R Reference Class

R Reference Class

In this article, you will learn to work with reference classes in R programming which is one of the three class systems (other two are S3 and S4).

Reference class in R programming is similar to the object oriented programming we are used to seeing in common languages like C++, Java, Python etc.

Unlike S3 and S4 classes, methods belong to class rather than generic functions. Reference class are internally implemented as S4 classes with an environment added to it.

How to define a reference class?

Defining reference class is similar to defining a S4 class. Instead of `setClass()` we use the `setRefClass()` function.

```
> setRefClass("student")
```

Member variables of a class, if defined, need to be included in the class definition. Member variables of reference class are called fields (analogous to slots in S4 classes).

Following is an example to define a class called `student` with 3 fields, `name`, `age` and `GPA`.

```
> setRefClass("student", fields = list(name = "character", age = "numeric", GPA = "numeric"))
```

How to create a reference objects?

The function `setRefClass()` returns a generator function which is used to create objects of that class.

```
> student <- setRefClass("student",
fields = list(name = "character", age = "numeric", GPA = "numeric"))
> # now student() is our generator function which can be used to create new objects
> s <- student(name = "John", age = 21, GPA = 3.5)
> s
Reference class object of class "student"
Field "name":
[1] "John"
Field "age":
[1] 21
Field "GPA":
[1] 3.5
```

How to access and modify fields?

Fields of the object can be accessed using the `$` operator.

```
> s$name
[1] "John"
> s$age
[1] 21
> s$GPA
[1] 3.5
```

Similarly, it is modified by reassignment.

```
> s$name <- "Paul"
> s
Reference class object of class "student"
Field "name":
[1] "Paul"
Field "age":
[1] 21
Field "GPA":
[1] 3.5
```

Warning Note

In R programming, objects are copied when assigned to new variable or passed to a function (pass by value). For example.

```
> # create list a and assign to b
> a <- list("x" = 1, "y" = 2)
> b <- a
> # modify b
> b$y = 3
> # a remains unaffected
> a
$x
[1] 1
$y
[1] 2
> # only b is modified
> b
$x
[1] 1
$y
[1] 3
```

But this is not the case with reference objects. Only a single copy exist and all variables reference to the same copy. Hence the name, reference.

```
> # create reference object a and assign to b
> a <- student(name = "John", age = 21, GPA = 3.5)
> b <- a
> # modify b
> b$name <- "Paul"
> # a and b both are modified
> a
Reference class object of class "student"
Field "name":
[1] "Paul"
Field "age":
[1] 21
Field "GPA":
[1] 3.5
> b
Reference class object of class "student"
Field "name":
[1] "Paul"
Field "age":
[1] 21
Field "GPA":
[1] 3.5
```

This can cause some unwanted change in values and be the source of strange bugs. We need to keep this in mind while working with reference objects. To make a copy, we can use the `copy()` method made availabe to us.

```
> # create reference object a and assign a's copy to b
> a <- student(name = "John", age = 21, GPA = 3.5)
> b <- a$copy()
> # modify b
> b$name <- "Paul"
> # a remains unaffected
> a
Reference class object of class "student"
Field "name":
[1] "John"
Field "age":
[1] 21
Field "GPA":
[1] 3.5
> # only b is modified
> b
Reference class object of class "student"
Field "name":
[1] "Paul"
Field "age":
[1] 21
Field "GPA":
[1] 3.5
```

Reference Methods

Methods are defined for a reference class and do not belong to generic functions as in S3 and S4 classes.

All reference class have some methods predefined because they all are inherited from the superclass `envRefClass`.

```
> student
Generator for class "student":
Class fields:
Name:       name      age      GPA
Class: character   numeric   numeric
Class Methods:
"callSuper", "copy", "export", "field", "getClass", "getRefClass",
"import", "initFields", "show", "trace", "untrace", "usingMethods"
Reference Superclasses:
"envRefClass"
```

We can see class methods like `copy()`, `field()` and `show()` in the above list. We can create our own methods for the class.

This can be done during the class definition by passing a list of function definitions to `methods` argument of `setRefClass()`.

```
student <- setRefClass("student",
fields = list(name = "character", age = "numeric", GPA = "numeric"),
methods = list(
inc_age = function(x) {
age <<- age + x
},
dec_age = function(x) {
age <<- age - x
}
)
)
```

In the above section of our code, we defined two methods called `inc_age()` and `dec_age()`. These two method modify the field `age`.

Note that we have to use the non-local assignment operator `<<-` since age isn't in the method's local environment. This is important.

Using the simple assignment operator `<-` would have created a local variable called `age`, which is not what we want. R will issue a warning in such case.

Here is a sample run where we use the above defined methods.

```
> s <- student(name = "John", age = 21, GPA = 3.5)
> s$inc_age(5)
> s$age
[1] 26
> s$dec_age(10)
> s$age
[1] 16
```