Package 'oops'

October 14, 2022

Type Package

Title S3 Style Object Oriented Programming

Version 0.2.0

Author Christopher Mann <cmann3@unl.edu>

Maintainer Christopher Mann <cmann3@unl.edu>

Description Create simple, hassle-free classes with reference semantics similar to 'RefClass' or 'R6' but relying on S3 methods. ``oops'' class instances tend to be lighter weight and faster to create. Creating a class is as easy creating a list, while generating an instance is a simple function call. Support for inheritance and fixed field classes.

License MIT + file LICENSE

Imports utils

Suggests knitr, rmarkdown

Encoding UTF-8

RoxygenNote 7.1.1

LazyData true

Depends R (>= 3.5)

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-03-02 22:50:02 UTC

R topics documented:

`add_fields` *Add Fields to oClasses and Other Objects*

Description

For environments, oClass instances and generator, `add_fields` is a wrapper for `list2env`; it adds the objects in `...` the environment if they are named. For list and other objects, it behaves similar to appending `...` as a list.

Usage

```
add_fields(x, ...)
```

Arguments

`x`	oClass instance, generator, environment, list, or other object
`...`	named objects to be added to `x`

Value

object of same type as `x` or list

Examples

```
clown <- oClass("clown")
add_fields(clown, laugh = "haha", is_funny=TRUE)

clown$laugh
```

`as.oClass` *Convert Object to an oClass Generator*

Description

This function takes any named object such as an environment, fully-named list, or an Instance and converts it to an `oClass` generator function so that instances have access to the fields in the named object. See `oClass` for details about the arguments and functionality of the oClass generator.

Usage

```
as.oClass(
  x,
  name = NULL,
  inherit = NULL,
  portable = FALSE,
  hash = FALSE,
  formals = NULL,
  deep = TRUE,
  ...
)
```

Arguments

`x`	object to be cloned and converted
`name`	character string describing the name of the class
`inherit`	oClass used as the `parent.env` for the generated instances
`portable`	logical indicating whether all inherited values should be copied into each instance
`hash`	logical indicating whether instances should use hashing, see `new.env`
`formals`	list containing the formal arguments for the resulting generator function. These are passed to the `init` function when a new instance is created.
`deep`	logical. Should the object be deep-cloned?
`...`	named fields inherited by the class instance

Value

a function of class `"ClassGenerator"` with attributes describing each generated class instance

change_formals *Change the Formal Arguments of a oClass Generator*

Description

This accepts an oClass generator and updates its formal arguments based either on those passed in ... or the function passed to from_init. The results will be passed to the appropriate init function each time an instance is generated.

Usage

```
change_formals(x, ..., envir = parent.frame(), from_init = NULL)
```

Arguments

x	oClass generator function
...	named or unnamed objects used as the formal arguments of the generator function
envir	environment from which to evaluate arguments
from_init	function containing the formal arguments to use; typically an init function. ... and envir are ignored if not NULL.

Value

oClass generator function

Examples

```
clown <- oClass("clown")
clown

# 'init' requires a laugh
init.clown <- function(x, laugh, ...){
  x$laugh <- laugh
  add_fields(x, ...)
  return(x)
}

# change formals of clown
clown <- change_formals(clown, from_init = init.clown)

# alternatively,
clown <- change_formals(clown, laugh, ..dots)

# creation
happy_clown <- clown("HAHA")
sad_clown   <- clown("ha")
```

change_inherit *Change the Inheritance of an oClass*

Description

This function takes twos `oClass` generator function and alters the first so that it inherits the template and classes of the second. Existing instances will inherit the objects contained in the new parent, but will not gain access to the S3 methods.

Usage

```
change_inherit(x, parent)
```

Arguments

`x`	oClass generator function
`parent`	oClass generator function from which `x` inherits

Value

oClass generator function

Examples

```
typist <- oClass("typist")
job    <- oClass("job", hours = 40, pay=15)

typist <- change_inherit(typist, job)
typist$hours
```

clone *Create a Copy of an oClass Instance*

Description

A copy of all objects and attributes within an environment. If `deep=TRUE`, all objects inside of `x`, including other environments, will also be deeply "cloned". The global and base environments will not be cloned.

Usage

```
clone(x, deep = FALSE, ...)

clone_attributes(x, deep = FALSE, cloned = NULL)
```

Arguments

x	environment of class "Instance"
deep	logical for whether clone should be applied to all objects.
...	arguments passed to methods
cloned	environment containing references to environments that have already been cloned. This is passed to internal methods when deep=TRUE and should not be set directly.

Value

environment of class "Instance"

Functions

- clone_attributes: Clone the attributes of an object.

---

cpi_data *Price Inflation Data*

Description

This data set contains monthly observations of annualized price inflation from January 1949 until November 2021. Price inflation is calculated by taking the log difference between the CPI for Urban Consumers in one period and its value exactly one year earlier.

Usage

```
cpi_data
```

Format

Data frame with 875 rows and 7 variables:

date date in "YYYY-MM-DD" format

pi price inflation in decimal format

pi.1 price inflation last month

pi.2 price inflation two months ago

pi.3 price inflation three months ago

pi.6 price inflation six months ago

pi.12 price inflation one year ago

Source

https://fred.stlouisfed.org/series/CPIAUCSL

Extract *Extract or Replace Parts of a Class or Instance*

Description

Operators acting on oClass generators and their instances.

Usage

```
## S3 method for class 'ClassGenerator'
x$name

## S3 method for class 'ClassGenerator'
x[[i, exact = TRUE, inherits = TRUE]]

## S3 replacement method for class 'ClassGenerator'
x$name <- value

## S3 replacement method for class 'ClassGenerator'
x[[name]] <- value

## S3 method for class 'Instance'
x$name

## S3 method for class 'Instance'
x[[i, exact = TRUE, inherits = TRUE]]
```

Arguments

x	object of class "Instance" or "ClassGenerator"
i, name	character or symbol for `$` describing field name to return or set
exact	logical controlling whether a partial match is acceptable. Defaults to TRUE for no partial matching
inherits	logical describing whether parent environments should be searched
value	new field value

Details

For oClass instances, `$` and `[` first search the instance environment for the object. If no object is found, then all inherited objects are searched in order. Any object assigned to the instance will be inserted into the instance's environment. These operators act on the underlying Class template environment when applied to a Class generator.

Value

Environment of class "Instance" or function of class "ClassGenerator"

`init` *Initialize Class Instance*

Description

Function called on `oClass` instance when it is created. Users create `init` methods to customize creation behavior for their Classes. All `init` methods should return the Instance. `init_next` calls the objects next `init` methods. `init_next` should only be used inside if `init`.

Usage

```
init(x, ...)

init_next(x, ...)
```

Arguments

`x`	environment of class `"Instance"`
`...`	named fields inherited by the class instance or passed to methods

Value

environment of class `"Instance"`

Functions

- `init_next`: Initialize the inherited Class.

Examples

```
Animal <- oClass("Animal")

init.Animal <- function(self, x, y){
  self$x <- x
  self$y <- y
  self
}

turtle <- Animal(5, 10)
turtle$x == 5    # TRUE
turtle$y == 10   # TRUE
```

init.Instance *Init Method for Instance*

Description

See `init` for details.

Usage

```
## S3 method for class 'Instance'
init(x, ...)
```

Arguments

`x`	environment of class `"Instance"`
`...`	named fields inherited by the class instance or passed to methods

Value

environment of class `"Instance"`

is.Instance *Is Object a Class Instance?*

Description

Check whether object inherits the `"Instance"` class. See `is.oClass` to check whether object is a `oClass` generator.

Usage

```
is.Instance(x)
```

Arguments

`x`	object to be tested

Value

`TRUE` if object inherits `"Instance"`, `FALSE` otherwise

is.oClass *Is Object an "oClass" Generator?*

Description

Check whether object inherits the `"ClassGenerator"` class. This is used to check `oClass` generators, not the instance. See `is.Instance` to check whether object is an oClass instance.

Usage

```
is.oClass(x)

is.ClassGenerator(x)
```

Arguments

`x`	object to be tested

Value

`TRUE` if object inherits `"ClassGenerator"`, `FALSE` otherwise

Functions

- `is.ClassGenerator`: check whether object is an oClass generator

oClass *Create an Object Class*

Description

Create a function used to generate instances (environments) with the specified class and fields.

Usage

```
oClass(
  name = NULL,
  inherit = NULL,
  portable = FALSE,
  hash = FALSE,
  formals = NULL,
  ...
)
```

Arguments

name	character string describing the name of the class
inherit	oClass used as the parent.env for the generated instances
portable	logical indicating whether all inherited values should be copied into each instance
hash	logical indicating whether instances should use hashing, see new.env
formals	list containing the formal arguments for the resulting generator function. These are passed to the init function when a new instance is created.
...	named fields inherited by the class instance

Details

oClass is used to create classes with reference semantics that modify in place similar to R5 and R6 classes. Unlike those, functions on oClass instances dispatch using the standard S3 dispatch system. Furthermore, oClass objects and instances are created similar to other R objects to ensure that they are easy and painless to use.

To create a new object class, provide its name and a named list of its fields and their default values. This generates a function that creates a new "instance" of the class each time that it is called. For example, poly <- oClass("polygon", sides = NA) creates a new class called "polygon" with a field called "sides" that can be created using poly(). Object methods that act on the instance are created in the same manner as S3 methods. Therefore, class methods should be created separately.

Each instance of the object class is an environment. The parent environment of the instance is attached to the attributes of the function created by the oClass function. This environment in the function attributes serves as a instance template. Any variables that are specified during the creation of the object instance are placed within the environment of said instance. When searching for an object within an instance, the instance environment is first searched, then the template. This ensures that each object instance remains as small as necessary and minimizes copying. A hashmap is not used by default so that the instance size is smaller, but this can be changed by the oClass function.

oClass objects can also inherit other class objects. If another class object is inherited, the template environment in the inherited object's attributes is added to each instances search path. Furthermore, the name of the inherited class **(and all classes it inherits)** is added to each instance's S3 class. If an environment is inherited, then it is added to the search path.

Since oClass relies on pointers to other environments, oClass instances are generally not portable. If portable=TRUE is added, then each instance will include the default values of each inherited oClass. This generally increases creation time and memory usage, but may result in marginally faster field access. If the fields are relatively few and small, though, memory usage may decline when each Instance is portable.

oClass instances automatically call init when created. Write custom S3 methods for init to control this behavior. This requires the Class to be named so that instances inherit the named S3 class. The formals defines the Class generator's formal function arguments. If used, then an init method for the Class should be created with identical formal arguments; otherwise, instance creation may fail. If no formals are defined, then all objects passed to the generator function are passed to init at creation.

Value

a function of class "ClassGenerator" with attributes describing each generated class instance

Examples

```
## Creating a Stack
stack <- oClass(
  "stack",
  data = list()
)

# Methods
print.stack <- function(x, ...) print(x$data, ...)
push <- function(x, item){
  x$data[[length(x$data)+1]] <- item
  x
}
pop <- function(x){
  n <- length(x$data)
  last <- x$data[[n]]
  x$data[[n]] <- NULL
  last
}

# Create a new instance
x <- stack()
push(x, 6)
push(x, 7)

identical(x$data, list(6, 7)) # TRUE

last <- pop(x)
identical(last, 7)            # TRUE
identical(x$data, list(6))    # TRUE


## Person/Student
##   Example of Inheritance and using Formals

# Declare formal arguments of Person Generator
Person <- oClass(
  "Person",
  formals = list(first, last)
)

# Formal arguments of init should match Person
init.Person <- function(x, first, last){
  x$first <- first
  x$last  <- last
  return(x)
}
```

```
# Create init for Student class
init.Student <- function(x, first, last, year = 1, major = "Econ", ...){
  x$year  <- year
  x$major <- major
  add_fields(x, ...)
  init_next(x, first = first, last = last)
  return(x)
}

# Create Student class, inherits Person
Student <- oClass(
  "Student",
  inherit = Person,
  formals = init.Student
)

# Creating a student
Student("Chris", "Mann", 4, gpa = 4.0)
```

Index