Introduction to OOP

Criscely Luján

December 1, 2020

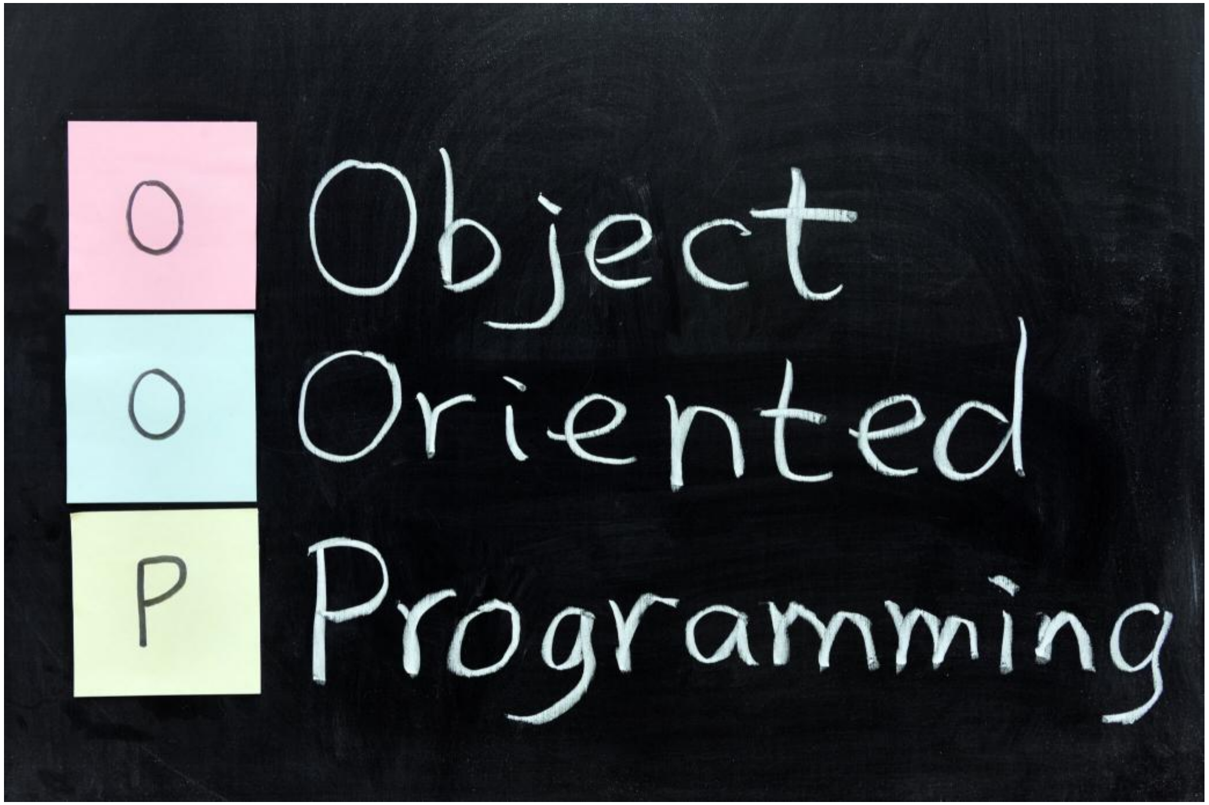
To keep in touch !

Personal email: criscelylujan@gmail.com
Profesional email: criscely.lujan@ird.fr
Twitter: @CriscelyLP
GitHub: @CriscelyLP

LET'S GET STARTED!

OOP!

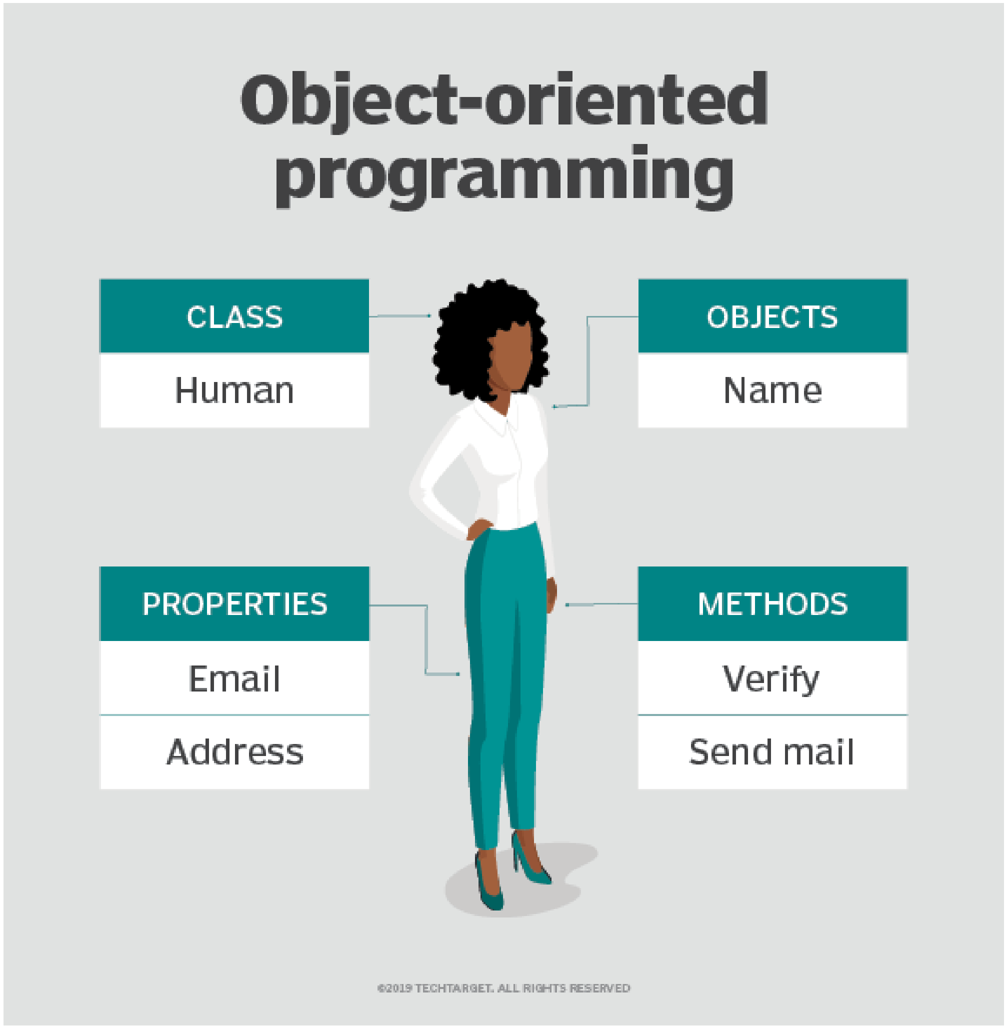
OOP?



www.roberthalf.com

In a OOP system ...

- Programmers define a data type of a data structure (`object`),
- Types of operations (`methods`) that can be applied to the data structure.
- All objects with the same attributes and behavior are grouped into collections (`classes`)



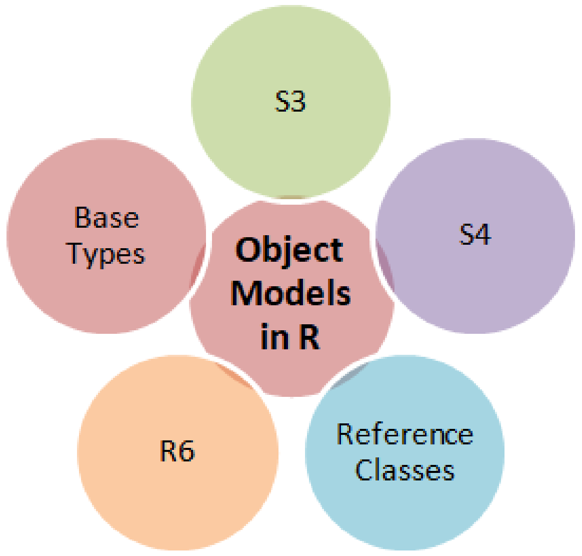
Techtarget

OOP representation

OOP in R

- `R` has four OOP implementations: S3, S4, RC (reference classes) and R6.
- `Base type` system is not a OOP !

OOP in R



- S3 is the `simplest OOP` system in R
- Most `commonly` used system in CRAN packages
- The only OOP system used in the `base` and `stats` packages

(Wickham, 2014)

Recognising objects

- An easy way to recognise objects is by the use of the package `pryr`.

```
library(pryr)
```

```
# We create a data as an example:
data = data.frame(x = 1:10, y = letters[1:10])
str(data)
```

```
## 'data.frame':    10 obs. of  2 variables:
##  $ x: int  1 2 3 4 5 6 7 8 9 10
##  $ y: Factor w/ 10 levels "a","b","c","d",..: 1 2 3 4 5 6 7 8 9 10
```

Recognising objects

- The `otype` function determine the object type

```r
otype(data) # The data frame is an S3 class
```

```
## [1] "S3"
```

```r
otype(data$x) # A vector is not S3
```

```
## [1] "base"
```

```r
otype(data$y) # A factor is S3
```

```
## [1] "S3"
```

Exploring classes

- By the use of the function `class`

```
class(data)
```

```
## [1] "data.frame"
```

```
class(data$x)
```

```
## [1] "integer"
```

```
class(data$y)
```

```
## [1] "factor"
```

Using generic functions

- In S3 methods belong to functions called `generic functions`.
- To determine if a function is a S3 generic, you can inspect its source code for a call to `UseMethod()`.
- `ftype` also describe the function type.

```r
mean
```

```
## function (x, ...)
## UseMethod("mean")
## <bytecode: 0x5640e8925eb0>
## <environment: namespace:base>
```

```r
ftype(mean) # S3 generic function for the arithmetic mean
```

```
## [1] "s3"      "generic"
```

Calling methods!

- Given a class, the job of an S3 generics is to call the right `S3 method`
- You can see all the methods that belong to a generic with `methods()`

```
methods("mean")
```

```
## [1] mean.Date     mean.default  mean.difftime mean.POSIXct  mean.POSIXlt 
## [6] mean.quosure*
## see '?methods' for accessing help and source code
```

```
methods("plot")
```

```
##  [1] plot.acf*           plot.data.frame*    plot.decomposed.ts*
##  [4] plot.default        plot.dendrogram*    plot.density*      
##  [7] plot.ecdf           plot.factor*        plot.formula*      
## [10] plot.function       plot.hclust*        plot.histogram*    
## [13] plot.HoltWinters*   plot.isoreg*        plot.lm*           
## [16] plot.medpolish*     plot.mlm*           plot.ppr*          
## [19] plot.prcomp*        plot.princomp*      plot.profile.nls*  
## [22] plot.raster*        plot.spec*          plot.stepfun       
## [25] plot.stl*           plot.table*         plot.ts            
## [28] plot.tskernel*      plot.TukeyHSD*     
## see '?methods' for accessing help and source code
```

```r
methods("print") # Needing more space ! :)
```

```
##   [1] print.acf*
##   [2] print.AES*
##   [3] print.anova*
##   [4] print.aov*
##   [5] print.aovlist*
##   [6] print.ar*
##   [7] print.Arima*
##   [8] print.arima0*
##   [9] print.AsIs
##  [10] print.aspell*
##  [11] print.aspell_inspect_context*
##  [12] print.bibentry*
##  [13] print.Bibtex*
##  [14] print.browseVignettes*
##  [15] print.by
##  [16] print.bytes*
##  [17] print.changedFiles*
##  [18] print.check_code_usage_in_package*
##  [19] print.check_compiled_code*
##  [20] print.check_demo_index*
##  [21] print.check_depdef*
##  [22] print.check_details*
##  [23] print.check_details_changes*
##  [24] print.check_doi_db*
##  [25] print.check_dotInternal*
##  [26] print.check_make_vars*
##  [27] print.check_nonAPI_calls*
##  [28] print.check_package_code_assign_to_globalenv*
##  [29] print.check_package_code_attach*
##  [30] print.check_package_code_data_into_globalenv*
##  [31] print.check_package_code_startup_functions*
##  [32] print.check_package_code_syntax*
```

… Almost ready!

Some concluding remark

- OOP is important for all R users.
- If you use R, for SURE you were using classes, methods, and generics! (`OOP system`)
- Not all R packages use OOP system! :(
- Always there are new things to learn!
- Are you creating a new package? Let's use this time OOP! 💜 ...

... this will make life easier for users !

SOY UNA
HUGGER

Thanks!

- R-Ladies Freiburg for the invitation.
- R-Ladies Montpellier for the co-organization.

Bibliography

- A. C. Kak. Programming with Objects: A Comparative Presentation of Object-Oriented Programming With C++ and Java. John Wiley and Sons, Inc, 2003.
- J. M. Garrido. Object-oriented programming: from problem solving to Java. Charles River Media, 1 edition, 2003.
- I. D. Craig. Object-oriented programming languages: Interpretation. 8, 2007
- H. Wickham. Advanced R. Chapman and Hall/CRC, first edition edition, 2014. ISBN13: 978-1466586963.
- H. Wickham. Advanced R. Chapman and Hall/CRC, second edition edition, 2019. ISBN13: 978-0815384571.