Reference Classes

Martin Morgan

Fred Hutchinson Cancer Research Center

17-18 November, 2010

R Memory and Class Systems

Memory model

- Everything in memory
- Illusion of copy-on-change

Class systems

S3 Instance-based; single inheritance, single dispatch. Common use case: create once (?).

S4 Class definition; multiple inheritance, multiple dispatch. Updateable, e.g., `phenoData(x)[["foo"]] <- <...>`

S4 challenges

- Novel programming concepts.
- Poor memory management.
- Incomplete integration with *R* (recent example: `$<-`)

Reference Classes

- Introduced in *R*-2.12.
- Suitable when all code should see the same object, e.g., internet resource, GUI window, read-only data base.
- Single inheritance, single dispatch, fields and methods – Java-like; methods associated with classes, rather than generics.
- Unique *R* implementation.

Example: *Stream*

```
> setOldClass(c("file", "connection"))
> ## Define base class; .Stream is a `generator'
> .Stream <- setRefClass("Stream",
+   fields = list(
+     .con = "file", ## convention; no 'private' fields
+     fileName = "character", chunk = "integer"),
+   methods = list(
+     open = function() {
+       "open connection for reading"
+       ## Documentation: Stream$help("open")
+       .self$.con <- file(fileName, "r")
+       invisible(.self)
+     }, close = function() {
+       "close connection"
+       base::close(.con)
+     }))
```

Using *Stream*

```
> fl <- system.file("extdata", "s_1_sequence.txt",
+                    package="Biostrings")
> strm <- .Stream$new(fileName=fl, chunk=100L)
> strm$chunk

[1] 100

> strm$open()
> strm$close()
> try(strm$close())     ## already closed!
> strm$open()$close()   ## use return value of open()
```

Example: *ReadLinesStream*

```
> .ReadLinesStream <- setRefClass("ReadLinesStream",
+   contains = "Stream",
+   methods=list(
+     stream = function() readLines(.con, chunk)))
> ##
> ## Use
> strm <- .ReadLinesStream$new(fileName=fl, chunk=500L)
> strm$open()
> while (length(res <- strm$stream()))
+   cat("length = ", length(res), "\n")

length =  500
length =  500
length =  24

> strm$close()
```

ReadLinesStream user interface I

- Provide user with convenient and familiar interface
- Document with established conventions

```
> ReadLinesStream <- function(fileName, chunk=256L)
+ {
+     .ReadLinesStream$new(fileName=fileName,
+                          chunk=chunk)
+ }
> setMethod("show", "ReadLinesStream", function(object)
+ {
+     cat("class:", class(object), "\n")
+     cat("fileName:", basename(object$fileName), "\n")
+ })
```

ReadLinesStream user interface II

```
> setGeneric("open")
> setMethod(open, "Stream",
+     function(con, ...) con$open())
> setGeneric("close")
> setMethod(close, "Stream",
+     function(con, ...) con$close())
> setGeneric("stream",
+     function(object, ...) standardGeneric("stream"))
> setMethod("stream", "ReadLinesStream",
+     function(object, ...) object$stream())
```

ReadLinesStream user interface III

```
> strm <- ReadLinesStream(fl)
> strm

class: ReadLinesStream
fileName: s_1_sequence.txt

> open(strm)
> while (length(res <- stream(strm)))
+     cat("length =", length(res), "\n")

length = 256
length = 256
length = 256
length = 256

> close(strm)
```

Example: *FastqSampler* in *ShortRead*

- Import a random sub-sample from a `fastq` file.
- Can be invoked repeatedly, for different sub-samples.

Implementation

- Class hierarchy: *Sampler*, *FastqSampler*.
- Generator `ShortRead:::.FastqSampler`
- Exposed interface: `FastqSampler`, `yield`
- Additional features, e.g., finalizer (close stream on garbage collection)

```
> ?FastqSampler
```

Challenges

- Retaining copy-on-change illusion when appropriate.
- User interface: another set of conventions for access, vs. implementation of additional user interface?
- Method documentation – not leveraged or enforced by *R* documentation system.
- Novel approaches, e.g., generators.

Resources

```
?ReferenceClasses
```