root class of all other Package classes.

**Usage**

```
Package (name=NULL)
```

**Arguments**

```
name           Name of the package.
```

**Fields and Methods****Methods:**

<a href="#">as.character</a>	Gets a string representation of this package.
<a href="#">getAuthor</a>	Gets the Author of this package.
<a href="#">getBundle</a>	Gets the Bundle that this package might belong to.
<a href="#">getBundlePackages</a>	Gets the names of the other packages that is in the same bundle as this package.
<a href="#">getChangeLog</a>	Gets the change log of this package.
<a href="#">getClasses</a>	Gets all classes of a package.
<a href="#">getContents</a>	Gets the contents of this package.
<a href="#">getContribUrl</a>	Gets the URL(s) from where this package can be installed.
<a href="#">getDataPath</a>	Gets the path to the data (data/) directory of this package.
<a href="#">getDate</a>	Gets the date when package was build.
<a href="#">getDescription</a>	Gets the description of the package.
<a href="#">getDescriptionFile</a>	Gets the description file of this package.
<a href="#">getDevelUrl</a>	Gets the URL(s) from where the developers version of this package can be installed.
<a href="#">getDocPath</a>	Gets the path to the accompanying documentation (doc/) directory of this package.
<a href="#">getEnvironment</a>	Gets the environment of a loaded package.
<a href="#">getExamplePath</a>	Gets the path to the example (R-ex/) directory of this package.
<a href="#">getHistory</a>	-
<a href="#">getHowToCite</a>	Gets the howToCite of this package.
<a href="#">getLicense</a>	Gets the License of this package.
<a href="#">getMaintainer</a>	Gets the Maintainer of this package.
<a href="#">getName</a>	Gets the name of this package.
<a href="#">getNews</a>	-
<a href="#">getPath</a>	Gets the library (system) path to this package.
<a href="#">getPosition</a>	Gets the search path position of the package.
<a href="#">getTitle</a>	Gets the Title of this package.
<a href="#">getUrl</a>	Gets the URL of this package.
<a href="#">getVersion</a>	Gets the version of this package.

<code>isLoaded</code>	Checks if the package is installed on the search path or not.
<code>ll</code>	Generates a list of informative properties of all members of the package.
<code>load</code>	Loads a package.
<code>showChangeLog</code>	Show the change log of this package.
<code>showContents</code>	Show the CONTENTS file of this package.
<code>showDescriptionFile</code>	Show the DESCRIPTION file of this package.
<code>showHistory</code>	-
<code>showHowToCite</code>	Show the HOWTOCITE file of this package.
<code>showNews</code>	-
<code>unload</code>	Unloads a package.
<code>update</code>	Updates the package if a newer version is available.

**Methods inherited from Object:**

`<-`, `[[`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**Examples**

```
## Not run: # By defining .First.lib() as follows in zzz.R for a package, an
# instance of class Package with the same name as the package will
# be made available on the search path. More over, the code below
# will also inform the user that the package has been loaded:
#
# > library(R.oo)
# R.oo v0.52 (2003/04/13) was successfully loaded.
#
.First.lib <- function(libname, pkgname) {
  pkg <- Package(pkgname);
  assign(pkgname, pkg, pos=getPosition(pkg));
  cat(getName(pkg), " v", getVersion(pkg), " (" , getDate(pkg), ")",
      " was successfully loaded.\n", sep="");
}

# The Package class works for any packages, loaded or not.

# Some information about the base package
pkg <- Package("base")
print(pkg)
# [1] "Package: base v1.6.2 (NA) is loaded (pos=5). The official webpage
#      is NA and the maintainer is R Core Team <R-core@r-project.org>. The
#      package is installed in c:/PROGRA~1/R/rw1062/library/base/."
print(list.files(Package("base")$dataPath))

# Some information about the R.oo package
print(R.oo)
# [1] "Package: R.oo v0.52 (2003/04/13) is loaded (pos=2). The official
```

```
# webpage is http://www.braju.com/R/ and the maintainer is Henrik
# Bengtsson <henrikb@braju.com>. The package is installed in
# c:/PROGRA~1/R/rw1062/library/R.oo/."

# To check for updates and update a package, just do
update(R.oo)

## End(Not run)
```

---

RccViolationException

*An RccViolationException indicates a violation of the R Coding Conventions (RCC)*

---

## Description

Package: R.oo

### Class RccViolationException

#### Object

```
~~|
~~+--try-error
~~~~~|
~~~~~+--condition
~~~~~|
~~~~~+--error
~~~~~|
~~~~~+--simpleError
~~~~~|
~~~~~+--Exception
~~~~~|
~~~~~+--RccViolationException
```

#### Directly known subclasses:

```
public static class RccViolationException
extends Exception
```

An RccViolationException indicates a violation of the R Coding Conventions (RCC). It is generated by `setConstructorS3()` and `setMethodS3()`. It is *not* meant to be caught, but instead the source code that violates the RCC should be fixed. For more information about RCC, see references below.

## Usage

```
RccViolationException(...)
```

**Arguments**

... Any arguments accepted by the constructor of Exception, i.e. one or several `character` strings, which will be concatenated and contain informative message about why the RCC was violated.

**Details**

Since it is not possible to assert that the RCC is followed during the parsing of the source code, but first only when the source code is actually executed.

**Fields and Methods****Methods:**

`as.character` Gets a string representing of the RCC violation.  
`getRccUrl` Static method to get a URL where the RCC can be found.

**Methods inherited from Exception:**

`as.character`, `getCall`, `getLastException`, `getMessage`, `getStackTrace`, `getWhen`, `print`, `printStackTrace`, `throw`

**Methods inherited from error:**

`as.character`, `throw`

**Methods inherited from condition:**

`as.character`, `conditionCall`, `conditionMessage`, `print`

**Methods inherited from Object:**

`<-`, `[[`, `[[<-`, `as.character`, `attach`, `attachLocally`, `clearCache`, `clone`, `detach`, `equals`, `extend`, `finalize`, `gc`, `getEnvironment`, `getFields`, `getInstantiationTime`, `getStaticInstance`, `hasField`, `hashCode`, `ll`, `load`, `objectSize`, `print`, `registerFinalizer`, `save`

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

See also `try()` and `tryCatch()`. For detailed information about exceptions see [Exception](#). The R Coding Conventions (RCC) can be found at <http://www.maths.lth.se/help/R/RCC/>.

**Examples**

```
## Not run:
setConstructorS3("myClass", function() { extends(Object(), .value=0) })
setMethodS3("MyMethod", "myClass", function(this) { "Hullo!" })

## End(Not run)
```

Rdoc

*Class for converting Rdoc comments to Rd files***Description**

Package: R.oo

**Class Rdoc**[Object](#)

~~|

~~+---Rdoc

**Directly known subclasses:**public static class **Rdoc**extends [Object](#)

Class for converting Rdoc comments to Rd files.

**Usage**

Rdoc ()

**Fields and Methods****Methods:**

<a href="#">argsToString</a>	Gets the arguments signature of a function.
<a href="#">check</a>	Checks the compiled Rd files.
<a href="#">compile</a>	Compile source code files containing Rdoc comments into Rd files.
<a href="#">createManPath</a>	Creates the directory where the Rd files should be saved.
<a href="#">createName</a>	Creates a class-method name.
<a href="#">declaration</a>	Gets the class declaration.
<a href="#">escapeRdFilename</a>	Escape non-valid characters in a filename.
<a href="#">getClassS4Usage</a>	Gets the usage of a S4 class.
<a href="#">getKeywords</a>	Gets the keywords defined in R with descriptions.
<a href="#">getManPath</a>	Gets the path to the directory where the Rd files will be saved.
<a href="#">getNameFormat</a>	Gets the current name format.
<a href="#">getPackageNameOf</a>	Gets the package of a method or an object.
<a href="#">getRdTitle</a>	Extracts the title string of a Rd file.
<a href="#">getUsage</a>	Gets the usage of a method.
<a href="#">hierarchy</a>	Gets the class hierarchy.
<a href="#">isKeyword</a>	Checks if a word is a Rd keyword.
<a href="#">isVisible</a>	Checks if a member is visible given its modifiers.

`methodsInheritedFrom` Gets all methods inherited from a class in Rd format.  
`setManPath` Sets the path to the directory where the Rd files should be saved.  
`setNameFormat` Sets the current name format.

**Methods inherited from Object:**

,<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**References**

R developers, *Guidelines for Rd files*, <http://developer.r-project.org/Rds.html>, 2003

**Examples**

```
## Not run: # Set default author
author <- "Henrik Bengtsson, \url{http://www.braju.com/R/}"

# Show the file containing the Rdoc comments
rdocFile <- system.file("misc", "ASCII.R", package="R.oo")
file.show(rdocFile)

# Compile the Rdoc:s into Rd files (saved in the destPath directory)
destPath <- tempdir()
Rdoc$compile(rdocFile, destPath=destPath)

# List the generated Rd files
rdFiles <- list.files(destPath, full.names=TRUE)
print(rdFiles)

# Show one of the files
file.show(rdFiles[1])

# Clean up
file.remove(rdFiles)
## End(Not run)
```

**Description**

Package: R.oo

**Class RdocException**

Object

```

~~|
~~+--try-error
~~~~~|
~~~~~+--condition
~~~~~|
~~~~~+--error
~~~~~|
~~~~~+--simpleError
~~~~~|
~~~~~+--Exception
~~~~~|
~~~~~+--RdocException

```

**Directly known subclasses:**

```

public static class RdocException
extends Exception

```

RdocException are thrown by the Rdoc compiler when it fails to generate a Rd file from an Rdoc comment.

**Usage**

```
RdocException(..., source=NULL)
```

**Arguments**

...	Any arguments accepted by <a href="#">Exception</a>
source	Object specifying the source where the Rdoc error occurred. This is commonly a filename <a href="#">character</a> string.
.	

**Fields and Methods****Methods:**

<a href="#">as.character</a>	Gets a character string representing of the RdocException.
<a href="#">getSource</a>	Gets the source of the exception.

**Methods inherited from Exception:**

as.character, getCall, getLastException, getMessage, getStackTrace, getWhen, print, printStackTrace, throw

**Methods inherited from error:**

as.character, throw

**Methods inherited from condition:**

as.character, conditionCall, conditionMessage, print

**Methods inherited from Object:**

,<-, [[, [[<-, as.character, attach, attachLocally, clearCache, clone, detach, equals, extend, finalize, gc, getEnvironment, getFields, getInstantiationTime, getStaticInstance, hasField, hashCode, ll, load, objectSize, print, registerFinalizer, save

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

For detailed information about exceptions see [Exception](#).

---

setConstructorS3     *Defines a class in S3/UseMethod style*

---

**Description**

Defines a class in R.oo/S3 style. The class name is validated so it starts with a letter and it also gives a [warning](#) if its first letter is *not* capital. The reason for this is to enforce a naming convention that names classes with upper-case initial letters and methods with lower-case initial letters (this is also the case in for instance Java).

What this function currently does is simply creating a constructor function for the class.

*Note: The constructor must be able to be called with no arguments, i.e. use default values for all arguments or make sure you use `missing()` or similar!* For instance the following definition is *not* correct: `setConstructorS3("Foo", function(x) extend(Object(), "Foo", x=x))` whereas this one is `setConstructorS3("Foo", function(x=NA) extend(Object(), "Foo", x=x))`

**Usage**

```
## Default S3 method:
setConstructorS3(name, definition, private=FALSE, protected=FALSE, static=FALSE, ab
```

**Arguments**

name	The name of the class.
definition	The constructor definition. <i>Note: The constructor must be able to be called with no arguments, i.e. use default values for all arguments or make sure you use <code>missing()</code> or similar!</i>
static	If <b>TRUE</b> this class is defined to be static, otherwise not. Currently this has no effect expect as an indicator.
abstract	If <b>TRUE</b> this class is defined to be abstract, otherwise not. Currently this has no effect expect as an indicator.
private	If <b>TRUE</b> this class is defined to be private.
protected	If <b>TRUE</b> this class is defined to be protected.
trial	If <b>TRUE</b> this class is defined to be a trial class, otherwise not. A trial class is a class that is introduced to be tried out and it might be modified, replaced or even removed in a future release. Some people prefer to call trial versions, beta version. Currently this has no effect expect as an indicator.
deprecated	If <b>TRUE</b> this class is defined to be deprecated, otherwise not. Currently this has no effect expect as an indicator.
envir	The environment for where the class (constructor function) should be stored.
enforceRCC	If <b>TRUE</b> , only class names following the R Coding Convention is accepted. If the RCC is violated an <code>RccViolationException</code> is thrown.
...	Not used.

Note: If a constructor is not declared to be private nor protected, it will be declared to be public.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

**See Also**

To define a method see [setMethodS3](#). For information about the R Coding Conventions, see [RccViolationException](#). For a thorough example of how to use this method see [Object](#).

**Examples**

```
## Not run: For a complete example see help(Object).
```

---

throw	<i>Throws an Exception</i>
-------	----------------------------

---

### Description

Throws an exception similar to `stop()`, but with support for exception classes. The first argument (`object`) is by default pasted together with other arguments (`...`) and with separator `sep=""`. For instance, to throw an exception, write

```
throw("Value out of range: ", value, ".")
```

which is short for

```
throw(Exception("Value out of range: ", value, "."))
```

Note that `throw()` can be defined for specific classes, which can then be caught (or not) using `tryCatch()`.

### Usage

```
## Default S3 method:  
throw(...)
```

### Arguments

`...` One or several strings that are concatenated and collapsed into one message string.

### Value

Returns nothing.

### Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

### See Also

See the `Exception` class for more detailed information.

### Examples

```
rbern <- function(n=1, prob=1/2) {  
  if (prob < 0 || prob > 1)  
    throw("Argument 'prob' is out of range: ", prob)  
  rbinom(n=n, size=1, prob=prob)  
}  
  
rbern(10, 0.4)  
# [1] 0 1 0 0 0 1 0 0 1 0  
tryCatch(rbern(10, 10*0.4),
```

```

    error=function(ex) {}
  )

```

---

<code>throw.error</code>	<i>Throws (rethrows) an object of class 'error'</i>
--------------------------	---

---

### Description

Rethrows an 'error' object. The 'error' class was introduced in R v1.8.1 with the new error handling mechanisms.

### Usage

```

## S3 method for class 'error':
throw(error, ...)

```

### Arguments

<code>error</code>	An object or class 'error'.
<code>...</code>	Not used.

### Value

Returns nothing.

### Author(s)

Henrik Bengtsson (<http://www.braju.com/R/>)

### See Also

See the `tryCatch()` method etc. See the [Exception](#) class for more detailed information.

---

<code>trim</code>	<i>Converts to a string and removes leading and trailing whitespace</i>
-------------------	---

---

### Description

Converts to a string and removes leading and trailing whitespace.

### Usage

```

## Default S3 method:
trim(object, ...)

```

**Arguments**

object        A [vector](#) of R objects to be trimmed.  
...            Not used.

**Value**

Returns a [vector](#) of [character](#) strings.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

---

typeOfClass        *Gets the type of a class (S3 or S4)*

---

**Description**

Gets the type of a class (S3 or S4).

**Usage**

```
## Default S3 method:  
typeOfClass(object, ...)
```

**Arguments**

object        The object to be checks.  
...            Not used.

**Value**

Returns a [character](#) string "S3", "S3-Object" or "S4", or [NA](#) if neither.

**Author(s)**

Henrik Bengtsson (<http://www.braju.com/R/>)

# Index

## \*Topic **attribute**

- dimension, 5
- equals, 6
- objectSize, 23
- objectSize.environment, 24

## \*Topic **character**

- charToInt, 3
- intToChar, 15
- trim, 35
- typeofClass, 36

## \*Topic **classes**

- Class, 4
- Exception, 7
- InternalErrorException, 14
- Object, 18
- Package, 24
- RccViolationException, 27
- Rdoc, 29
- RdocException, 30

## \*Topic **documentation**

- Rdoc, 29

## \*Topic **error**

- Exception, 7
- InternalErrorException, 14
- RccViolationException, 27
- RdocException, 30
- throw, 34
- throw.error, 35

## \*Topic **methods**

- Class, 4
- Exception, 7
- extend, 10
- getConstructorS3, 12
- getName.environment, 12
- hashCode, 13
- InternalErrorException, 14
- Object, 18
- objectSize.environment, 24
- Package, 24

- RccViolationException, 27
- RdocException, 30
- setConstructorS3, 32
- throw.error, 35

## \*Topic **package**

- R.oo-package, 2

## \*Topic **programming**

- Class, 4
- Exception, 7
- extend, 10
- getConstructorS3, 12
- getName.environment, 12
- hashCode, 13
- InternalErrorException, 14
- Object, 18
- Package, 24
- RccViolationException, 27
- RdocException, 30
- setConstructorS3, 32

## \*Topic **utilities**

- dimension, 5
- equals, 6
- ll, 16
- objectSize, 23
- objectSize.environment, 24

- argsToString, 5, 29

- as.character, 5, 8, 18, 25, 28, 31

- attach, 18

- attachLocally, 18

- character, 3, 13, 14, 28, 31, 36

- charToInt, 3, 16

- check, 29

- Class, 4

- clearCache, 18

- clone, 18

- compile, 29

- createManPath, 29

- createName, 29

- data.frame, 17
- declaration, 29
- detach, 18
- dim, 6
- dimension, 5
  
- environment, 12, 16, 24
- environmentName, 13
- equals, 6, 18
- escapeRdFilename, 29
- Exception, 7, 14, 15, 27, 28, 31, 32, 34, 35
- extend, 10, 18
  
- FALSE, 6, 7
- finalize, 18
- forName, 5
- function, 4, 16
  
- gc, 18
- getAuthor, 25
- getBundle, 25
- getBundlePackages, 25
- getChangeLog, 25
- getClasses, 25
- getClassS4Usage, 29
- getConstructorS3, 12
- getContents, 25
- getContribUrl, 25
- getDataPath, 25
- getDate, 25
- getDescription, 25
- getDescriptionFile, 25
- getDetails, 5
- getDevelUrl, 25
- getDocPath, 25
- getEnvironment, 18, 25
- getExamplePath, 25
- getFields, 5, 18
- getHowToCite, 25
- getInstantiationTime, 18
- getInternalAddress, 19
- getKeywords, 29
- getKnownSubclasses, 5
- getLastException, 8
- getLicense, 25
- getMaintainer, 25
- getManPath, 29
- getMessage, 8, 15
- getMethods, 5
- getMethodS3, 12
- getName, 5, 25
- getName.environment, 12
- getNameFormat, 29
- getPackage, 5, 15
- getPackageNameOf, 29
- getPath, 25
- getPosition, 25
- getRccUrl, 28
- getRdDeclaration, 5
- getRdHierarchy, 5
- getRdMethods, 5
- getRdTitle, 29
- getSource, 31
- getStackTrace, 8
- getStackTraceString, 8
- getStaticInstance, 5, 19
- getSuperclasses, 5
- getTitle, 25
- getUrl, 25
- getUsage, 29
- getVersion, 25
- getWhen, 8
  
- hasField, 19
- hashCode, 13, 19
- hierarchy, 29
  
- identical, 6, 7
- integer, 3, 6, 13, 15, 16, 23, 24
- InternalErrorException, 7, 14
- intToChar, 4, 15
- isAbstract, 5
- isBeingCreated, 5
- isDeprecated, 5
- isGenericS3, 12
- isKeyword, 29
- isLoaded, 26
- isPrivate, 5
- isProtected, 5
- isPublic, 5
- isReferable, 19
- isStatic, 5
- isVisible, 29
  
- length, 6
- list, 13
- ll, 16, 19, 26
- ll.default, 6

ll.Object, 17  
load, 19, 26  
ls, 24

methodsInheritedFrom, 30

NA, 36  
newInstance, 5, 19  
novirtual, 19  
NULL, 6, 14, 16

Object, 2, 4, 7, 14, 18, 24, 25, 27, 29, 31, 33  
object.size, 23, 24  
objectSize, 19, 23  
objectSize.environment, 24

Package, 14, 24  
print, 5, 8, 19  
printStackTrace, 8

R.oo (R.oo-package), 2  
R.oo-package, 2  
RccViolationException, 7, 27, 33  
Rdoc, 29  
RdocException, 7, 30  
registerFinalizer, 19

save, 19  
setConstructorS3, 12, 32  
setManPath, 30  
setMethodS3, 33  
setNameFormat, 30  
showChangeLog, 26  
showContents, 26  
showDescriptionFile, 26  
showHowToCite, 26  
staticCode, 19

throw, 8, 34  
throw.error, 35  
trim, 35  
TRUE, 6, 7, 16, 33  
try, 9, 28  
tryCatch, 9, 28, 34  
typeofClass, 36

unload, 26  
update, 26  
utf8Conversion, 4, 16

vector, 3, 6, 13, 15, 16, 36

warning, 32