VERSION 1.0

MAY 29, 2017

INTERNET CAFÉ MANAGEMENT

Produce : 3c14 Nguyen Manh Tien

Hoang Thai Duy Le Phong Sac Pham Thanh Dat

Abstract

*As information technologies continues to develop rapidly, the need of accessing online information, data and online entertainment is increasing fast as well. Therefore Café Internet is becoming more and more popular through the whole country and all over the world. With Internet Café, people can access to the Internet very easily with high quality computer and Internet connection, with only a small amount of money spent. The number of Internet Café has been raising sharply in recent years. Many businessman tend to spend a large amount of money and invest in Internet Café. Nevertheless, the more Internet Cafés are opened, the more competitive the market become. Customer will choose the Internet Café with better gears quality, better internet connection and better environment. In order to maintain the best quality in the Internet Café, a good management application is indeed required. Our project, based on various researches and works, aim to create an application that is able to greatly manage an Internet Café, with the hope that this program will help many Internet Café develop well and stay strong in the market.*

Contents

[I. Introduction 3](#_Toc483856744)

[II. Related Work 4](#_Toc483856745)

[1. Use case diagram 4](#_Toc483856746)

[2. UCDs 5](#_Toc483856747)

[3.1 Domain app tool 6](#_Toc483856748)

[3.2 UML class diagram 7](#_Toc483856749)

[4. Implementation 10](#_Toc483856750)

[4.1 Cluster 10](#_Toc483856751)

[4.2 How to run this app 12](#_Toc483856752)

[III. Result and disscustion 14](#_Toc483856753)

[IV. Conclution 19](#_Toc483856773)

[V. Source Code 19](#_Toc483856774)

# Introduction

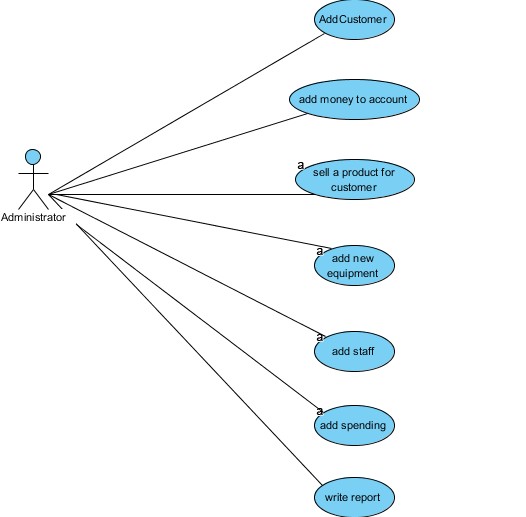
Nowadays, Internet Café is becoming more and more popular in the society, especially among young people. New Internet Café stores are opened every day, everywhere. Many businessman tend to spend a lot of money to open his or her own internet café, consider it as a second source of income. Nevertheless, having enough money to invest into Internet Café is just the start, managing and developing it is a much harder process. In reality, many Internet café stores cannot survive long due to poor management. Customers in any field always want to use services that include good terms, good management, etc. Café Internet is not an exception. If mangers cannot find a way to control what happens every day at the store, the business will soon be a failure and result in a big financial loss.

Our group, which is the combination of 4 young people who like to play games and using internet café services a lot, have been thinking of a way to solve the problem. We thought that with good management, the internet café will have a better chance to survive in a very big and competitive market. Internet Café Management Software is exactly what we need. We have done some research in some famous Internet café in Hanoi, including Phoenix, Cyber, Imba Gaming, etc. We tried to navigate and understand their management software, found out the good and the bad features of those software and then came up with our own.

We want to fully control of all the stuffs and event happens in the Internet café daily, so we try to divide our software into many parts, as much detailed as possible. It has to make sure that all the information of customers, equipment, staffs and finance are carefully written and saved by the application, Furthermore, in order to make the software simple and easy to use, and with the help of the DomainAppTools (a pure Java software development tool), we have to make sure that every function of our software is fit into the tool. In conclusion, the ability of keeping track of everything happened in the Internet café is our most important priority.

# .

# Use case diagram



# UCDs

|  |  |
| --- | --- |
| Functions | Description |
| Add Customer | To add new record of customer’s information into the database. User inputs all customer’s information (name, password, date of birth, address, email, phone number, time remain). If the information is valid, the system will return a successful message, otherwise print out an error message. |
| Add Money to Account | To add money to an user’s account. User input customer’s id, amount of money in ‘timeBill’ class. If the information is valid, the system will return a succesfull message , otherwise print out an error message |
| Sell products for customer | To sell a product for customer. Call class ‘productBill’. User input information (customerID, productID, cost). System returns the product bill. |
| Add new equipment | To add new record of an equipment’s information into the database. User input all equipment’s information (name, provider, cost). If the information is valid, the system will return a successful message, otherwise print out an error message. |
| Add staff | To add new record of a staff’s information into the database. User input all staff’s information (name, date of birth, address, salary, date start.). If the information is valid, the system will return a successful message, otherwise print out an error message. |
| Add spending | Call ‘spending’ class. User input spending information (electric money, internet money, date). System returns spending bill. |
| Write report | Call class ‘FinanceReport’ and input information. |

* 1. **UML class diagram**

# Domain app tool

DomainAppTool is a pure Java application tool that is used to quickly develop applications that conform to the domain driven software design approach. The tool is designed based on the MVC software architecture. Conceptually, the tool’s architecture is built from three key components: model manager, view manager and object manager.

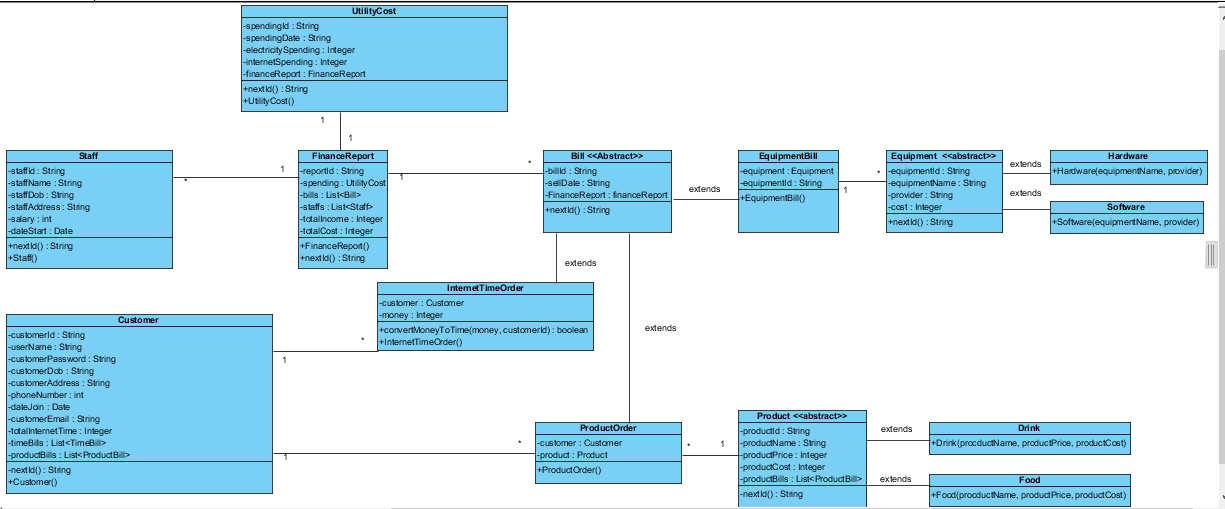
The model manager is responsible for handling the domain classes. A domain class is a Java class that is designed with domain-driven design features.

View manager is the component responsible for the task of automatically generating GUI at run- time from the design information embedded in the application’s domain classes.

The object manager effectively manages the objects in heap memory and provide mechanism to store objects into an external storage space.

We use the DomainAppTool because of the key feature that it only requires the developer to specify the set of domain classes of an application. The entire application (which includes a graphic user interface and an object storage) is automatically generated at run-time.

# UML class diagram

****

* 1. **Table**

\* Customer

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attributes | Type | Mutable | Optional | Length | Min | Max |
| customerId | String | False | False | 10 |  |  |
| customerPassword | String | True | False |  | 8 | 20 |
| userName | String | True | False | 30 |  |  |
| customerDob | String | True | False | 10 |  |  |
| customerAddress | String | True | False | 30 |  |  |
| phoneNumber | Int | True | False | 11 |  |  |
| dateJoin | String | True | False | 10 |  |  |
| customerEmail | String | True | False | 30 |  |  |
| timeRemain | Int | True | False | 10 | 0 |  |
| timeBill | Collection | True | False |  |  |  |
| productBill | Collection | True | False |  |  |  |

\* Staff

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute | Type | Mutable | Optional | Length | Min | Max |
| staffId | String | False | False | 10 |  |  |
| staffName | String | True | False | 30 |  |  |
| staffDob | String | True | False | 10 |  |  |
| staffAddress | String | True | False | 30 |  |  |
| Salary | Int | True | False | 10 | 0 |  |
| dateStart | String | True | False | 10 |  |  |

\* UtilityCost :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute | Type | Mutable | Optional | Length | Min | Max |
| utilityCostId | String | False | False | 10 |  |  |
| spendingDate | String | True | False | 10 |  |  |
| electricitySpending | Integer | True | False | 10 |  |  |
| internetSpending | Integer | True | False | 10 |  |  |
| financeReport | Domain | True | False |  |  |  |

\* Equipment :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Attribute | Type | Mutable | Optional | Length | Min | Max |  |
| equipmentId | String | False | false | 10 |  |  |
| equipmentName | String | True | false | 30 |  |  |
| provider | String | True | False | 30 |  |  |  |
| cost | Integer | True | False | 10 | 0 |  |  |

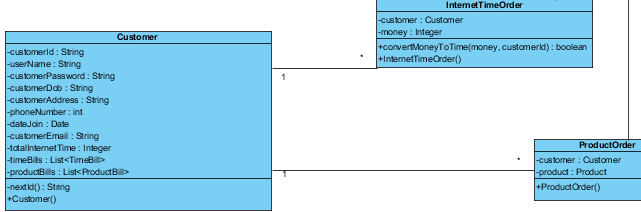
\* Finance Report :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute | Type | Mutable | Optional | Length | Min | Max |
| reportId | String | False | False |  |  |  |
| Spendings | Collection | True | True |  |  |  |
| Bills | Collection | True | True |  |  |  |
| Staffs | Collection | True | True |  |  |  |
| totalIncome | Integer | True | False |  |  |  |
| totalCost | Integer | True | False |  |  |  |

# Implementation

# Cluster

**Time order**

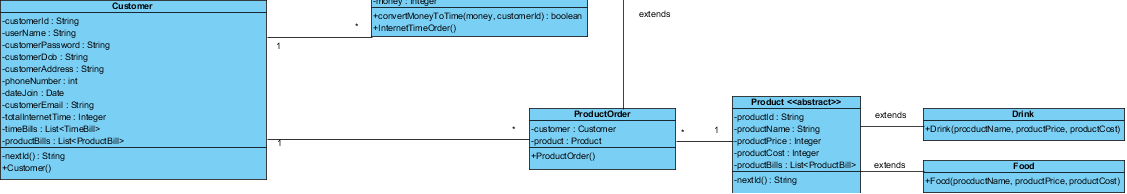


* Introduction : This cluster for customer order a time in their account
* Detail : First we implement class Customer for create a customer account

Second we implement class InternetTimeOrder

* Data table : each class is linked to database ia domainAppTool

**Product**

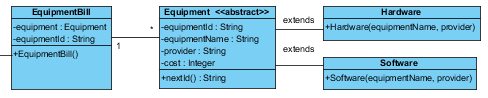


* Introduction : This cluster for customer order a product
* Detail : Class Customer is implemented in the previous cluster

We comtiniue implement abstract class Product and two sub class is Food and Drink. Finnaly we create class ProductOrder for linked between Customer and ProductOrder

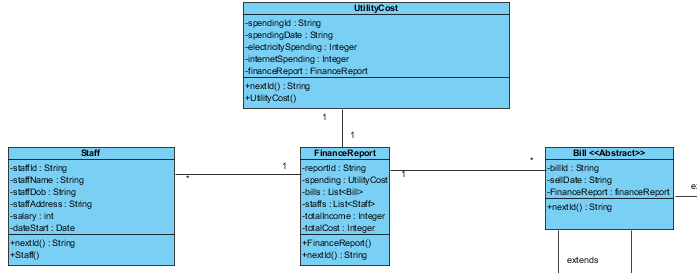
* Data table : each class is linked to database ii domainAppTool.

**Equipment**

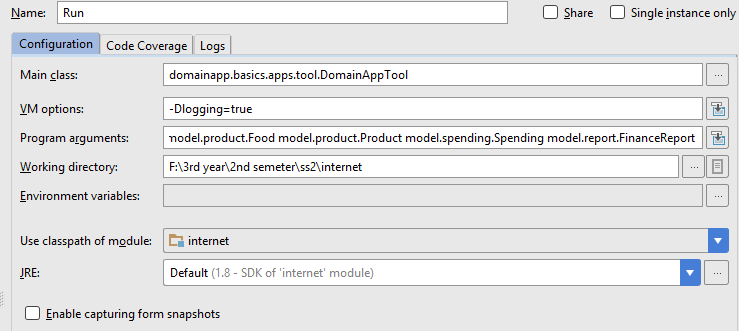


* Introduction : The equipment include Hardware and Software class extends abstract Equipment class . All of them has full information about name, provider, cost.
* Detail : Two class will be developed by java swing with domainAppTool
* lData table : each class is linked to database ia domainAppTool

**FinanceReport**



* Introduction : This cluster for summary the interest rate of this cyber
* Detail : Abstract class bill is implement in previous cluster. We create UtilityCost for input electric and internet spending, And the salary of staff is relate to report so we implement Staff class in here
* Data table : each class is linked to database ia domainAppTool.
  1. **How to run this app**
* 1 step : install source code and rebuild in your ide
* 2 step : click on edit configure



Full in like that

\* Notice : program and argument : configure

model.bill.Bill model.bill.EquipmentBill model.bill.ProductOrder model.bill.InternetTimeOrder model.equipment.Equipment model.equipment.Hardware model.equipment.Software model.people.Customer model.people.Staff model.product.Drink model.product.Food model.product.Product

model.spending.UtilityCost model.report.FinanceReport

# Result and disscustion

# Create a report

# 

# Create customer

# 

# Create a product

# 

# Create a equipment

# 

# Create a product bill

# 

# Create an equipment bill

# 

# Create a time bill

# 

# Create a staff

# 

# Finance report

# After create other things, finance report auto update its properties

# 

# Conclution

In this paper, we have introduced a efficient architecture to build a internet café managerment tool. The system has been proved to be compatible with English and. In the near future, basing on the promising result achieved from our knowledge, we will make it can be faster and more convinent.

# Source Code

# Bill packet

1. **Bill**

**package** model.bill;  
  
**import** domainapp.basics.exceptions.ConstraintViolationException;  
**import** domainapp.basics.model.meta.DAssoc;  
**import** domainapp.basics.model.meta.DAttr;  
**import** domainapp.basics.model.meta.DClass;  
**import** domainapp.basics.model.meta.DOpt;  
**import** domainapp.basics.util.Tuple;  
**import** model.people.Customer;  
**import** model.report.FinanceReport;  
  
  
@DClass(schema = **"internetCafe"**)  
**public abstract class** Bill {  
 @DAttr(name=**"billId"**, id = **true**, auto = **true**, type= DAttr.Type.***String***, length = 10, mutable = **false**, optional = **false**)  
 **private** String **billId**;  
 **private static int** *idCounter* = 0;  
  
 @DAttr(name = **"sellDate"**, type = DAttr.Type.***String***, mutable = **true**, optional = **false**, length = 20)  
 **private** String **sellDate**;  
  
 @DAttr(name = **"financeReport"**, type = DAttr.Type.***Domain***, length = 5, optional = **false**)  
 @DAssoc(ascName = **"FinanceReport-has-bills"**, role = **"bill"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***Many***,  
 associate = @DAssoc.Associate(type = FinanceReport.**class**, cardMin = 1, cardMax = 1), dependsOn = **true**)  
 **private** FinanceReport **financeReport**;  
  
 **public** Bill(String billId, String date, FinanceReport financeReport) {  
 **this**.**billId** = billId;  
 **this**.**sellDate** = date;  
 **this**.**financeReport** = financeReport;  
 }  
  
 **public** Bill(String date, FinanceReport financeReport) {  
 **this**(**null**,date,financeReport);  
 }  
  
 **public** Bill(String sellDate) {  
 **this**(**null**,sellDate,**null**);  
 }  
  
 **private** String nextID(String currId){  
 **if** (currId == **null**){  
 *idCounter*++;  
 **return "B"**+ *idCounter*;  
 }**else** {  
 **int** num = Integer.*parseInt*(currId.substring(1));  
 **if** (num > *idCounter*){  
 *idCounter* = num;  
 }  
 **return** currId;  
 }  
 }  
  
 **public** String getBillId() {  
 **return billId**;  
 }  
  
 **public void** setBillId(String billId) {  
 **this**.**billId** = billId;  
 }  
  
 @DOpt(type=DOpt.Type.***AutoAttributeValueSynchroniser***)  
 **public static void** updateAutoGeneratedValue(  
 DAttr attrib,  
 Tuple derivingValue,  
 Object minVal,  
 Object maxVal) **throws** ConstraintViolationException {  
  
 **if** (minVal != **null** && maxVal != **null**) {  
 *//****TODO: update this for the correct attribute if there are more than one auto attributes of this class*** String maxId = (String) maxVal;  
 **try** {  
 **int** maxIdNum = Integer.*parseInt*(maxId.substring(1));  
  
 **if** (maxIdNum > *idCounter*) *// extra check  
 idCounter* = maxIdNum;  
  
 } **catch** (RuntimeException e) {  
 **throw new** ConstraintViolationException(  
 ConstraintViolationException.Code.***INVALID\_VALUE***, e, **new** Object[] {maxId});  
 }  
 }  
 }  
  
 **public static int** getIdCounter() {  
 **return** *idCounter*;  
 }  
  
 **public static void** setIdCounter(**int** idCounter) {  
 Bill.*idCounter* = idCounter;  
 }  
  
 **public** String getSellDate() {  
 **return sellDate**;  
 }  
  
 **public void** setSellDate(String sellDate) {  
 **this**.**sellDate** = sellDate;  
 }  
  
 **public** FinanceReport getFinanceReport() {  
 **return financeReport**;  
 }  
  
 **public void** setFinanceReport(FinanceReport financeReport) {  
 **this**.**financeReport** = financeReport;  
 }  
}

1. **EquipmentBill**

**package** model.bill;  
  
**import** domainapp.basics.exceptions.ConstraintViolationException;  
**import** domainapp.basics.model.meta.DAssoc;  
**import** domainapp.basics.model.meta.DAttr;  
**import** domainapp.basics.model.meta.DClass;  
**import** domainapp.basics.model.meta.DOpt;  
**import** domainapp.basics.util.Tuple;  
**import** model.equipment.Equipment;  
**import** model.product.Product;  
**import** model.report.FinanceReport;  
  
  
@DClass(schema = **"internetCafe"**)  
**public class** EquipmentBill **extends** Bill {  
  
 @DAttr(name = **"equipment"**, type = DAttr.Type.***Domain***, length = 5, optional = **false**)  
 @DAssoc(ascName = **"equipment-has-equipmentBills"**, role = **"equipmentBill"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***Many***,  
 associate = @DAssoc.Associate(type = Equipment.**class**, cardMin = 1, cardMax = 1), dependsOn = **true**)  
 **private** Equipment **equipment**;  
  
 **public** EquipmentBill(String billId, String date, FinanceReport financeReport, Equipment equipment) {  
 **super**( billId,date, financeReport);  
 **this**.**equipment** = equipment;  
 financeReport.setTotalCost(equipment.getCost());  
 financeReport.setInterestRate((- equipment.getCost()));  
 }  
  
 **public** EquipmentBill(String date, FinanceReport financeReport, Equipment equipment) {  
 **this**(**null**,date,financeReport,equipment);  
 }  
  
 **public** Equipment getEquipment() {  
 **return equipment**;  
 }  
  
 **public void** setEquipment(Equipment equipment) {  
 **this**.**equipment** = equipment;  
 }  
}

## InternetTimeOrder

**package** model.bill;  
  
**import** domainapp.basics.model.meta.DAssoc;  
**import** domainapp.basics.model.meta.DAttr;  
**import** domainapp.basics.model.meta.DClass;  
**import** model.people.Customer;  
**import** model.report.FinanceReport;  
  
*/\*\*  
 \* Created by nguye on 16/04/2017.  
 \*/*@DClass(schema = **"internetCafe"**)  
**public class** InternetTimeOrder **extends** Bill {  
 @DAttr(name = **"customer"**, type = DAttr.Type.***Domain***, length = 5, optional = **false**)  
 @DAssoc(ascName = **"customer-has-internetTimeOrders"**, role = **"InternetTimeOrder"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***Many***,  
 associate = @DAssoc.Associate(type = Customer.**class**, cardMin = 1, cardMax = 1), dependsOn = **true**)  
 **private** Customer **customer**;  
  
 @DAttr(name = **"money"**, type = DAttr.Type.***Integer***, mutable = **true**, optional = **false**, min = 0)  
 **private** Integer **money**;  
  
 **public** InternetTimeOrder(String billId, String date, FinanceReport financeReport, Customer customer, Integer money) {  
 **super**(billId, date, financeReport);  
 **this**.**customer** = customer;  
 **this**.**money** = money;  
 customer.setTotalInternetTime(convertMoneyToTime(money));  
 financeReport.setTotalIncome(money);  
 financeReport.setInterestRate(money);  
 }  
  
 **public** InternetTimeOrder(String date, FinanceReport financeReport, Customer customer, Integer money) {  
 **this**(**null**,date,financeReport,customer,money);  
 }  
  
 *// 1 hour = 3600 = 60\*60 => 1vnd/s  
 //* **private** Integer convertMoneyToTime(Integer money) {  
 **return** money / 50;  
 }  
  
 **public** Customer getCustomer() {  
 **return customer**;  
 }  
  
 **public void** setCustomer(Customer customer) {  
 **this**.**customer** = customer;  
 }  
  
 **public** Integer getMoney() {  
 **return money**;  
 }  
  
 **public void** setMoney(Integer money) {  
 **customer**.setTotalInternetTime(convertMoneyToTime(money));  
 }  
  
}

## Packet Equipment

1. **Equipment**

**package** model.equipment;  
  
**import** domainapp.basics.exceptions.ConstraintViolationException;  
**import** domainapp.basics.model.meta.\*;  
**import** domainapp.basics.util.Tuple;  
**import** model.bill.EquipmentBill;  
  
**import** java.util.List;  
  
@DClass(schema = **"internetCafe"**)  
**public abstract class** Equipment {  
 @DAttr(name = **"equipmentId"**, type = DAttr.Type.***String***, id = **true**, auto = **true**, mutable = **false**, optional = **false**, length = 10)  
 **private** String **equipmentId**;  
 **private static int** *idCounter*;  
 @DAttr(name = **"equipmentName"**, type = DAttr.Type.***String***, mutable = **true**, optional = **false**, length = 30)  
 **private** String **equipmentName**;  
 @DAttr(name = **"provider"**, type = DAttr.Type.***String***, mutable = **true**, optional = **false**, length = 30)  
 **private** String **provider**;  
 @DAttr(name = **"cost"**, type = DAttr.Type.***Integer***, mutable = **true**, optional = **false**, length = 10, min = 0)  
 **private int cost**;  
 @DAttr(name = **"equipmentBills"**, type = DAttr.Type.***Collection***, optional = **false**,  
 serialisable = **false**, filter = @Select(clazz = EquipmentBill.**class**))  
 @DAssoc(ascName = **"equipment-has-equipmentBills"**, role = **"equipment"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***One***,  
 associate = @DAssoc.Associate(type = EquipmentBill.**class**, cardMin = 0, cardMax = 25))  
 **private** List<EquipmentBill> **equipmentBills**;  
  
 **public** Equipment(String equipmentId, String equipmentName, String provider, Integer cost) {  
 **this**.**equipmentId** = nextID(equipmentId);  
 **this**.**equipmentName** = equipmentName;  
 **this**.**provider** = provider;  
 **this**.**cost** = cost;  
 }  
  
 **public** Equipment(String equipmentName, String provider, Integer cost) {  
 **this**(**null**, equipmentName, provider, cost);  
 }  
  
 **private** String nextID(String currId) {  
 **if** (currId == **null**) {  
 *idCounter*++;  
 **return "E"** + *idCounter*;  
 } **else** {  
 **int** num = Integer.*parseInt*(currId.substring(1));  
 **if** (num > *idCounter*) {  
 *idCounter* = num;  
 }  
 **return** currId;  
 }  
 }  
  
 @DOpt(type = DOpt.Type.***AutoAttributeValueSynchroniser***)  
 **public static void** updateAutoGeneratedValue(  
 DAttr attrib,  
 Tuple derivingValue,  
 Object minVal,  
 Object maxVal) **throws** ConstraintViolationException {  
  
 **if** (minVal != **null** && maxVal != **null**) {  
 *//****TODO: update this for the correct attribute if there are more than one auto attributes of this class*** String maxId = (String) maxVal;  
 **try** {  
 **int** maxIdNum = Integer.*parseInt*(maxId.substring(1));  
  
 **if** (maxIdNum > *idCounter*) *// extra check  
 idCounter* = maxIdNum;  
  
 } **catch** (RuntimeException e) {  
 **throw new** ConstraintViolationException(  
 ConstraintViolationException.Code.***INVALID\_VALUE***, e, **new** Object[]{maxId});  
 }  
 }  
 }  
  
 **public** String getEquipmentId() {  
 **return equipmentId**;  
 }  
  
 **public** String getEquipmentName() {  
 **return equipmentName**;  
 }  
  
 **public void** setEquipmentName(String equipmentName) {  
 **this**.**equipmentName** = equipmentName;  
 }  
  
 **public** String getProvider() {  
 **return provider**;  
 }  
  
 **public void** setProvider(String provider) {  
 **this**.**provider** = provider;  
 }  
  
 **public int** getCost() {  
 **return cost**;  
 }  
  
 **public void** setCost(**int** cost) {  
 **this**.**cost** = cost;  
 }  
  
 **public void** setEquipmentId(String equipmentId) {  
 **this**.**equipmentId** = equipmentId;  
 }  
  
 **public** List<EquipmentBill> getEquipmentBills() {  
 **return equipmentBills**;  
 }  
  
 **public void** setEquipmentBills(List<EquipmentBill> equipmentBills) {  
 **this**.**equipmentBills** = equipmentBills;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addEquipmentBill(EquipmentBill equipmentBill) {  
 **if** (!**this**.**equipmentBills**.contains(equipmentBill))  
 **this**.**equipmentBills**.add(equipmentBill);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewEquipmentBill(EquipmentBill equipmentBill) {  
 **this**.**equipmentBills**.add(equipmentBill);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addEquipmentBill(List<EquipmentBill> equipmentBills) {  
 **boolean** added = **false**;  
 **for** (EquipmentBill equipmentBill : equipmentBills) {  
 **if** (!**this**.**equipmentBills**.contains(equipmentBill)) {  
 **if** (!added) added = **true**;  
 **this**.**equipmentBills**.add(equipmentBill);  
 }  
 }  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewEquipmentBill(List<EquipmentBill> equipmentBills) {  
 **this**.**equipmentBills**.addAll(equipmentBills);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkRemover***)  
 **public boolean** removeEquipmentBill(EquipmentBill equipmentBill) {  
 **boolean** removed = **this**.**equipmentBills**.remove(equipmentBill);  
 **return false**;  
 }  
}

## Hardware

**package** model.equipment;  
  
**import** domainapp.basics.model.meta.DClass;  
  
@DClass(schema = **"internetCafe"**)  
**public class** Hardware **extends** Equipment {  
 **public** Hardware(String equipmentId, String equipmentName, String provider, Integer cost) {  
 **super**(equipmentId, equipmentName, provider, cost);  
 }  
  
 **public** Hardware(String equipmentName, String provider, Integer cost) {  
 **super**(equipmentName, provider, cost);  
 }  
}

## Software

**package** model.equipment;  
  
**import** domainapp.basics.model.meta.DClass;  
  
  
@DClass(schema = **"internetCafe"**)  
**public class** Software **extends** Equipment {  
 **public** Software(String equipmentId, String equipmentName, String provider, Integer cost) {  
 **super**(equipmentId, equipmentName, provider, cost);  
 }  
  
 **public** Software(String equipmentName, String provider, Integer cost) {  
 **super**(equipmentName, provider, cost);  
 }  
}

## Packet people

1. **Customer**

**package** model.people;  
  
**import** domainapp.basics.exceptions.ConstraintViolationException;  
**import** domainapp.basics.model.meta.\*;  
**import** domainapp.basics.util.Tuple;  
**import** model.bill.InternetTimeOrder;  
**import** model.bill.ProductOrder;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
@DClass(schema = **"internetCafe"**)  
**public class** Customer {  
 @DAttr(name = **"customerId"**, id = **true**, auto = **true**, type = DAttr.Type.***String***, length = 10, mutable = **false**, optional = **false**)  
 **private** String **customerId**;  
 **private static int** *idCounter* = 0;  
  
 @DAttr(name = **"userName"**, type = DAttr.Type.***String***, length = 30, mutable = **true**, optional = **false**)  
 **private** String **userName**;  
  
 @DAttr(name = **"customerPassWord"**, type = DAttr.Type.***String***, length = 20, mutable = **true**, optional = **false**, min = 0, max = 20)  
 **private** String **customerPassWord**;  
  
 @DAttr(name = **"customerDob"**, type = DAttr.Type.***String***, length = 10, mutable = **true**, optional = **false**)  
 **private** String **customerDob**;  
  
 @DAttr(name = **"customerAddress"**, type = DAttr.Type.***String***, length = 20, mutable = **true**, optional = **false**)  
 **private** String **customerAddress**;  
  
 @DAttr(name = **"phoneNumber"**, type = DAttr.Type.***String***, length = 11, mutable = **true**, optional = **false**)  
 **private** String **phoneNumber**;  
  
 @DAttr(name = **"dateJoin"**, type = DAttr.Type.***String***, length = 10, mutable = **true**, optional = **false**)  
 **private** String **dateJoin**;  
  
 @DAttr(name = **"customerEmail"**, type = DAttr.Type.***String***, length = 30, mutable = **true**, optional = **false**)  
 **private** String **customerEmail**;  
  
 @DAttr(name = **"totalInternetTime"**, type = DAttr.Type.***Integer***, length = 10, mutable = **true**, optional = **false**, min = 0)  
 **private int totalInternetTime**;  
  
 @DAttr(name = **"InternetTimeOrders"**, type = DAttr.Type.***Collection***, optional = **false**,  
 serialisable = **false**, filter = @Select(clazz = InternetTimeOrder.**class**))  
 @DAssoc(ascName = **"customer-has-internetTimeOrders"**, role = **"customer"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***One***,  
 associate = @DAssoc.Associate(type = InternetTimeOrder.**class**, cardMin = 0, cardMax = 25))  
 **private** ArrayList<InternetTimeOrder> **InternetTimeOrders**;  
  
 @DAttr(name = **"productOrders"**, type = DAttr.Type.***Collection***, optional = **false**,  
 serialisable = **false**, filter = @Select(clazz = ProductOrder.**class**))  
 @DAssoc(ascName = **"customer-has-productOrders"**, role = **"customer"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***One***,  
 associate = @DAssoc.Associate(type = ProductOrder.**class**, cardMin = 0, cardMax = 25))  
 **private** ArrayList<ProductOrder> **productOrders**;  
  
  
 **public** Customer(String userName, String customerPassWord, String customerDob, String customerAddress, String phoneNumber, String dateJoin, String customerEmail) {  
 **this**(**null**, userName, customerPassWord, customerDob, customerAddress, phoneNumber, dateJoin, customerEmail, 0);  
 }  
  
 **public** Customer(String userName, String customerPassWord, String customerDob, String customerAddress, String phoneNumber, String dateJoin, String customerEmail, Integer totalInternetTime) {  
 **this**(**null**, userName, customerPassWord, customerDob, customerAddress, phoneNumber, dateJoin, customerEmail, totalInternetTime);  
 }  
  
 **public** Customer(String customerId, String userName, String customerPassWord, String customerDob, String customerAddress, String phoneNumber, String dateJoin, String customerEmail, Integer totalInternetTime) {  
 **this**.**customerId** = nextID(customerId);  
 **this**.**customerPassWord** = customerPassWord;  
 **this**.**userName** = userName;  
 **this**.**customerDob** = customerDob;  
 **this**.**customerAddress** = customerAddress;  
 **this**.**phoneNumber** = phoneNumber;  
 **this**.**dateJoin** = dateJoin;  
 **this**.**customerEmail** = customerEmail;  
 **this**.**totalInternetTime** = totalInternetTime;  
 }  
  
 **private** String nextID(String currId) {  
 **if** (currId == **null**) {  
 *idCounter*++;  
 **return "C"** + *idCounter*;  
 } **else** {  
 **int** num = Integer.*parseInt*(currId.substring(1));  
 **if** (num > *idCounter*) {  
 *idCounter* = num;  
 }  
 **return** currId;  
 }  
 }  
  
 @DOpt(type = DOpt.Type.***AutoAttributeValueSynchroniser***)  
 **public static void** updateAutoGeneratedValue(  
 DAttr attrib,  
 Tuple derivingValue,  
 Object minVal,  
 Object maxVal) **throws** ConstraintViolationException {  
  
 **if** (minVal != **null** && maxVal != **null**) {  
 *//****TODO: update this for the correct attribute if there are more than one auto attributes of this class*** String maxId = (String) maxVal;  
 **try** {  
 **int** maxIdNum = Integer.*parseInt*(maxId.substring(1));  
  
 **if** (maxIdNum > *idCounter*) *// extra check  
 idCounter* = maxIdNum;  
  
 } **catch** (RuntimeException e) {  
 **throw new** ConstraintViolationException(  
 ConstraintViolationException.Code.***INVALID\_VALUE***, e, **new** Object[]{maxId});  
 }  
 }  
 }  
  
 **public** String getCustomerId() {  
 **return customerId**;  
 }  
  
 **public void** setCustomerId(String customerId) {  
 **this**.**customerId** = customerId;  
 }  
  
 **public** String getUserName() {  
 **return userName**;  
 }  
  
 **public void** setUserName(String userName) {  
 **this**.**userName** = userName;  
 }  
  
 **public** String getCustomerPassWord() {  
 **return customerPassWord**;  
 }  
  
 **public void** setCustomerPassWord(String customerPassWord) {  
 **this**.**customerPassWord** = customerPassWord;  
 }  
  
 **public** String getCustomerDob() {  
 **return customerDob**;  
 }  
  
 **public void** setCustomerDob(String customerDob) {  
 **this**.**customerDob** = customerDob;  
 }  
  
 **public** String getCustomerAddress() {  
 **return customerAddress**;  
 }  
  
 **public void** setCustomerAddress(String customerAddress) {  
 **this**.**customerAddress** = customerAddress;  
 }  
  
 **public** String getPhoneNumber() {  
 **return phoneNumber**;  
 }  
  
 **public void** setPhoneNumber(String phoneNumber) {  
 **this**.**phoneNumber** = phoneNumber;  
 }  
  
 **public** String getDateJoin() {  
 **return dateJoin**;  
 }  
  
 **public void** setDateJoin(String dateJoin) {  
 **this**.**dateJoin** = dateJoin;  
 }  
  
 **public** String getCustomerEmail() {  
 **return customerEmail**;  
 }  
  
 **public void** setCustomerEmail(String customerEmail) {  
 **this**.**customerEmail** = customerEmail;  
 }  
  
 **public** List<InternetTimeOrder> getInternetTimeOrders() {  
 **return InternetTimeOrders**;  
 }  
  
 **public int** getTotalInternetTime() {  
 **return totalInternetTime**;  
 }  
  
 **public void** setTotalInternetTime(**int** totalInternetTime) {  
 **this**.**totalInternetTime** += totalInternetTime;  
 }  
  
 **public void** setInternetTimeOrders(ArrayList<InternetTimeOrder> InternetTimeOrders) {  
 **this**.**InternetTimeOrders** = InternetTimeOrders;  
 }  
  
 **public** ArrayList<ProductOrder> getProductOrders() {  
 **return productOrders**;  
 }  
  
 **public void** setProductOrders(ArrayList<ProductOrder> productOrders) {  
 **this**.**productOrders** = productOrders;  
 }  
  
 *// product bill* @DOpt(type = DOpt.Type.***LinkAdder***)  
 *//only need to do this for reflexive association: @MemberRef(name="enrolments")* **public boolean** addProductBill(ProductOrder productOrder) {  
 **if** (!**productOrders**.contains(productOrder))  
 **productOrders**.add(productOrder);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewProductBill(ProductOrder productOrder) {  
 **productOrders**.add(productOrder);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 *//@MemberRef(name="enrolments")* **public boolean** addProductBill(List<ProductOrder> productOrders) {  
 **boolean** added = **false**;  
 **for** (ProductOrder productOrder : productOrders) {  
 **if** (!**this**.**productOrders**.contains(productOrder)) {  
 **if** (!added) added = **true**;  
 **this**.**productOrders**.add(productOrder);  
 }  
 }  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewProductBill(List<ProductOrder> productOrders) {  
 **this**.**productOrders**.addAll(productOrders);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkRemover***)  
 **public boolean** removeProductBill(ProductOrder productOrder) {  
 **boolean** removed = **this**.**productOrders**.remove(productOrder);  
 **return false**;  
 }  
  
 *// time bill* @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addInternetTimeOrder(InternetTimeOrder InternetTimeOrder) {  
 **if** (!**InternetTimeOrders**.contains(InternetTimeOrder))  
 **InternetTimeOrders**.add(InternetTimeOrder);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewInternetTimeOrder(InternetTimeOrder InternetTimeOrder) {  
 **InternetTimeOrders**.add(InternetTimeOrder);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addInternetTimeOrder(List<InternetTimeOrder> InternetTimeOrders) {  
 **boolean** added = **false**;  
 **for** (InternetTimeOrder InternetTimeOrder : InternetTimeOrders) {  
 **if** (!**this**.**InternetTimeOrders**.contains(InternetTimeOrder)) {  
 **if** (!added) added = **true**;  
 **this**.**InternetTimeOrders**.add(InternetTimeOrder);  
 }  
 }  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewInternetTimeOrder(List<InternetTimeOrder> InternetTimeOrders) {  
 **this**.**InternetTimeOrders**.addAll(InternetTimeOrders);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkRemover***)  
 **public boolean** removeInternetTimeOrder(InternetTimeOrder InternetTimeOrder) {  
 **boolean** removed = **this**.**InternetTimeOrders**.remove(InternetTimeOrder);  
 **return false**;  
 }  
  
}

## Staff

**package** model.people;  
  
**import** domainapp.basics.exceptions.ConstraintViolationException;  
**import** domainapp.basics.model.meta.DAssoc;  
**import** domainapp.basics.model.meta.DAttr;  
**import** domainapp.basics.model.meta.DClass;  
**import** domainapp.basics.model.meta.DOpt;  
**import** domainapp.basics.util.Tuple;  
**import** model.report.FinanceReport;  
  
@DClass(schema = **"internetCafe"**)  
**public class** Staff {  
 @DAttr(name=**"staffId"**, id = **true**, auto = **true**, type= DAttr.Type.***String***, length = 10, mutable = **false**, optional = **false**)  
 **private** String **staffId**;  
 **private static int** *idCounter* = 0;  
  
 @DAttr(name=**"name"**, type = DAttr.Type.***String***, length = 30, mutable = **true**, optional = **false**)  
 **private** String **name**;  
  
 @DAttr(name=**"staffDob"**, type = DAttr.Type.***String***, length = 10, mutable = **true**, optional = **false**)  
 **private** String **staffDob**;  
  
 @DAttr(name = **"staffAddress"**, type = DAttr.Type.***String***, length = 20, mutable = **true**, optional = **false**)  
 **private** String **staffAddress**;  
  
 @DAttr(name=**"salary"**, type = DAttr.Type.***Integer***, length = 11, mutable = **true**, optional = **false**)  
 **private** Integer **salary**;  
  
 @DAttr(name=**"startDate"**, type = DAttr.Type.***String***,length = 10, mutable = **true**, optional = **false**)  
 **private** String **startDate**;  
  
 @DAttr(name = **"financeReport"**, type = DAttr.Type.***Domain***, length = 5, optional = **false**)  
 @DAssoc(ascName = **"FinanceReport-has-salary"**, role = **"staff"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***Many***,  
 associate = @DAssoc.Associate(type = Customer.**class**, cardMin = 1, cardMax = 1), dependsOn = **true**)  
 **private** FinanceReport **financeReport**;  
  
 **public** Staff(String staffId, String name, String staffDob, String staffAddress, Integer salary, String startDate, FinanceReport financeReport) {  
 **this**.**staffId** = nextID(staffId);  
 **this**.**name** = name;  
 **this**.**staffDob** = staffDob;  
 **this**.**staffAddress** = staffAddress;  
 **this**.**salary** = salary;  
 **this**.**startDate** = startDate;  
 **this**.**financeReport** = financeReport;  
 financeReport.setTotalCost(salary);  
 financeReport.setInterestRate((-salary));  
 }  
  
 **public** Staff(String name, String staffDob, String staffAddress, Integer salary, String startDate, FinanceReport financeReport) {  
 **this**(**null**,name,staffDob,staffAddress,salary,startDate,financeReport);  
 }  
  
 **private** String nextID(String currId){  
 **if** (currId == **null**){  
 *idCounter*++;  
 **return "C"**+ *idCounter*;  
 }**else** {  
 **int** num = Integer.*parseInt*(currId.substring(1));  
 **if** (num > *idCounter*){  
 *idCounter* = num;  
 }  
 **return** currId;  
 }  
 }  
  
 @DOpt(type=DOpt.Type.***AutoAttributeValueSynchroniser***)  
 **public static void** updateAutoGeneratedValue(  
 DAttr attrib,  
 Tuple derivingValue,  
 Object minVal,  
 Object maxVal) **throws** ConstraintViolationException {  
  
 **if** (minVal != **null** && maxVal != **null**) {  
 *//****TODO: update this for the correct attribute if there are more than one auto attributes of this class*** String maxId = (String) maxVal;  
 **try** {  
 **int** maxIdNum = Integer.*parseInt*(maxId.substring(1));  
  
 **if** (maxIdNum > *idCounter*) *// extra check  
 idCounter* = maxIdNum;  
  
 } **catch** (RuntimeException e) {  
 **throw new** ConstraintViolationException(  
 ConstraintViolationException.Code.***INVALID\_VALUE***, e, **new** Object[] {maxId});  
 }  
 }  
 }  
  
 **public** String getStaffId() {  
 **return staffId**;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** String getStaffDob() {  
 **return staffDob**;  
 }  
  
 **public void** setStaffDob(String staffDob) {  
 **this**.**staffDob** = staffDob;  
 }  
  
 **public** String getStaffAddress() {  
 **return staffAddress**;  
 }  
  
 **public void** setStaffAddress(String staffAddress) {  
 **this**.**staffAddress** = staffAddress;  
 }  
  
 **public** Integer getSalary() {  
 **return salary**;  
 }  
  
 **public void** setSalary(Integer salary) {  
 **this**.**salary** = salary;  
 }  
  
 **public** String getStartDate() {  
 **return startDate**;  
 }  
  
 **public void** setStartDate(String startDate) {  
 **this**.**startDate** = startDate;  
 }  
  
 **public** FinanceReport getFinanceReport() {  
 **return financeReport**;  
 }  
  
 **public void** setFinanceReport(FinanceReport financeReport) {  
 **this**.**financeReport** = financeReport;  
 }  
}

## Product

1. **Product**

**package** model.product;  
  
**import** domainapp.basics.exceptions.ConstraintViolationException;  
**import** domainapp.basics.model.meta.\*;  
**import** domainapp.basics.util.Tuple;  
**import** model.bill.ProductOrder;  
  
**import** java.util.List;  
  
@DClass(schema = **"internetCafe"**)  
**public abstract class** Product {  
 @DAttr(name = **"productId"**, type = DAttr.Type.***String***, length = 10, auto = **true**, mutable = **false**, id = **true**, optional = **false**)  
 **private** String **productId**;  
 **private static int** *idCounter* = 0;  
 @DAttr(name = **"productName"**, type = DAttr.Type.***String***, length = 30, mutable = **true**, optional = **false**)  
 **private** String **productName**;  
 @DAttr(name = **"productPrice"**, type = DAttr.Type.***Integer***, length = 10, mutable = **true**, optional = **false**)  
 **private** Integer **productPrice**;  
 @DAttr(name = **"productCost"**, type = DAttr.Type.***Integer***, length = 10, mutable = **true**, optional = **false**)  
 **private** Integer **productCost**;  
  
 @DAttr(name = **"productOrders"**, type = DAttr.Type.***Collection***, optional = **false**,  
 serialisable = **false**, filter = @Select(clazz = ProductOrder.**class**))  
 @DAssoc(ascName = **"product-has-productOrders"**, role = **"product"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***One***,  
 associate = @DAssoc.Associate(type = ProductOrder.**class**, cardMin = 0, cardMax = 25))  
 **private** List<ProductOrder> **productOrders**;  
  
 **public** Product(String productId, String productName, Integer productPrice, Integer productCost) {  
 **this**.**productId** = nextID(productId);  
 **this**.**productName** = productName;  
 **this**.**productPrice** = productPrice;  
 **this**.**productCost** = productCost;  
  
 }  
  
 **public** Product(String productName, Integer productPrice, Integer productCost) {  
 **this**(**null**, productName, productPrice, productCost);  
 }  
  
 **private** String nextID(String currId) {  
 **if** (currId == **null**) {  
 *idCounter*++;  
 **return "P"** + *idCounter*;  
 } **else** {  
 **int** num = Integer.*parseInt*(currId.substring(1));  
 **if** (num > *idCounter*) {  
 *idCounter* = num;  
 }  
 **return** currId;  
 }  
 }  
  
 @DOpt(type = DOpt.Type.***AutoAttributeValueSynchroniser***)  
 **public static void** updateAutoGeneratedValue(  
 DAttr attrib,  
 Tuple derivingValue,  
 Object minVal,  
 Object maxVal) **throws** ConstraintViolationException {  
  
 **if** (minVal != **null** && maxVal != **null**) {  
 *//****TODO: update this for the correct attribute if there are more than one auto attributes of this class*** String maxId = (String) maxVal;  
 **try** {  
 **int** maxIdNum = Integer.*parseInt*(maxId.substring(1));  
  
 **if** (maxIdNum > *idCounter*) *// extra check  
 idCounter* = maxIdNum;  
  
 } **catch** (RuntimeException e) {  
 **throw new** ConstraintViolationException(  
 ConstraintViolationException.Code.***INVALID\_VALUE***, e, **new** Object[]{maxId});  
 }  
 }  
 }  
  
 **public** String getProductId() {  
 **return productId**;  
 }  
  
  
 **public void** setProductId(String productId) {  
 **this**.**productId** = productId;  
 }  
  
 **public** String getProductName() {  
 **return productName**;  
 }  
  
 **public void** setProductName(String productName) {  
 **this**.**productName** = productName;  
 }  
  
 **public** Integer getProductPrice() {  
 **return productPrice**;  
 }  
  
 **public void** setProductPrice(Integer productPrice) {  
 **this**.**productPrice** = productPrice;  
 }  
  
 **public** Integer getProductCost() {  
 **return productCost**;  
 }  
  
 **public void** setProductCost(Integer productCost) {  
 **this**.**productCost** = productCost;  
 }  
  
 **public** List<ProductOrder> getProductOrders() {  
 **return productOrders**;  
 }  
  
 **public void** setProductOrders(List<ProductOrder> productOrders) {  
 **this**.**productOrders** = productOrders;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 *//only need to do this for reflexive association: @MemberRef(name="enrolments")* **public boolean** addProductBill(ProductOrder p) {  
 **if** (!**productOrders**.contains(p))  
 **productOrders**.add(p);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewProductBill(ProductOrder p) {  
 **productOrders**.add(p);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addProductBill(List<ProductOrder> productOrders) {  
 **boolean** added = **false**;  
 **for** (ProductOrder p : productOrders) {  
 **if** (!**this**.**productOrders**.contains(p)) {  
 **if** (!added) added = **true**;  
 **this**.**productOrders**.add(p);  
 }  
 }  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewProductBill(List<ProductOrder> productOrders) {  
 **this**.**productOrders**.addAll(productOrders);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkRemover***)  
 **public boolean** removeProductBill(ProductOrder productOrder) {  
 **boolean** removed = **this**.**productOrders**.remove(productOrder);  
 **return false**;  
 }  
  
}

## Drink

**package** model.product;  
  
**import** domainapp.basics.model.meta.DClass;  
  
  
@DClass(schema = **"internetCafe"**)  
**public class** Drink **extends** Product {  
 **public** Drink(String productName, Integer productPrice, Integer productCost) {  
 **super**(productName, productPrice, productCost);  
 }  
  
 **public** Drink(String productId, String productName, Integer productPrice, Integer  
 productCost) {  
 **super**(productId, productName, productPrice, productCost);  
 }  
}

## Food

**package** model.product;  
  
**import** domainapp.basics.model.meta.DClass;  
  
  
@DClass(schema = **"internetCafe"**)  
**public class** Food **extends** Product {  
 **public** Food(String productId, String productName, Integer productPrice, Integer productCost) {  
 **super**(productId, productName, productPrice, productCost);  
 }  
  
 **public** Food(String productName, Integer productPrice, Integer productCost) {  
 **super**(productName, productPrice, productCost);  
 }  
}

## Report

**package** model.report;  
  
**import** domainapp.basics.exceptions.ConstraintViolationException;  
**import** domainapp.basics.model.meta.\*;  
**import** domainapp.basics.util.Tuple;  
**import** domainapp.basics.util.cache.StateHistory;  
**import** model.bill.Bill;  
**import** model.bill.EquipmentBill;  
**import** model.bill.InternetTimeOrder;  
**import** model.bill.ProductOrder;  
**import** model.people.Staff;  
**import** model.utilityCost.UtilityCost;  
  
**import** java.util.List;  
  
**public class** FinanceReport {  
 @DAttr(name = **"reportId"**, id = **true**, auto = **true**, type = DAttr.Type.***String***, length = 10, mutable = **false**, optional = **false**)  
 **private** String **reportId**;  
 **private static int** *idCounter* = 0;  
  
 @DAttr(name = **"utilityCosts"**, type = DAttr.Type.***Collection***, optional = **true**,  
 serialisable = **false**, filter = @Select(clazz = UtilityCost.**class**))  
 @DAssoc(ascName = **"FinanceReport-has-utilityCosts"**, role = **"financeReport"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***One***,  
 associate = @DAssoc.Associate(type = UtilityCost.**class**, cardMin = 0, cardMax = 25))  
 **private** List<UtilityCost> **utilityCosts**;  
  
 @DAttr(name = **"bills"**, type = DAttr.Type.***Collection***, optional = **true**,  
 serialisable = **false**, filter = @Select(clazz = Bill.**class**))  
 @DAssoc(ascName = **"FinanceReport-has-bills"**, role = **"financeReport"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***One***,  
 associate = @DAssoc.Associate(type = Bill.**class**, cardMin = 0, cardMax = 25))  
 **private** List<Bill> **bills**;  
  
 @DAttr(name = **"salary"**, type = DAttr.Type.***Collection***, optional = **true**,  
 serialisable = **false**, filter = @Select(clazz = Staff.**class**))  
 @DAssoc(ascName = **"FinanceReport-has-salary"**, role = **"financeReport"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***One***,  
 associate = @DAssoc.Associate(type = Staff.**class**, cardMin = 0, cardMax = 25))  
 **private** List<Staff> **salary**;  
  
 @DAttr(name=**"totalCost"**, type = DAttr.Type.***Integer***,mutable = **true**,optional = **true**,length = 10)  
 **private** Integer **totalCost**;  
  
 @DAttr(name=**"totalIncome"**,type = DAttr.Type.***Integer***,mutable = **true**,optional = **true**,length = 10)  
 **private** Integer **totalIncome**;  
  
 @DAttr(name = **"interestRate"**,type = DAttr.Type.***Integer***,mutable = **true**,optional = **true**,length = 10)  
 **private** Integer **interestRate**;  
  
  
 **public** FinanceReport(String reportId, List<UtilityCost> utilityCosts, List<Bill> bills, List<Staff> salary, Integer totalCost, Integer totalIncome) {  
 **this**.**reportId** = nextID(reportId);  
 **this**.**utilityCosts** = utilityCosts;  
 **this**.**bills** = bills;  
 **this**.**salary** = salary;  
 **this**.**totalCost** = totalCost;  
 **this**.**totalIncome** = totalIncome;  
 }  
  
 **public** FinanceReport(Integer totalCost, Integer totalIncome) {  
 **this**(**null**,**null**,**null**,**null**,totalCost,totalIncome);  
 }  
  
 **public** FinanceReport(String reportId, List<UtilityCost> utilityCosts, List<Bill> bills, List<Staff> salary) {  
 **this**(**null**, utilityCosts,bills, salary,0,0);  
 }  
  
 **public** FinanceReport(String reportId, Integer totalCost, Integer totalIncome) {  
 **this**(**null**,**null**,**null**,**null**,totalCost,totalIncome);  
 }  
  
 **public** FinanceReport(String reportId) {  
 **this**(**null**,**null**,**null**,**null**,0,0);  
 }  
  
 **public** FinanceReport() {  
 **this**(**null**,**null**,**null**,**null**,0,0);  
 }  
  
 **public** FinanceReport(List<UtilityCost> utilityCosts, List<Bill> bills, List<Staff> salary) {  
 **this**(**null**, utilityCosts, bills, salary,0,0);  
 }  
  
  
 **private** String nextID(String currId) {  
 **if** (currId == **null**) {  
 *idCounter*++;  
 **return "FR"** + *idCounter*;  
 } **else** {  
 **int** num = Integer.*parseInt*(currId.substring(2));  
 **if** (num > *idCounter*) {  
 *idCounter* = num;  
 }  
 **return** currId;  
 }  
 }  
  
 @DOpt(type = DOpt.Type.***AutoAttributeValueSynchroniser***)  
 **public static void** updateAutoGeneratedValue(  
 DAttr attrib,  
 Tuple derivingValue,  
 Object minVal,  
 Object maxVal) **throws** ConstraintViolationException {  
  
 **if** (minVal != **null** && maxVal != **null**) {  
 String maxId = (String) maxVal;  
 **try** {  
 **int** maxIdNum = Integer.*parseInt*(maxId.substring(2));  
  
 **if** (maxIdNum > *idCounter*) *// extra check  
 idCounter* = maxIdNum;  
  
 } **catch** (RuntimeException e) {  
 **throw new** ConstraintViolationException(  
 ConstraintViolationException.Code.***INVALID\_VALUE***, e, **new** Object[]{maxId});  
 }  
 }  
 }  
  
 **public** String getReportId() {  
 **return reportId**;  
 }  
  
 **public void** setReportId(String reportId) {  
 **this**.**reportId** = reportId;  
 }  
  
 **public** List<UtilityCost> getUtilityCosts() {  
 **return utilityCosts**;  
 }  
  
 **public void** setUtilityCosts(List<UtilityCost> utilityCosts) {  
 **this**.**utilityCosts** = utilityCosts;  
 }  
  
 **public** List<Bill> getBills() {  
 **return bills**;  
 }  
  
 **public void** setBills(List<Bill> bills) {  
 **this**.**bills** = bills;  
 }  
  
 **public** List<Staff> getSalary() {  
 **return salary**;  
 }  
  
 **public void** setSalary(List<Staff> salary) {  
 **this**.**salary** = salary;  
 }  
  
 **public** Integer getTotalCost() {  
 **return totalCost**;  
 }  
  
 **public** Integer getInterestRate() {  
 **return interestRate**;  
 }  
  
 **public void** setInterestRate(Integer interestRate) {  
 **if** (**this**.**interestRate** == **null** ){  
 **this**.**interestRate** = interestRate;  
 }**else** {  
 **this**.**interestRate** += interestRate;  
 }  
  
 }  
  
 **public void** setTotalCost(Integer totalCost) {  
 **if** (**this**.**totalCost** == **null** ){  
 **this**.**totalCost** = totalCost;  
 }**else** {  
 **this**.**totalCost** += totalCost;  
 }  
 }  
  
 **public** Integer getTotalIncome() {  
 **return totalIncome**;  
 }  
  
 **public void** setTotalIncome(Integer totalIncome) {  
 **if** (**this**.**totalIncome** == **null** ){  
 **this**.**totalIncome** = totalIncome;  
 }**else** {  
 **this**.**totalIncome** += totalIncome;  
 }  
 }  
  
 *//bill* @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addBill(Bill bill) {  
 **if** (!**bills**.contains(bill))  
 **bills**.add(bill);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewBill(Bill bill) {  
 **bills**.add(bill);  
 **if** (bill **instanceof** InternetTimeOrder){  
 **this**.setTotalIncome(((InternetTimeOrder) bill).getMoney());  
 }**else if**(bill **instanceof** ProductOrder){  
 **this**.setTotalCost(((ProductOrder) bill).getProduct().getProductCost());  
 **this**.setTotalIncome(((ProductOrder) bill).getProduct().getProductPrice());  
 }**else**{  
 **this**.setTotalCost(((EquipmentBill) bill).getEquipment().getCost());  
 }  
 **return true**;  
 *// return false;* }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addBill(List<Bill> bills) {  
 **boolean** added = **false**;  
 **for** (Bill bill : bills) {  
 **if** (!**this**.**bills**.contains(bill)) {  
 **if** (!added) added = **true**;  
 **this**.**bills**.add(bill);  
 }  
 }  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewBill(List<Bill> bills) {  
 **this**.**bills**.addAll(bills);  
 **for** (Bill bill : bills) {  
 **if** (bill **instanceof** InternetTimeOrder){  
 **this**.setTotalIncome(((InternetTimeOrder) bill).getMoney());  
 }**else if**(bill **instanceof** ProductOrder){  
 **this**.setTotalCost(((ProductOrder) bill).getProduct().getProductCost());  
 **this**.setTotalIncome(((ProductOrder) bill).getProduct().getProductPrice());  
 }**else**{  
 **this**.setTotalCost(((EquipmentBill) bill).getEquipment().getCost());  
 }  
 }  
 **return true**;  
 *//return false;* }  
  
 @DOpt(type = DOpt.Type.***LinkRemover***)  
 **public boolean** removeBill(Bill bill) {  
 **boolean** removed = **this**.**bills**.remove(bill);  
 **if** (bill **instanceof** InternetTimeOrder){  
 **this**.**totalIncome** -= ((InternetTimeOrder) bill).getMoney();  
 }**else if**(bill **instanceof** ProductOrder){  
 **this**.**totalIncome** -= ((ProductOrder) bill).getProduct().getProductPrice();  
 **this**.**totalCost** -= ((ProductOrder) bill).getProduct().getProductCost();  
 }**else**{  
 **this**.**totalCost** -= ((EquipmentBill) bill).getEquipment().getCost();  
 }  
 **return true**;  
 *//  
 // return false;* }  
  
 *// utilityCost* @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addUtilityCost(UtilityCost utilityCost) {  
 **if** (!**utilityCosts**.contains(utilityCost))  
 **utilityCosts**.add(utilityCost);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewUtilityCost(UtilityCost utilityCost) {  
 **utilityCosts**.add(utilityCost);  
 **this**.setTotalCost(utilityCost.getInternetSpending() + utilityCost.getElectricitySpending());  
 **return true**;  
 *//return false;* }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addUtilityCost(List<UtilityCost> utilityCosts) {  
 **boolean** added = **false**;  
 **for** (UtilityCost utilityCost : utilityCosts) {  
 **if** (!**this**.**utilityCosts**.contains(utilityCost)) {  
 **if** (!added) added = **true**;  
 **this**.**utilityCosts**.add(utilityCost);  
 }  
 }  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewUtilityCost(List<UtilityCost> utilityCosts) {  
 **this**.**utilityCosts**.addAll(utilityCosts);  
 **for** (UtilityCost utilityCost : utilityCosts) {  
 **this**.setTotalCost(utilityCost.getInternetSpending() + utilityCost.getElectricitySpending());  
 }  
 **return true**;  
 *//return false;* }  
  
 @DOpt(type = DOpt.Type.***LinkRemover***)  
 **public boolean** removeUtilityCost(UtilityCost utilityCost) {  
 **boolean** removed = **this**.**utilityCosts**.remove(utilityCost);  
 **this**.**totalCost** -= (utilityCost.getInternetSpending() + utilityCost.getInternetSpending());  
 **return true**;  
 *// return false;* }  
  
 *// staff* @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addStaff(Staff staff) {  
 **if** (!**salary**.contains(staff))  
 **salary**.add(staff);  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewStaff(Staff staff) {  
 **salary**.add(staff);  
 **this**.setTotalCost(staff.getSalary());  
 **return true**;  
 *// return false;* }  
  
 @DOpt(type = DOpt.Type.***LinkAdder***)  
 **public boolean** addStaff(List<Staff> staffs) {  
 **boolean** added = **false**;  
 **for** (Staff staff : staffs) {  
 **if** (!**this**.**salary**.contains(staff)) {  
 **if** (!added) added = **true**;  
 **this**.**salary**.add(staff);  
 }  
 }  
 **return false**;  
 }  
  
 @DOpt(type = DOpt.Type.***LinkAdderNew***)  
 **public boolean** addNewStaff(List<Staff> staffs) {  
 **this**.**salary**.addAll(staffs);  
 **for** (Staff s: **salary**) {  
 **this**.setTotalCost(s.getSalary());  
 }  
  
 *//* ***TODO: update totalCost & totalIncome for each staff s.salary where s in salary* return true**;  
  
 *// return false;* }  
  
 @DOpt(type = DOpt.Type.***LinkRemover***)  
 **public boolean** removeStaff(Staff staff) {  
 **boolean** removed = **this**.**salary**.remove(staff);  
 **this**.**totalCost** -= staff.getSalary();  
 *//****TODO: update totalCost & totalIncome from staff.salary* return true**;  
 *// return false;* }  
}

## UtilityCost

**package** model.utilityCost;  
  
**import** domainapp.basics.exceptions.ConstraintViolationException;  
**import** domainapp.basics.model.meta.\*;  
**import** domainapp.basics.util.Tuple;  
**import** model.report.FinanceReport;  
  
@DClass(schema = **"internetCafe"**)  
**public class** UtilityCost {  
 @DAttr(name = **"utilityCostId"**, type = DAttr.Type.***String***, mutable = **false**, optional = **true**, length = 10, id = **true**, auto = **true**)  
 **private** String **utilityCostId**;  
 **private static int** *idCounter*;  
 @DAttr(name = **"spendingDate"**, type = DAttr.Type.***String***, mutable = **true**, optional = **false**, length = 10)  
 **private** String **spendingDate**;  
 @DAttr(name = **"electricitySpending"**, type = DAttr.Type.***Integer***, mutable = **true**, optional = **false**, length = 10)  
 **private** Integer **electricitySpending**;  
 @DAttr(name = **"internetSpending"**, type = DAttr.Type.***Integer***, mutable = **true**, optional = **false**, length = 10)  
 **private** Integer **internetSpending**;  
  
  
 @DAttr(name = **"financeReport"**, type = DAttr.Type.***Domain***, length = 5, optional = **false**)  
 @DAssoc(ascName = **"FinanceReport-has-utilityCosts"**, role = **"utilityCost"**,  
 ascType = DAssoc.AssocType.***One2Many***, endType = DAssoc.AssocEndType.***Many***,  
 associate = @DAssoc.Associate(type = FinanceReport.**class**, cardMin = 1, cardMax = 1), dependsOn = **true**)  
 **private** FinanceReport **financeReport**;  
  
 **public** UtilityCost(String utilityCostId, String spendingDate, Integer electricitySpending, Integer internetSpending, FinanceReport financeReport) {  
 **this**.**utilityCostId** = nextID(utilityCostId);  
 **this**.**spendingDate** = spendingDate;  
 **this**.**electricitySpending** = electricitySpending;  
 **this**.**internetSpending** = internetSpending;  
 **this**.**financeReport** = financeReport;  
 financeReport.setTotalCost(internetSpending+electricitySpending);  
 financeReport.setInterestRate((-internetSpending-electricitySpending));  
 }  
  
 **public** UtilityCost(String utilityCostId, String spendingDate, Integer electricitySpending, Integer internetSpending) {  
 **this**(**null**,spendingDate,electricitySpending,internetSpending,**null**);  
 }  
  
 **public** UtilityCost(String spendingDate, Integer electricitySpending, Integer internetSpending, FinanceReport financeReport) {  
 **this**(**null**,spendingDate,electricitySpending,internetSpending,financeReport);  
 }  
  
 **public** UtilityCost(String spendingDate, Integer electricitySpending, Integer internetSpending) {  
 **this**(**null**,spendingDate,electricitySpending, internetSpending,**null**);  
 }  
  
 **private** String nextID(String currId) {  
 **if** (currId == **null**) {  
 *idCounter*++;  
 **return "U"** + *idCounter*;  
 } **else** {  
 **int** num = Integer.*parseInt*(currId.substring(1));  
 **if** (num > *idCounter*) {  
 *idCounter* = num;  
 }  
 **return** currId;  
 }  
 }  
  
 @DOpt(type = DOpt.Type.***AutoAttributeValueSynchroniser***)  
 **public static void** updateAutoGeneratedValue(  
 DAttr attrib,  
 Tuple derivingValue,  
 Object minVal,  
 Object maxVal) **throws** ConstraintViolationException {  
  
 **if** (minVal != **null** && maxVal != **null**) {  
  
 String maxId = (String) maxVal;  
 **try** {  
 **int** maxIdNum = Integer.*parseInt*(maxId.substring(1));  
  
 **if** (maxIdNum > *idCounter*) *// extra check  
 idCounter* = maxIdNum;  
  
 } **catch** (RuntimeException e) {  
 **throw new** ConstraintViolationException(  
 ConstraintViolationException.Code.***INVALID\_VALUE***, e, **new** Object[]{maxId});  
 }  
 }  
 }  
  
 **public** String getUtilityCostId() {  
 **return utilityCostId**;  
 }  
  
 **public void** setUtilityCostId(String utilityCostId) {  
 **this**.**utilityCostId** = utilityCostId;  
 }  
  
 **public** String getSpendingDate() {  
 **return spendingDate**;  
 }  
  
 **public void** setSpendingDate(String spendingDate) {  
 **this**.**spendingDate** = spendingDate;  
 }  
  
 **public** Integer getElectricitySpending() {  
 **return electricitySpending**;  
 }  
  
 **public void** setElectricitySpending(Integer electricitySpending) {  
 **this**.**electricitySpending** = electricitySpending;  
 }  
  
 **public** Integer getInternetSpending() {  
 **return internetSpending**;  
 }  
  
 **public void** setInternetSpending(Integer internetSpending) {  
 **this**.**internetSpending** = internetSpending;  
 }  
  
 **public** FinanceReport getFinanceReport() {  
 **return financeReport**;  
 }  
  
 **public void** setFinanceReport(FinanceReport financeReport) {  
 **this**.**financeReport** = financeReport;  
 }  
}

**REFERENCES**

[1] http://freeprojectscode.com/java-projects/cyber-cafe-management-system-project-java/2135/

[2] https://www.freeprojectz.com/java-projects-projects/java-project-cyber-cafe-management-system

[3] https://sourceforge.net/projects/cybercafejava/

[4] https://sites.google.com/site/ignoubcafinalyearprojects/project-report/cyber-caffe-management-system-project-report-1