Module # 7 R Object: S3 vs. S4

**1.How do you tell what OO system (S3 vs. S4) an object is associated with?**

Object Oriented system **(S3 vs. S4)**  an object is associated with, can be determined by using following functions **‘is’** and **‘methods’.**

methods(class=class(x))

It will result in methods linked with the object’s class. If the object is S3 class, it will result in names of the function without class. But, if it S4 class it will result in function names which are defined with functions ‘setMethod’ or ‘setGeneric’.

In addition, function ‘isS4(x)’ can be used to determine if the object is S4 or not. If it results in TRUE, then the object is S4 class or if it results in FALSE, then the object is S3 class or other system.

**2. How do you determine the base type (like integer or list) of an object?**

The base type of an object can be determined by using the following function ‘class’, which when implemented will result in its base type like integer or list.

class(x)

If the resultant is ‘list’, then the object is a list. If the resultant is ‘integer’, then the object is integer.

> s <- list(name = "Myself", age = 29, GPA = 3.5)

> class(s)

[1] "list"

Sometimes, the object can have more than one class, in such a case, it will result vector of character strings.

**3. What is a generic function?**

A generic function is a kind of function used in object-oriented programming that performs differently based on the class or type of the object to which it is utilized. This can use different types of objects, as opposed to being limited to a particular data type, which is why it is named "generic." Methods in S3 and S4 object systems are part of generic functions rather than classes. In fact in S3, ‘print’ function is generic and ‘print.default’ is a method of generic function.

With the help of the ‘setGeneric’ function in S4, it's indeed feasible to develop functions that will have various interpretations based on the class of the object to which they are implemented. The ‘setMethod’ function can be utilized to create alternatives for the generic function.

**4. What are the main differences between S3 and S4?**

**S3 classes:**

Most common class used. Can be applied to multiple data types IE Plot () print () summary (). Can only dispatch on the first argument. Allows you to easily change the class of existing objects (Bad). S3 class has no formal, predefined definition. Basically, a list with its class attribute set to some class name, is an S3 object. The components of the list become the member variables of the object. Assigning a character string to an object's class attribute defines a class in S3, whereas ‘setClass’ function is required in S4. A class can only inherit from another class in S3. Alternatively, S4 supports multiple inheritances, allowing a single category to inherit from multiple others. A class attribute is used to dispatch methods in S3 based on its class attribute. Generic functions are called by R by looking for methods with matching names and class attributes. A subclass of the object's class will be sought if no method is found with a class attribute. Based on the inheritance hierarchy of the classes involved, method dispatch in S4 is more complex and flexible.

**S4 classes:**

Safer. More rigorous error checking. Can dispatch on multiple argument. Does not allow you to easily change the class of existing objects (Good).

|  |  |  |
| --- | --- | --- |
| **Operation** | **S3** | **S4** |
| Define class | Implicit in constructor code | setClass() |
| Create object | Build list, set class attr | new() |
| Reference member variable | **$** | **@** |
| Implement generic f() | Define f.classname() | setMethod() |
| Declare generic | UseMethod() | setGeneric() |

**Examples of S3:**

**Ex1:**

> #list named ‘happy’ is created with three elements

> happy <- list(first="one", second="two", third="third")

> #new class ‘monday’ is added to the existing class attribute of the list

> class(happy) <- append(class(happy),"monday")

> happy

$first

[1] "one"

$second

[1] "two"

$third

[1] "third"

attr(,"class")

[1] "list" "monday"

> #function ‘GetFirst’ is defined and it uses ‘UseMethod’ function

> GetFirst <- function(x)

+ { UseMethod("GetFirst",x) }

> #method of ‘GetFirst’ for object with class ‘monday’ is defined and results in first element

> GetFirst.monday <- function(x)

+ { return(x$first) }

> #function ‘GetFirst’ is implemented on list ‘happy’

> GetFirst(happy)

[1] "one"

**Ex2:**

# list named ‘student’ is created with three elements

> student <- list(Name = "Happy", Age = 21, Location="Tampa")

#class ‘studentdata’ is added to the list ‘student’

> class(student) <- "Studentdata"

> student

$Name

[1] "Happy"

$Age

[1] 21

$Location

[1] "Tampa"

attr(,"class")

[1] "Studentdata"

#function ‘location’ is defined and uses ‘UseMethod’ function

> location<- function(object) {

+ UseMethod("location")

+ }

#method of ‘location’ for object with class ‘Studentdata’ is defined and function ‘cat’ is implemented to print the message which is saved in ‘location’

> location.Studentdata <- function(object) {

+ cat("My location is", object$Location, "\n")

+ }

# function ‘location’ is implemented on list ‘student’ and this prints the message.

> location(student)

My location is Tampa

**Examples of S4:**

**Ex1:**

> # S4 class with a slot is defined

> setClass("student", slots = c(name = "character", age = "numeric"))

> # object of the class is created

> mestudent<- new("student", name = "happy", age = 26)

> # generic method is defined

> setGeneric("myMethod", function(object, ...) {

+ standardGeneric("myMethod")

+ })

[1] "myMethod"

> # method for the class is defined

> setMethod("myMethod", signature("student"), function(object, ...) {

+ print(paste("Name:", object@name))

+ print(paste("Age:", object@age))

+ })

> # generic method on the object is implemented

> myMethod(mestudent)

[1] "Name: happy"

[1] "Age: 26"

**Ex2:**

> # generic function called "greet" is defined

> setGeneric("greet", function(student) {

+ standardGeneric("greet")

+ })

[1] "greet"

> setClass("student",

+ representation(

+ name = "character",

+ age = "numeric"

+ )

+ )

> # method for the "greet" function for the "student" class is defined

> setMethod("greet", "student", function(student) {

+ paste0("Hello, my name is ", student@name, " and I'm ", student@age, " years old.")

+ })

> # Create instances of the "student" class

> student <- new("student", name = "Happy", age = 21)

> # function "greet" on the instances of the classes is implemented

> greet(student)

[1] "Hello, my name is Happy and I'm 21 years old."

**Dataset'**mtcars**':**

data("mtcars")  
head (mtcars, 6)  
list(mtcars, 6)

✔Generic function cannot be applied directly to the ‘mtcars’ dataset as it is a inbuilt in R and exists as data.frame. Importantly, generic function is defined for classes.

But, the generic function to the "mtcars" dataset can be applied only by defining it as a function that operates on data frames. Despite this, the dataset itself does not have the function assigned to it.

#Generic function ‘car’ is defined

car <- function(df) {

summary(df)

}

# function ‘car’ created is applied to the dataset

car(mtcars)

✔As it is mentioned earlier that ‘mtcars’ dataset is inbuilt in R and exists as data.frame, S3 and S4 cannot be directly applied to it. But S3 or S4 objects can be created that consists of data from ‘mtcars’ and then it can be managed accordingly.

# **S3** object is created from "mtcars"

car <- list(

data = mtcars,

class = "happy"

)

# method for the ‘happy’ class is defined

summary.happy <- function(object) {

summary(object$data)

}

# "summary" method on the ‘car’ object is implemented

summary(car)

# **S4** class "happy" is defined

setClass("happy", slots = c(data = "data.frame"))

# S4 object ‘car’ from "mtcars" is created

car <- new("happy", data = mtcars)

# method for "happy" class is defined

summary.happy <- function(object) {

summary(object@data)

}

# "summary" method on the "car" object is implemented

summary(car)

**References:**

Venables,W.N., Smith,D.M., & R core team.(2022).*An Introduction to R*(Version 4.2.2).URL: [https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf](https://www.blogger.com/blog/post/edit/2482057455959808426/3452863189251029109)

Matloff. N. *The Art of R programming*(*Object Oriented Programming)*. Page number: 207-226