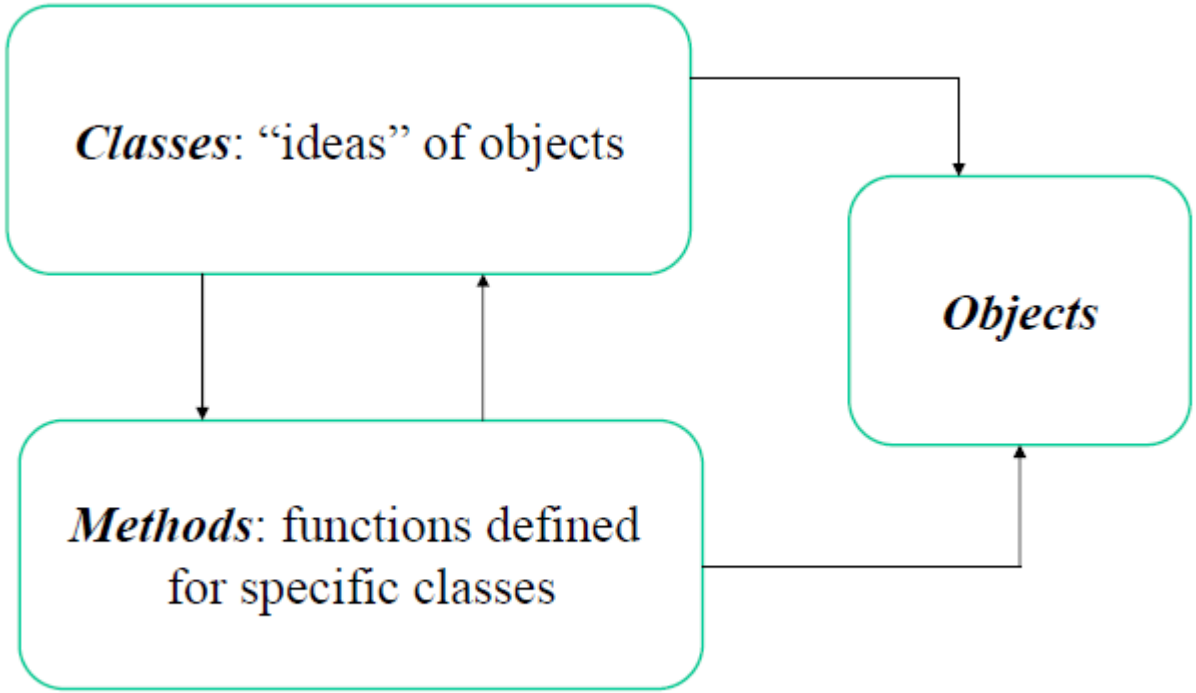
Object Oriented Programming

$\tau\rho$

What you can do with R OO-programming

- You can *easily define complex data structures*, starting from simpler ones
- You can define methods that behave differently according to the objects they are applied to
- *Modelling real problems is simpler*

OO programming in R: classes and methods



Object

- → a thing that has state, behaviour, and identity
  - state : all of the properties of the object
  - behaviour : how an object acts and reacts from the change of state and interactions with other objects
  - identity :unique property of an object

Classes

- In R all software entities are $objects$
- Each $object$ belong to a $class$
- A $class$ is a general scheme for $objects$:
  - From a general standpoint it represents an "$idea$“ (i.e general structure of a given object)
- $Objects$ are realizations or instances of a $class$

Classes in R

- First we will take a look at S3 classes. Base R uses S3 more or less exclusively
- The greatest use of object oriented programming in R is through print methods, summary methods and plot methods.
- An S3 class is (most often) a list with a class attribute. It is constructed by the following code

```
class(obj) <- "class.name"
```

Example 1

Constructing new S3 classes

- Create an $object$ of $class$ "human" that consist the height, weight, & name.
  - Height : $2.54 \times 12 \times 6/100$
  - Weight : $180/2.2$
  - Name : James

Answer 1

```
jim <- list(height = 2.54 * 12 * 6/100, weight =
180/2.2,name = "James")
class(jim)="human"
class(jim)
jim
```

back

Example 2

- Create an $object$ of $class$ "colour" that consist the Cyan Component, Magenta Component, Yellow Component, Black Component, Name, & Type of colour.
  - CyanComp : 0
  - MagentaComp : 0
  - YellowComp : 0
  - BlackComp :100
  - Name : Black
  - Type :CMYK

Answer 2

```
col<- list (CyanComp=0, MagentaComp=0, YellowComp=0,
BlackComp=100, Colour="Black",
Type="CMYK")
class(col)="colour"
col
```

back

Example 3

Constructing new S4 classes

- A *class* representing codons (triplets of nucleotides)

```
setClass("triplets", representation(x="character"),
                     prototype(x="UAG"))
```

- Construction of objects of the “triplets” class

```
seq0 <- new("triplets"); # prototype is called
seq <- new("triplets",
x = c("AUG","CCA","CCA","GAA","UGA","CCA"));
typeof(seq)
```

back

Example 4

- Assume we have a simple *class* with two slots below

```
track <- setClass("track",slots = c(x="numeric",
                                    y="numeric"))
```

with an *object* from the *class*

```
t1 <- track(x = 1:10, y = 1:10 + rnorm(10))
```

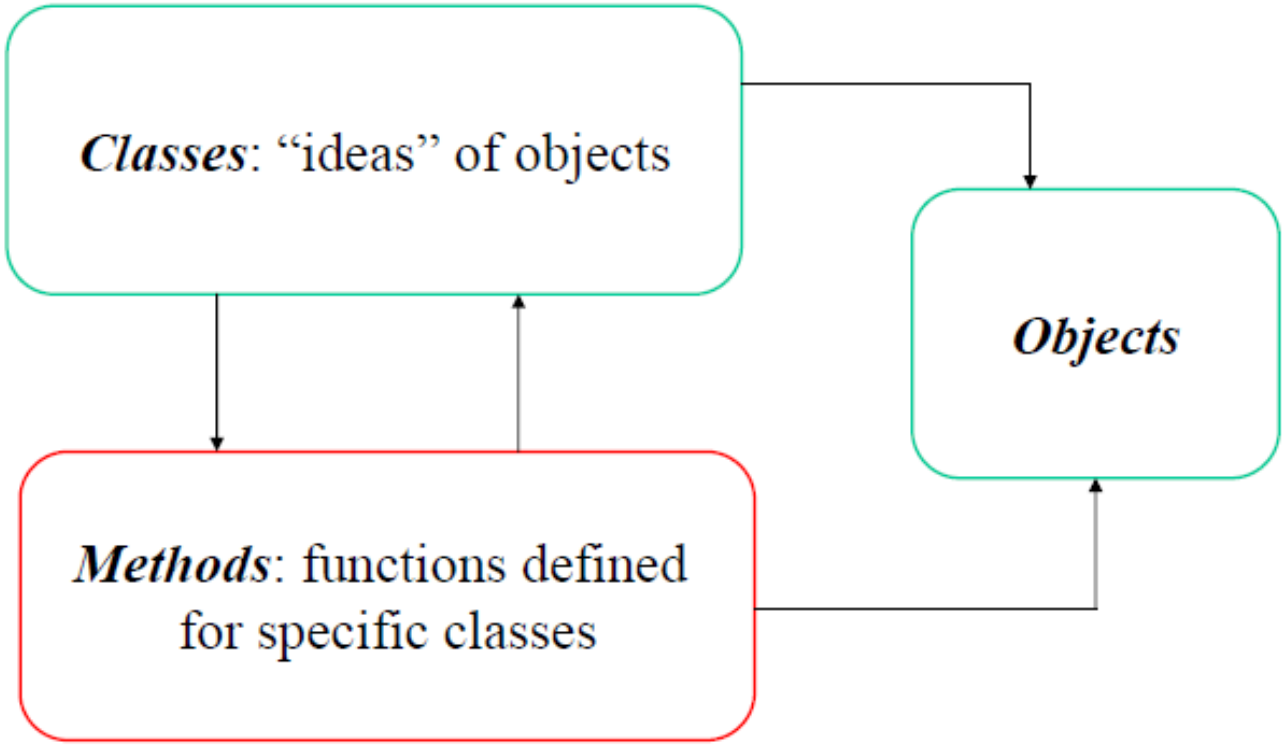
Create a new *class* `"trackCurve"` by adding one more slot from the *class* before

Answer 4

```
trackCurve <- setClass("trackCurve",
    slots = c(smooth = "numeric"),
    contains = "track")

t1s <- trackCurve(t1, smooth = 1:10)
```

OO programming in R: classes and methods



Methods

- Can we make general functions, that behave differently with different $objects$?

- In R we need:

  1. A *generic function*: *what* we want to do ?
  2. A *method*: *how* we do it with an $object$ of a specific class ?

Example 5

- From Example 1, difine a $method$ for printing the “human” $class$

Answer 5

```
jim <- list(height = 2.54 * 12 * 6/100, weight =
180/2.2,name = "James")
class(jim)="human"

print.human <- function(x, ...) {
cat("name:", x$name, "\n")
cat("height:", x$height, "meters", "\n")
cat("weight:", x$weight, "kilograms", "\n")
}
print(jim)
```

Example 6

- From Example 2, difine a $method$ for printing the characteristics of “colour” $class$

Answer 6

```
col<- list (CyanComp=0, MagentaComp=0, YellowComp=0,
BlackComp=100, Colour="Black",
Type="CMYK")
class(col)="colour"

print.colour <- function(obj, ...) {
cat("Colour Type:", obj$Type, "\n")
cat("Colour Name:", obj$Colour, "\n")
matrix(c(obj$CyanComp, obj$MagentaComp, obj$YellowComp,
obj$BlackComp),nc=1,
dimnames=list(c("C","M","Y","K"),c("Value")))
}
print(col)
```

Example 7

- From Example 3, we need a method to compute the number of codons that are present in a given object of $class$ "triplets".

Answer 7

```
setMethod("length", "triplets",
              function(object) {
                     l <- length(object@x);
                     return(l);
              }
          )

seq <- new("triplets",
x = c("AUG","CCA","GGA","UGA","CCA"));
length(seq)
```

Thank you ☺