R Inheritance - DataMentor

R Inheritance - DataMentor

In this article, you'll learn everything about inheritance in R. More specifically, how to create inheritance in S3, S4 and Reference classes, and use them efficiently in your program.

Inheritance is one of the key features of object-oriented programming which allows us to define a new

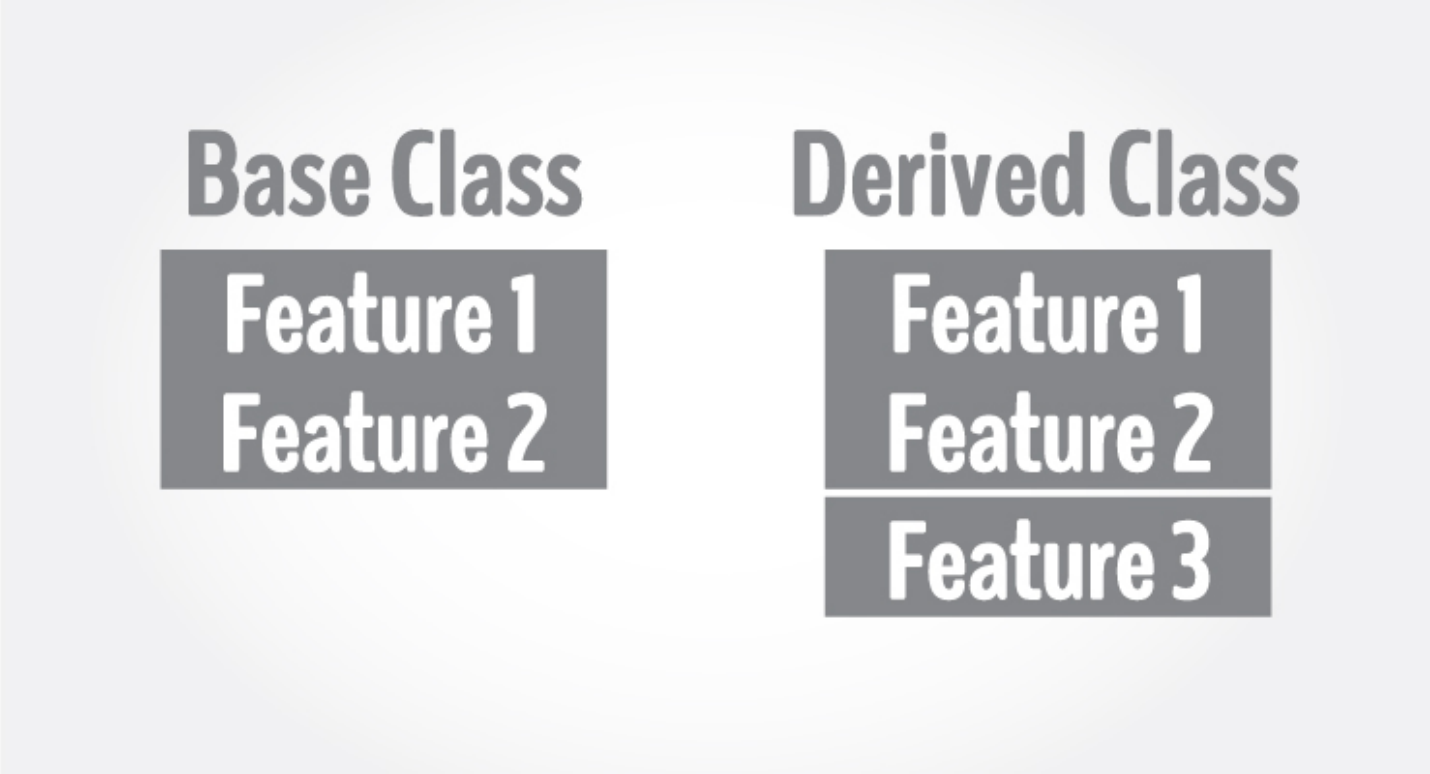
class

out of existing classes.

This is to say, we can derive new classes from existing base classes and adding new features. We don't have to write from scratch. Hence, inheritance provides reusability of code.

Inheritance forms a hierarchy of class just like a family tree. Important thing to note is that the attributes define for a base class will automatically be present in the derived class.

Moreover, the methods for the base class will work for the derived.



Below, we discuss how inheritance is carried out for the three different class systems in R programming language.

Inheritance in S3 Class

S3 classes do not have any fixed definition. Hence attributes of S3 objects can be arbitrary.

Derived class, however, inherit the methods defined for base class. Let us suppose we have a function that creates new objects of class `student` as follows.

```
student <- function(n,a,g) {
value <- list(name=n, age=a, GPA=g)
attr(value, "class") <- "student"
value
}
```

Furthermore, we have a method defined for generic function `print()` as follows.

```
print.student <- function(obj) {
cat(obj$name, "\n")
cat(obj$age, "years old\n")
cat("GPA:", obj$GPA, "\n")
}
```

Now we want to create an object of class `InternationalStudent` which inherits from `student`.

This is be done by assigning a character vector of class names like `class(obj) <- c(child, parent)`.

```
> # create a list
> s <- list(name="John", age=21, GPA=3.5, country="France")
> # make it of the class InternationalStudent which is derived from the class student
> class(s) <- c("InternationalStudent","student")
> # print it out
> s
John
21 years old
GPA: 3.5
```

We can see above that, since we haven't defined any method of the form `print.InternationalStudent()`, the method `print.student()` got called. This method of class `student` was inherited.

Now let us define `print.InternationalStudent()`.

```
print.InternationalStudent <- function(obj) {
cat(obj$name, "is from", obj$country, "\n")
}
```

This will overwrite the method defined for class `student` as shown below.

```
> s
John is from France
```

We can check for inheritance with functions like `inherits()` or `is()`.

```
> inherits(s,"student")
[1] TRUE
> is(s,"student")
[1] TRUE
```

Inheritance in S4 Class

Since S4 classes have proper definition, derived classes will inherit both attributes and methods of the parent class.

Let us define a class `student` with a method for the generic function `show()`.

```
# define a class called student
setClass("student",
slots=list(name="character", age="numeric", GPA="numeric")
)
# define class method for the show() generic function
setMethod("show",
"student",
function(object) {
cat(object@name, "\n")
cat(object@age, "years old\n")
cat("GPA:", object@GPA, "\n")
}
)
```

Inheritance is done during the derived class definition with the argument `contains` as shown below.

```
# inherit from student
setClass("InternationalStudent",
slots=list(country="character"),
contains="student"
)
```

Here we have added an attribute `country`, rest will be inherited from the parent.

```
> s <- new("InternationalStudent",name="John", age=21, GPA=3.5, country="France")
> show(s)
John
21 years old
GPA: 3.5
```

We see that method define for class `student` got called when we did `show(s)`.

We can define methods for the derived class which will overwrite methods of the base class, like in the case of S3 systems.

Inheritance in Reference Class

Inheritance in reference class is very much similar to that of the S4 class. We define in the `contains` argument, from which base class to derive from.

Here is an example of `student` reference class with two methods `inc_age()` and `dec_age()`.

```
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
}
)
)
```

Now we will inherit from this class. We also overwrite `dec_age()` method to add an integrity check to make sure `age` is never negative.

Let us put it to test.

```
> s <- InternationalStudent(name="John", age=21, GPA=3.5, country="France")
> s$dec_age(5)
> s$age
[1] 16
> s$dec_age(20)
Error in s$dec_age(20) : Age cannot be negative
> s$age
[1] 16
```

In this way, we are able to inherit from the parent class.