R Cheat Sheet: OOP and S3 Classes

What is object oriented programming? What are S3 classes

While definitions for OOP abound without # An S3 class is any R object to which a
clear agreement, OOP languages typically # class attribute has been attached.
focus programmers on the actors/objects
(nouns) of a problem rather than the S3 classes – key functions
actions/procedures (verbs), by using a class(x); class(x) <- 'name' #get/set class
common set of language features, including: methods('method') # list S3 methods
1) Encapsulation of data and code – the UseMethod('method', x) # generic dispatch
data and the code that manages that data NextMethod() # inheritance sub-dispatch
are kept together by the language (in
classes, modules or clusters, etc.) Class code example
Implicitly, this includes the notion of c.list <- list(hrs=12, mins=0, diem='am')
class definitions and class instances. class(c.list) <- 'clock' # set the class
2) Information hiding – an exposed API with class(c.list) # -> "clock"
a hidden implementation of code and # Also can be constructed with structure()
data; encourages programming by contract # (not recommended), and attr() functions
3) Abstraction and inheritance – so that clock <- structure(list(hrs = 12, mins = 0,
similarities and differences in the diem = "am"), .Names = c("hrs",
underlying model/data/code/logic for "mins", "diem"), class = "clock")
related objects can be grouped & reused c.list <- unclass(c.list) # remove class
4) Dynamic dispatch – more than one method attr(c.list, 'class') <- 'clock' # and back
with the same name – where the method
used is selected at compile or run-time Dynamic dispatch – UseMethod()
by the class of the object and also the # the UseMethod for print already exists:
class of the method parameter types and # print <- function(x) UseMethod('print',x)
their arity (argument number). # So we just need to add a generic method:
Note: R is a functional programing language print.clock <- function(x) {
(FPL). Typically FPLs are a better approach cat(x$hrs); cat(':');
than OOP for the scientific analysis of cat(sprintf('%02d', x$mins));
large data sets. Nonetheless, over time, cat(' '); cat(x$diem); cat('\n')
some OOP features have been added to R. }
print(c.list) # prints "12:00 am"
Four R mechanisms with some OOP features # you can find the many S3 print methods:
1) Lexical scoping – simple – encapsulation methods('print') # -> a very long list ...
– mutability - information hiding – BUT
not real classes – no inheritance. Inheritance dispatch - NextMethod()
2) S3 classes – multiple dispatch on class # S3 classes allow for a limited form of
only - inheritance – BUT just a naming # class inheritance for the purposes of
convention – no encapsulation – no # method dispatch. Try the following code:
information hiding – no control over use sound <- function(x) UseMethod('sound', x)
– no consistency checks – easy to abuse. sound.animal <- function(x) NextMethod()
3) S4 formal classes – multiple inheritance sound.human <- function(x) 'conversation'
– multiple dispatch – inheritance – type sound.cat <- function(x) 'meow'
checking – BUT no information hiding – sound.default <- function(x) 'grunt'
verbose and complex to code – lots of Cathy <- list(legs=4)
new terms – immutable classes only. class(Cathy) <- c('animal', 'cat')
4) R5 reference classes – built on S4 - Harry <- list(legs=2)
mutable (more like Java, C++) – type class(Harry) <- c('animal', 'human')
checking – multiple inheritance – BUT no Leroy <- list(legs=4)
information hiding – inconsistent with class(Leroy) <- c('animal', 'llama')
R's functional programming heritage sound(Cathy); sound(Harry); sound(Leroy)
Note: None of R's OOP systems are as full
featured or as robust as (say) Java or C++. Should I use S3 or S4 or R5?
(See table at the bottom of this sheet). 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