Object Oriented Programming in R

An Introduction to S3

Lukas Huwiler

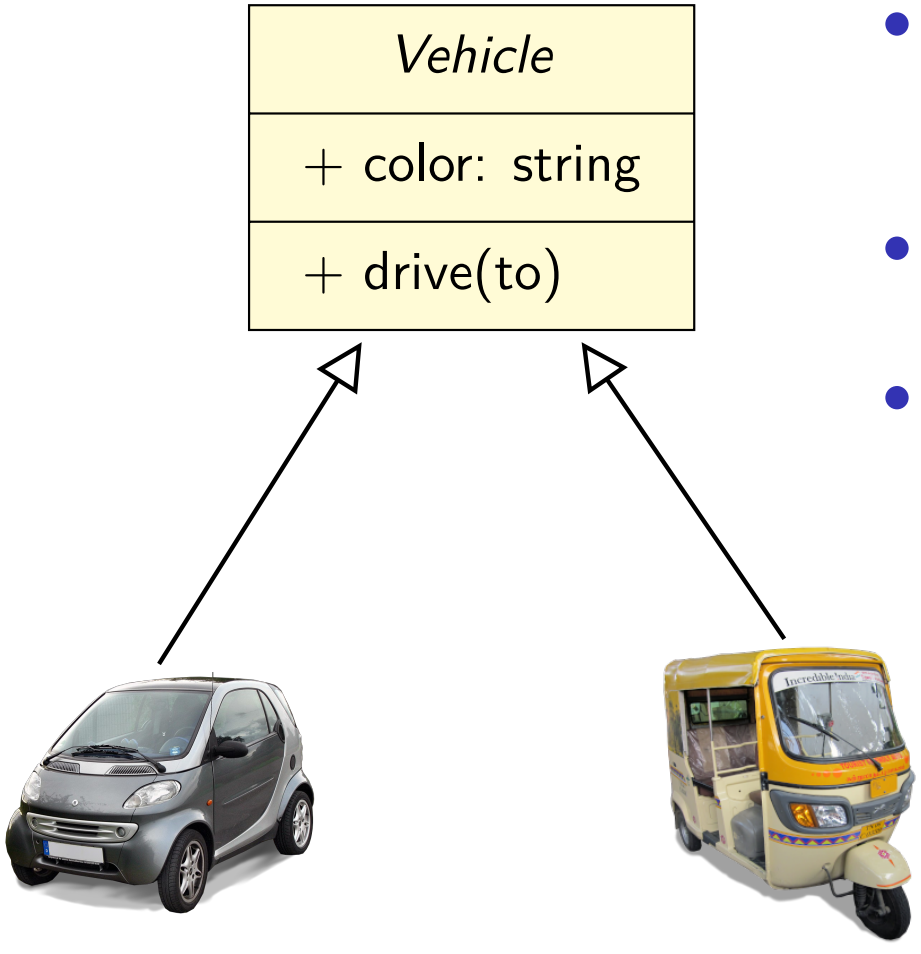
November 25, 2020

A *Very* Short Introduction to Object Oriented Programming

Speaking Object Oriented

- It's a car $\rightarrow$ It's an *instance* or *object* of *class* car
- It's a *grey* car $\rightarrow$ The class car has an *attribute* `color` (data)
- A car can *drive* $\rightarrow$ The class car has a *method* `drive` (behaviour)

Class Hierarchies



- Cars and rickshaws are vehicles $\rightarrow$ Car and Rickshaw *inherit* from Vehicle
- Car and Rickshaw are child classes of Vehicle
- Vehicle is the parent class of Car and Rickshaw

OOP in R

It's a Mess!

R has three (or even four) different class systems:

- **S3:** oldest class system; very simple but elegant; comes with some limitations
- **S4:** more formal; adds multiple dispatch and type checks
- **R5:** Reference classes; feels similar to C++, Java, or Python; reference semantics
- **(R6):** the unofficial (but better?) implementation of R5

I will focus on S3 in this talk...

The S3 Class System

Have You Ever Wondered?

```
summary(car_model)
```

```
##
## Call:
## lm(formula = mpg ~ hp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7121 -2.1122 -0.8854  1.5819  8.2360
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
## hp          -0.06823    0.01012  -6.742 1.79e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.863 on 30 degrees of freedom
## Multiple R-squared:  0.6024, Adjusted R-squared:  0.5892
## F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
```

```
summary("It's all about cars")
```

```
##    Length     Class      Mode
##         1 character character
```

```
summary(mtcars[1:2])
```

```
##       mpg             cyl
##  Min.   :10.40   Min.   :4.000
##  1st Qu.:15.43   1st Qu.:4.000
##  Median :19.20   Median :6.000
##  Mean   :20.09   Mean   :6.188
##  3rd Qu.:22.80   3rd Qu.:8.000
##  Max.   :33.90   Max.   :8.000
```

```
summary(table(mtcars$cyl))
```

```
## Number of cases in table: 32
## Number of factors: 1
```

```
summary(as.factor(mtcars$cyl))
```

```
##  4  6  8
## 11  7 14
```

The mechanism why summary works for (almost) everything is called *method dispatch*

How Method Dispatch Works

```
class(mtcars)
```

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

```
summary
```

```
## function (object, ...)
## UseMethod("summary")
## <bytecode: 0x5607e5cf85f0>
## <environment: namespace:base>
```

```
methods(summary)[1:16]
```

```
##  [1] "summary.aov"                "summary.aovlist"
##  [3] "summary.aspell"             "summary.check_packages_in_dir"
##  [5] "summary.connection"         "summary.data.frame"
##  [7] "summary.Date"               "summary.default"
##  [9] "summary.ecdf"               "summary.factor"
## [11] "summary.glm"                "summary.infl"
## [13] "summary.lm"                 "summary.loess"
## [15] "summary.manova"             "summary.matrix"
```

Create Your Own Generic Method

```r
# Define methods
drive <- function(x, location) UseMethod("drive", x)

drive.Car <- function(x, location) {
  cat("drive car...\n")
  x$location <- location
  x$gas <- pmax(x$gas - 10, 0)
  x
}

drive.Rickshaw <- function(x, location) {
  cat("drive rickshaw...\n")
  x$location <- location
  x
}

# Instantiate objects
car <- structure(list(color = "grey", location = "Zurich", gas = 100),
                 class = "Car")
rickshaw <- list(color = "blue", location = "Munich")
class(rickshaw) <- "Rickshaw"
```

```r
# Call generic methods
car <- drive(car, "Lucerne")
```

```
## drive car...
```

```r
rickshaw <- drive(rickshaw, "Lucerne")
```

```
## drive rickshaw...
```

Class Hierarchies in S3

```r
# Make the objects inheriting from Vehicle (don't show this to a Java developer)
class(car) <- c("Car", "Vehicle")
class(rickshaw) <- c("Rickshaw", "Vehicle")
rm(drive.Rickshaw)

drive.Vehicle <- function(x, location) {
  cat("Change location...\n")
  x$location <- location
  x
}

drive.Car <- function(x, location) {
  x <- NextMethod()
  cat("Consume gasoline...\n")
  x$gas <- pmax(x$gas - 10, 0)
  x
}
```

```r
car <- drive(car, "Lucerne")
```

```
## Change location...
## Consume gasoline...
```

```r
rickshaw <- drive(rickshaw, "Lucerne")
```

```
## Change location...
```

Some Additional Points

```r
# You can define a default method as a fallback option:
drive.default <- function(x, location) {
  sprintf("An instance of class %s cannot drive...", class(x))
}

drive(2, "Lucerne")
```

```
## [1] "An instance of class numeric cannot drive..."
```

```r
# The package sloop offers some nice functions for a better understanding:
sloop::s3_dispatch(drive(car, "Zurich"))
```

```
## => drive.Car
## -> drive.Vehicle
##  * drive.default
```

```r
# It's a good idea to write constructors for your S3 class:
Car <- function(location, color = "black", gas_level = 100) {
  car <- list(color = color, location = location, gas = gas)
  class(car) <- c("Car", "Vehicle")
  car
}
```

Questions?