# Programming Assignment 2: makeCacheMatrix() as an Object

In R, the list data type is the basis for the S3 object system, one of two "object" systems in R. The second system is the S4 object system, as referenced in [Forms of the Extract Operator](http://bit.ly/2bzLYTL). Programming Assignment 2 is based on the S3 object system.

When an object is defined as an S3 object, it includes not only the functions in its list, but also any variables in memory of the function where the list was created. This is what makes it an "object," it contains both behavior (the functions), and state (the variables in the environment).

In this context, the functions in the list are the equivalent of methods in a Java class, and the matrix that was originally passed to makeCacheMatrix() is still available within the environment of the object to which the list was returned.

A subtlety about the S3 model that isn't explained in the R Programming lectures is that S3 objects rely on a "trick" that makes them work. When an R function returns an object that contains functions to its parent environment (as is the case with a call like myMatrix <- makeCacheMatrix(a)), not only does myMatrix have access to the specific functions in its list, but it also retains access to the entire environment defined by makeCacheMatrix(), including the original argument used to start the function.

Why is this the case? myMatrix contains pointers to functions that are within the makeCacheMatrix() environment after the function ends, so these pointers prevent the memory consumed by makeCacheMatrix() from being released by the garbage collector. Therefore, the entire makeCacheMatrix() environment stays in memory, and myMatrix can access its functions as well as any data in that environment that is referenced in its functions.

This is why x (the argument initialized on the original function call) is accessible by subsequent calls to functions onmyMatrix such as myMatrix$get(), and it also explains why the code works without having to explicitly issue myMatrix$set()to set the value of x.

Reference: Software for Data Analysis, Kindle Edition, location 1683.

To illustrate this point with the makeVector() function that is used as the reference example for Assignment 2, notice that the function declaration along with the first line of code provide the same functionality as the set() function.

makeVector <- function(x) {

m <- NULL

set <- function(y) {

x <<- y

m <<- NULL

}

...

}

if R was a more strongly typed language, the function stub in Professor Peng's repository might look like:

cacheSolve <- function(makeCacheMatrix x, ...) {

# return the inverse of x, or calculate & return if cache is empty

}

This type of specification would make it obvious that cacheSolve() requires as its input the type of object that is output bymakeCacheMatrix().

For additional discussion of the R features used in the programming assignment, please review the article [Demystifying MakeVector](https://github.com/lgreski/datasciencectacontent/blob/master/markdown/rprog-breakingDownMakeVector.md).