Linking S4 classes to a common object (Customer, Order, etc.)

Source: http://stackoverflow.com/questions/29779909/how-to-automatically-update-a-slot-of-s4-class-in-r

You can't link across two independent objects, so you need methods that use both. Here is an example with a replacement method:

```
Customer <- setClass("Customer",
                     slots = c(
                         CustomerID = "numeric",
                         Name = "character",
                         OrderHistory = "list"),
                     prototype = list(
                         OrderHistory = list()))

Order <- setClass(Class="Order",
                  slots = c(
                      Description = "character",
                      Cost = "numeric"))

setGeneric("add<-", function(object, value, ...) StandardGeneric("add<-"))
```

```
[1] "add<-"
```

```
setMethod(f = "add<-",
          signature = c("Customer", "Order"),
          definition = function(object, value) {
              object@OrderHistory <- append(object@OrderHistory, value)
              object
  })

setMethod(f = "show",
          signature = "Customer",
          definition = function(object) {
              cat("** Customer #", object@CustomerID, ": ", object@Name, "\n", sep="")
              for(i in object@OrderHistory)
                  cat("\t", i@Description, "\t", i@Cost, "\n", sep="")
  })

firstCustomer <- new("Customer", CustomerID = 1, Name = "test")
add(firstCustomer) <- new("Order", Description = "new iPhone", Cost = 145)
add(firstCustomer) <- new("Order", Description = "macbook", Cost = 999)

firstCustomer
```

```
** Customer #1: test
    new iPhone  145
    macbook 999
```

Customers, items, transaction and business classes

The following doesn't add to @BrodieG's approach, but emphasizes that you probably want to model tables of Customers, Items, etc., rather than individual customers &c. Also, in many cases I think classes are like data base tables, and principles of good data base design probably apply to good class design (again remembering the S4 classes and R's copy-on-change semantics mean that the classes model columns rather than rows as in many other languages).

```
## Customers -- analogous to a data.frame or data base table
setClass(Class = "Customers", slots = c(
    CustomerId = "integer",
    Name = "character"))

## Items -- analogous to a data.frame or data base table
setClass(Class = "Items", slots = c(
    ItemId = "integer",
    Description = "character",
    Cost = "numeric"))

## Transactions -- analogous to a data.frame or data base table
setClass(Class = "Transactions", slots = c(
    TransactionId = "integer",
    CustomerId = "integer",
    ItemId = "integer"))


## Business -- analogous to a data *base*
Business = setClass(Class = "Business", slots = c(
    Customers = "Customers",          # use class `Customers`
    Items = "Items",                  # use class `Items`
    Transactions = "Transactions"))   # use class `Transactions`



# For a little completeness, here's a minimal implementation starting with
# some utility functions for generating sequential IDs and for updating object slots
# .nextid: increases the identifier for any slot
# .update:

.nextid <- function(x, slotName, n = 1L)
    max(0L, slot(x, slotName)) + seq_len(n)

.update <- function(x, ...) {
    args <- list(...)
    for (nm in names(args))
        args[[nm]] <- c(slot(x, nm), args[[nm]])
    do.call("initialize", c(list(x), args))
}



# The following add vectors of customers and items to the business
add_customers <- function(business, customerNames)
{
```

```
    customers <- slot(business, "Customers")
    len <- length(customerNames)
    initialize(business,
               Customers = .update(customers,
                                   CustomerId = .nextid(customers, "CustomerId",
                                                        len),
                                   Name = customerNames))
}

add_items <- function(business, descriptions, costs)
{
    items <- slot(business, "Items")
    len <- length(descriptions)
    initialize(business,
               Items = .update(items,
                               ItemId = .nextid(items, "ItemId", len),
                               Description = descriptions, Cost=costs))
}



.purchase <- function(business, customerId, itemIds)
{
    transactions <- slot(business, "Transactions")
    len <- length(itemIds)
    initialize(business,
               Transactions = .update(transactions,
                                      TransactionId = rep(.nextid(transactions,
                                                                  "TransactionId"),
                                                          len),
                                      CustomerId = rep(customerId, len),
                                      ItemId = itemIds))
}
```

Here's our business in action

```
bus <- Business()
bus <- add_customers(bus, c("Fred", "Barney"))
bus <- add_items(bus, c("Phone", "Tablet"), c(200, 250))
bus <- .purchase(bus, 1L, 1:2)  # Fred buys Phone, Tablet
bus <- .purchase(bus, 2L, 2L)   # Barney buys Tablet
```

and our total sales (we'd want nice accessors for this)

```
sum(bus@Items@Cost[bus@Transactions@ItemId])
```

```
[1] 700
```

```
bus <- Business()
add_customers(bus, c("Fred", "Barney"))
```

```
An object of class "Business"
Slot "Customers":
An object of class "Customers"
Slot "CustomerId":
[1] 1 2
```

```
Slot "Name":
[1] "Fred"   "Barney"


Slot "Items":
An object of class "Items"
Slot "ItemId":
integer(0)

Slot "Description":
character(0)

Slot "Cost":
numeric(0)


Slot "Transactions":
An object of class "Transactions"
Slot "TransactionId":
integer(0)

Slot "CustomerId":
integer(0)

Slot "ItemId":
integer(0)
```

```
add_items(bus, c("Phone", "Tablet"), c(200, 250))
```

```
An object of class "Business"
Slot "Customers":
An object of class "Customers"
Slot "CustomerId":
integer(0)

Slot "Name":
character(0)


Slot "Items":
An object of class "Items"
Slot "ItemId":
[1] 1 2

Slot "Description":
[1] "Phone"  "Tablet"

Slot "Cost":
[1] 200 250


Slot "Transactions":
An object of class "Transactions"
Slot "TransactionId":
```

```
integer(0)

Slot "CustomerId":
integer(0)

Slot "ItemId":
integer(0)
```

```
.purchase(bus, 1L, 1:2)  # Fred buys Phone, Tablet
```

```
An object of class "Business"
Slot "Customers":
An object of class "Customers"
Slot "CustomerId":
integer(0)

Slot "Name":
character(0)


Slot "Items":
An object of class "Items"
Slot "ItemId":
integer(0)

Slot "Description":
character(0)

Slot "Cost":
numeric(0)


Slot "Transactions":
An object of class "Transactions"
Slot "TransactionId":
[1] 1 1

Slot "CustomerId":
[1] 1 1

Slot "ItemId":
[1] 1 2
```

```
.purchase(bus, 2L, 2L)   # Barney buys Tablet
```

```
An object of class "Business"
Slot "Customers":
An object of class "Customers"
Slot "CustomerId":
integer(0)

Slot "Name":
character(0)
```

```
Slot "Items":
An object of class "Items"
Slot "ItemId":
integer(0)

Slot "Description":
character(0)

Slot "Cost":
numeric(0)


Slot "Transactions":
An object of class "Transactions"
Slot "TransactionId":
[1] 1

Slot "CustomerId":
[1] 2

Slot "ItemId":
[1] 2
```