Bank, CashPoint, ATM, Branch S4 classes

http://rsnippets.blogspot.com/2012/08/an-example-of-oop-in-gnu-r-using-s4.html

The problem

Every commercial bank needs to provide its customers with access to cash via hundreds of cash access points like ATMs or branches. Bank managers facing this problem have to handle three conflicting objectives:

(1) they have to ensure that there is enough cash in cash access points to maintain liquidity,

(2) they want to minimize the amount of cash frozen because it is not working for bank and

(3) they want to minimize transportation costs from central vault to access points.

This is a complex optimization problem which in particular involves the need to predict cash balance in access point every day using historic data.

When designing a solution providing forecasts of cash balances in access points a case for typical application of object oriented approach arises.

For each `cash point` we supply historical data having the same structure and want to obtain a balance prediction. However different access points have different customer usage characteristics and will require different statistical forecasting models. For example in an ATM one can only withdraw the money but in a branch you can as well make a deposit. Therefore we model the bank using S4 classes.

The implementation

There is a `Bank` class that can have many `CashPoints` associated with it. `CashPoint` is a virtual class that has two implementing classes `ATM` and `Branch`. This structure is shown on the figure below.

Each `CashPoint` holds its historical balances and has a givePrediction() method that provides a forecast. This method will be implemented differently in `ATM` and `Branch` classes. The example code implementing this structure is given on listing below.

First, by using a `setGeneric()` function we create a generic function givePrediction() that will dispatch appropriate methods following the class of its arguments

```
setGeneric("givePrediction", function(object) { standardGeneric("givePrediction")})
```

```
[1] "givePrediction"
```

Next we create definitions of Bank, ATM and Branch classes of S4 type using the setClass() function and create a formal method for `givePrediction()` function for those classes.

In our example, for the ATMs we use linear regression and for Branches simple mean as balance predictors. Notice this method defined for `CashPoint` class will be invoked if it will not be overridden by appropriate methods in subclasses (it is not possible to create an object of class CashPoint as it is defined virtual.

The code is run by invoking givePrediction() function on a new Bank class instance.

The bank class

The constructor of Bank class reads the bank structure data from the `bank_model.csv` file that contains the list of cash points with their ids and types (ATM or Branch).

```
bank_model.csv
    CashPointId;CashPointType
    CashPoint_1;ATM
    CashPoint_2;Branch
    CashPoint_3;ATM
    CashPoint_4;Branch
    CashPoint_5;ATM


# Branch or ATM get balance by id
# Example: new("ATM",    "CashPoint_x1")
#          new("Branch", "CashPoint_x2")
```

```
# ensure the files exist
file.exists("inst/extdata/bank_model.csv")
```

```
[1] TRUE
```

```
file.exists("inst/extdata/branch_balances_data.csv")
```

```
[1] TRUE
```

```
# utility functions

read_bank_data <- function() {
  read.table(file = "inst/extdata/bank_model.csv",
                    sep = ";", header = TRUE, stringsAsFactors = FALSE)
}

read_balances <- function() {
  read.table(file = "inst/extdata/branch_balances_data.csv",
                           sep = ";", header = TRUE)
}

cash_point <- function(cp) {
    # cp[2]: ATM or Branch; cp[1]: id
    cat(sprintf("%-8s %-14s \n", cp[2], cp[1]))

    # create class instance on the go.
    new(cp[2], cp[1])
}
```

```
setClass("Bank", slots = c(
             cashPoints = "list"))

# the Bank constructor reads a CSV file with the ATM id's and the
# type of CashPoint
setMethod("initialize", "Bank", function(.Object) {

    # The Bank Model is a dataframe of cash points id and type
    BankModel <- read_bank_data()

    # get cash point instances
    .Object@cashPoints <- apply(BankModel, 1, cash_point)

    # add a name to each list adding cash point type
```

```
    names(.Object@cashPoints) <- apply(BankModel, 1, paste, collapse = "_")
    # Example:
      # "CashPoint_1_ATM"     "CashPoint_2_Branch" "CashPoint_3_ATM"
      # "CashPoint_4_Branch"  "CashPoint_5_ATM"
    return(.Object)
})

setMethod("givePrediction", "Bank", function(object){
    # ocp <<- object@cashPoints
    return(sapply(object@cashPoints, "givePrediction"))
})
```

```
isGeneric("initialize")
```

```
[1] TRUE
```

```
hasMethod("initialize")
```

```
[1] TRUE
```

The cash point

Next, it invokes creation of CashPoint-s. Each cash point is initialized with data from the file `branch_balances_data.csv` which contains three columns: `BranchId`, `Date`, `Balance`. An appropriate subset of data is first selected using the `BranchId` column. Date and Balance are retained in balances field and contain historical data for this cash point.

After the creation of an object of type `Bank`, its `givePrediction()` method is invoked, which calls automatically via the S4 class system either ATM or Branch givePrediction() method according to the run-time type of cash point.

branch_balances_data.csv

```
    BranchId;Date;Balance
    CashPoint_1;2012-12-01;423000
    CashPoint_1;2012-12-02;312000
    CashPoint_1;2012-12-03;220000
    CashPoint_1;2012-12-04;123000
    CashPoint_2;2012-12-01;223000
    CashPoint_2;2012-12-02;212000
    CashPoint_2;2012-12-03;320000
    CashPoint_2;2012-12-04;223000
    CashPoint_3;2012-12-01;323000
    CashPoint_3;2012-12-02;312000
    CashPoint_3;2012-12-03;270000
    CashPoint_3;2012-12-04;223000
    CashPoint_4;2012-12-01;323000
    CashPoint_4;2012-12-02;412000
    CashPoint_4;2012-12-03;320000
    CashPoint_4;2012-12-04;373000
    CashPoint_5;2012-12-01;223000
    CashPoint_5;2012-12-02;192000
    CashPoint_5;2012-12-03;150000
    CashPoint_5;2012-12-04;133000
```

```
# Virtual class
# CashPoint has two slots: id and balances
# cannot be called with new()
setClass("CashPoint", slots = c(
                        id       = "character",
                        balances = "data.frame"),
                      contains = "VIRTUAL")


# CashPoint doesn't do anything; only lends structure to ATM or Branch
setMethod("initialize", "CashPoint", function(.Object, cashPointId) {
    .Object@id <- cashPointId
    balances <- read_balances()  # it has three variables: BranchId, Date and Balance

    .Object@balances <- subset(balances,                      # get balance by id
                               balances$BranchId == .Object@id, -BranchId)
    .Object@balances$Date <- as.Date(.Object@balances$Date) # convert to date

    return(.Object)   # returns two slots: character and a dataframe
})

setMethod("givePrediction", "CashPoint", function(object){
    stop("no givePrediction method for this class")
})
```

The Branch

For Branches we use the simple mean as balance predictor.

```
setClass("Branch", contains = "CashPoint")

setMethod("givePrediction", "Branch", function(object){
    # for Branch only take the mean
    return(mean(object@balances$Balance))
})
```

The ATM

For the ATMs we use linear regression.

```
setClass("ATM", contains = "CashPoint")

setMethod("givePrediction", "ATM", function(object) {
    LM <- lm(Balance ~ as.numeric(Date), data = object@balances)
    prediction <- predict(LM, data.frame(Date = 1 + max(object@balances$Date)))
    return(unname(prediction))
})
```

```
# this starts the prediction process
print(givePrediction(new("Bank")))
```

```
ATM      CashPoint_1
Branch   CashPoint_2
```

```
ATM      CashPoint_3
Branch   CashPoint_4
ATM      CashPoint_5
   CashPoint_1_ATM CashPoint_2_Branch    CashPoint_3_ATM
             21500             244500             196500
CashPoint_4_Branch    CashPoint_5_ATM
            357000              96500
```

sandbox

```
# test how class Bank works
bank <- new("Bank")
```

```
ATM      CashPoint_1
Branch   CashPoint_2
ATM      CashPoint_3
Branch   CashPoint_4
ATM      CashPoint_5
```

```
# ATM      CashPoint_1
# Branch   CashPoint_2
# ATM      CashPoint_3
# Branch   CashPoint_4
# ATM      CashPoint_5

# [1] "CashPoint_1_ATM"    "CashPoint_2_Branch" "CashPoint_3_ATM"    "CashPoint_4_Branch"
# [5] "CashPoint_5_ATM"
```

```
new("ATM", "CashPoint_3")
```

```
An object of class "ATM"
Slot "id":
[1] "CashPoint_3"

Slot "balances":
         Date Balance
9  2012-12-01  323000
10 2012-12-02  312000
11 2012-12-03  270000
12 2012-12-04  223000
```

```
new("Branch", "CashPoint_2")
```

```
An object of class "Branch"
Slot "id":
[1] "CashPoint_2"

Slot "balances":
        Date Balance
5 2012-12-01  223000
6 2012-12-02  212000
7 2012-12-03  320000
8 2012-12-04  223000
```

```r
# weird! CashPoint_2 belongs to a Branch but still shows up.
# we need some validation
new("ATM", "CashPoint_2")
```

```
An object of class "ATM"
Slot "id":
[1] "CashPoint_2"

Slot "balances":
        Date Balance
5 2012-12-01  223000
6 2012-12-02  212000
7 2012-12-03  320000
8 2012-12-04  223000
```

```r
bank_model <- read_bank_data()
bank_model
```

```
  CashPointId CashPointType
1 CashPoint_1           ATM
2 CashPoint_2        Branch
3 CashPoint_3           ATM
4 CashPoint_4        Branch
5 CashPoint_5           ATM
```

```r
balances <- read_balances()
balances
```

```
      BranchId       Date Balance
1  CashPoint_1 2012-12-01  423000
2  CashPoint_1 2012-12-02  312000
3  CashPoint_1 2012-12-03  220000
4  CashPoint_1 2012-12-04  123000
5  CashPoint_2 2012-12-01  223000
6  CashPoint_2 2012-12-02  212000
7  CashPoint_2 2012-12-03  320000
8  CashPoint_2 2012-12-04  223000
9  CashPoint_3 2012-12-01  323000
10 CashPoint_3 2012-12-02  312000
11 CashPoint_3 2012-12-03  270000
12 CashPoint_3 2012-12-04  223000
13 CashPoint_4 2012-12-01  323000
14 CashPoint_4 2012-12-02  412000
15 CashPoint_4 2012-12-03  320000
16 CashPoint_4 2012-12-04  373000
17 CashPoint_5 2012-12-01  223000
18 CashPoint_5 2012-12-02  192000
19 CashPoint_5 2012-12-03  150000
20 CashPoint_5 2012-12-04  133000
```

```r
cash_point_show <- function(cp) {
    # cp[2]: ATM or Branch; cp[1]: id
    cat(sprintf("%-8s %-14s \n", cp[2], cp[1]))
    c(cp[2], cp[1])  # return two vectors
}
```

```r
# show the cash point and id
cash_points <- apply(bank_model, 1, cash_point_show)
```

```
ATM      CashPoint_1
Branch   CashPoint_2
ATM      CashPoint_3
Branch   CashPoint_4
ATM      CashPoint_5
```

```r
class(cash_points)  # a matrix
```

```
[1] "matrix"
```

```r
cash_points
```

```
              [,1]          [,2]          [,3]          [,4]
CashPointType "ATM"         "Branch"      "ATM"         "Branch"
CashPointId   "CashPoint_1" "CashPoint_2" "CashPoint_3" "CashPoint_4"
              [,5]
CashPointType "ATM"
CashPointId   "CashPoint_5"
```

```r
# form an identifier to name the members of the list
apply(bank_model, 1, paste, collapse = "_")
```

```
[1] "CashPoint_1_ATM"    "CashPoint_2_Branch" "CashPoint_3_ATM"
[4] "CashPoint_4_Branch" "CashPoint_5_ATM"
```

```r
# get balance for a cashpoint id
id <- "CashPoint_3"
subset(balances, balances$BranchId == id, -BranchId)
```

```
         Date Balance
9  2012-12-01  323000
10 2012-12-02  312000
11 2012-12-03  270000
12 2012-12-04  223000
```