R Cheat Sheet: OOP and S3 Classes

What is object oriented programming?

While definitions for OOP abound without clear agreement, OOP languages typically focus programmers on the actors/objects (nouns) of a problem rather than the actions/procedures (verbs), by using a common set of language features, including:

1) Encapsulation of data and code – the data and the code that manages that data are kept together by the language (in classes, modules or clusters, etc.) Implicitly, this includes the notion of class definitions and class instances.
2) Information hiding – an exposed API with a hidden implementation of code and data; encourages programming by contract
3) Abstraction and inheritance – so that similarities and differences in the underlying model/data/code/logic for related objects can be grouped & reused
4) Dynamic dispatch – more than one method with the same name – where the method used is selected at compile or run-time by the class of the object and also the class of the method parameter types and their arity (argument number).

Note: R is a functional programing language (FPL). Typically FPLs are a better approach than OOP for the scientific analysis of large data sets. Nonetheless, over time, some OOP features have been added to R.

Four R mechanisms with some OOP features

1) Lexical scoping – simple – encapsulation – mutability - information hiding – BUT not real classes – no inheritance.
2) S3 classes – multiple dispatch on class only - inheritance – BUT just a naming convention – no encapsulation – no information hiding – no control over use – no consistency checks – easy to abuse.
3) S4 formal classes – multiple inheritance – multiple dispatch – inheritance – type checking – BUT no information hiding – verbose and complex to code – lots of new terms – immutable classes only.
4) R5 reference classes – built on S4 - mutable (more like Java, C++) – type checking – multiple inheritance – BUT no information hiding – inconsistent with R's functional programming heritage

Note: None of R's OOP systems are as full featured or as robust as (say) Java or C++. (See table at the bottom of this sheet).

What are S3 classes

```
# An S3 class is any R object to which a
# class attribute has been attached.
```

S3 classes – key functions

```
class(x); class(x) <- 'name' #get/set class
methods('method')         # list S3 methods
UseMethod('method', x)   # generic dispatch
NextMethod()     # inheritance sub-dispatch
```

Class code example

```
c.list <- list(hrs=12, mins=0, diem='am')
class(c.list) <- 'clock' # set the class
class(c.list) # -> "clock"
# Also can be constructed with structure()
# (not recommended), and attr() functions
clock <- structure(list(hrs = 12, mins = 0,
         diem = "am"), .Names = c("hrs",
         "mins", "diem"), class = "clock")
c.list <- unclass(c.list) # remove class
attr(c.list, 'class') <- 'clock' # and back
```

Dynamic dispatch – UseMethod()

```
# the UseMethod for print already exists:
# print <- function(x) UseMethod('print',x)
# So we just need to add a generic method:
print.clock <- function(x) {
    cat(x$hrs);  cat(':');
    cat(sprintf('%02d', x$mins));
    cat(' '); cat(x$diem);  cat('\n')
}
print(c.list) # prints "12:00 am"
# you can find the many S3 print methods:
methods('print') # -> a very long list ...
```

Inheritance dispatch - NextMethod()

```
# S3 classes allow for a limited form of
# class inheritance for the purposes of
# method dispatch. Try the following code:
sound <- function(x) UseMethod('sound', x)
sound.animal  <- function(x) NextMethod()
sound.human   <- function(x) 'conversation'
sound.cat     <- function(x) 'meow'
sound.default <- function(x) 'grunt'
Cathy <- list(legs=4)
class(Cathy) <- c('animal', 'cat')
Harry <- list(legs=2)
class(Harry) <- c('animal', 'human')
Leroy <- list(legs=4)
class(Leroy) <- c('animal', 'llama')
sound(Cathy); sound(Harry); sound(Leroy)
```

Should I use S3 or S4 or R5?

S3: for small/medium projects; S4 for larger; R5 if mutability is necessary

The various OOP features available in R

	Type checking	**Mutable classes**	Encapsulation	Info hiding	Data abstraction	Inheritance	Dynamic dispatch
LS	No	Yes	Yes	Yes	No	No	No
S3	No	No	No	No	No	Yes, clunky	Yes/limited
S4	Yes	No	Yes	No	Yes	Yes	Yes
R5	Yes	Yes	Yes	No	Yes	Yes	Yes

