Introduction to OO Programming

Continuous Assessment Instructions

SE-BSED (SE24)

University Souvenir Store Application

ASSIGNMENT

Introduction

To show your ability to create an object-oriented program, you will construct the Java application as detailed in this document.

Your work will be assessed not only on the basis of how well the program works, but also on how closely you have fulfilled the specification, how well you have structured your code and how well you have followed the Test Driven Development. You are advised to spend some quality time in team discussions to determine a suitable design.

Deliverables and Structure

You will deliver the following:

1. A ZIP file named **se24_*xxx*.zip** where ***xxx*** is your team number (e.g. P01, P02 etc. for Part-time teams, F01, F02 etc. for Full-time teams). The ZIP file will have the following folders:
    - **\src** folder which will contain the full set of working source files for this application, i.e. the ***.java** files, arranged hierarchically into folders according to their path. The system must compile and run without errors.
    - **\classes** folder for the compiled classes. This will be empty (i.e. class files must not be provided in the ZIP file).
    - **\test** folder for the test case classes.
      (Please follow the TDD approach for at least the important classes, excluding the GUI classes)
    - **\docs** folder which will contain the Project Report **report.doc**
    - **\data** folder which will contain the data files (see later sections)
    - **** folder which must contain
        a. **setenv.bat** file which *must contain only* the following lines:
        ```
        set JAVA_HOME=path\to\your\jdk\folder
        set PATH=%JAVA_HOME%\bin
        ```
        b. **compile.bat** file that fully compiles the sources. It must invoke **setenv.bat**, and use **JAVA_HOME** to locate the compiler.
        c. **run.bat** file that is executed to run the GUI application. It must invoke **setenv.bat**, and use **JAVA_HOME** to locate the runtime.

   Note: You may use Eclipse or any other IDE for development. But make sure that the batch files work.

2. A Project Report named **report.doc**, placed in the **\docs** folder of the ZIP file. The report cover must clearly show the name of your team and a list of members. The report must contain:
    - UML class diagrams showing the domain classes (most important classes) of your software. The diagrams may be drawn using tools of your choice (e.g. WhiteStarUML, MS Word diagrams, scanned hand drawings, Enterprise Architect tool diagrams, etc.). They must show classes, their main attributes and methods (including key parameters), their relationships with roles and multiplicity, uses of inheritance, etc. No text explanation is required for the UML

class diagrams.

Delivery and Support Mechanism

The assignment will be supported by our IVLE site (http://ivle.nus.edu.sg/). The course id is **SG4101** (Basic Software Engineering Discipline).

On the course site you will see that you have been grouped in your normal groups. The following facilities are available:

- The course Forum, which is used to discuss issues amongst yourselves and with the lecturer
- The course Workbin, where files are exchanged. In particular, the **Files** (**Workbin)** contains a folder for the peer assessments as well as two folders where **groups will upload their deliverables** (one folder for Part-time teams and one for Full-time teams)
- A Project space for each group, where the team can keep files for this project, and communicate with the facilities provided.
  (Note: This is NOT the place for the final deliverables)

Upon completion of the project, and no later than the deadlines listed below, you will upload the completed assignment deliverables to the appropriate folder in the Workbin. Only one submission is required per team.

Peer Assessment

Each member of the team is required to privately submit a peer assessment form. Download the empty form from the Files (Workbin), rename it as your matriculation number, fill in your assessment and upload it in the same folder. Your submission is private – it can only be viewed by yourself and the lecturers.

Presentation Dates

Each team will be given **20 minutes** to present their design consisting of UML class diagrams (**Domain classes**). Instructors will give feedback on your design before you start on the testing and coding activities. The presentation schedule is as follows. Note: It is important to keep to your allocated time.

Venue: Meeting Room 4-05

Monday, 7 March 2016

Team	**Time**
1FT	9.30am to 9.50am
2 FT	9.55am to 10.15am
3 FT	10.20am to 10.40am
Break	*10.45am to 11.00am*
4 FT	11.00am to 11.20am
5 FT	11.25am to 11.45am
6 FT	11.50am to 12.10pm
Lunch	*12.10 to 1.15pm*
7 FT	1.15pm to 1.35pm
8 FT	1.40pm to 2.00pm
9 FT	2.10pm to 2.30pm

Saturday, 12 March 2016

Team	**Time**
1 PT	9.30am to 9.50am
2 PT	9.55am to 10.15am
3 PT	10.20am to 10.40am
4 PT	10.45am to 11.05am
Break	*11.05am to 11.25am*
5 PT	11.25am to 11.45am
6PT	11.50am to 12.10pm
7PT	12.15pm to 12.35pm
8PT	12.40pm to 1.00pm

Final Deliverable Deadlines

Part-time teams:	**Sat 09 April 2016**
Full-time teams:	**Mon 04 April 2016**

Software Restrictions

- You may use JDK1.7 or JDK1.8 version only
- Use JUnit 4.0
- Use Swing APIs for building the GUI
- Use Java IO APIs for manipulating data from files.
  *(You may only use text or binary format files. Use of any database or other APIs such as XML etc. is **not** permitted)*

How to score high...

1. Follow the specification carefully, make the software do what it is supposed to!
2. Ensure the application works smoothly. Fix the bugs!
3. Make sure it's installable: once you've built your ZIP file, try unpacking it on a different system, in a folder with a different name, with a JDK at a different location, and verify the batch files work properly.
4. If you think you know how to make this application more impressive, enhance it as appropriate.
5. Participate in the IVLE forums and help others with their questions.
6. Do not copy code. You will lose marks for direct use of code from solutions you find elsewhere, and for sharing solutions with other teams.
7. Write test case classes for as many important classes. Test important classes thoroughly.
8. Ensure that you create ample data to demonstrate your system

Assumptions

No system requirements can be specified completely and accurately, using only the English language ☺. Hence please make reasonable assumptions, where requirements are ambiguous. Ensure that you have documented them in your report. Clarify with the instructor on critical doubts.

SYSTEM REQUIREMENTS

You are assigned to implement a simple point-of-sale system for automating the University Souvenir Store.

The system, operated by the store keeper, will provide facilities to make payment, replenish stock, check inventory, member registration. The present system is GUI-based, with flat files to store data.

The design of your system should be flexible enough to cater to new requirements and/or changes in the requirements.

Currently there is only one user of the system - the store keeper.

General Requirements

User Interface

The application will be GUI-based using Java Swing APIs.

When the user starts the system, he/she is shown a login screen. Upon successful login, he/she is taken to a main menu screen in which various options are shown. This may take the user to other screens based on the option chosen. Each of the functions described below should end by returning the user to the main menu screen.

Data Files

The data for the inventory and users, as well as transaction records, are currently stored in text-based files. The exact formats of these files is detailed in a later section. It is *mandatory* that you use those formats.

The data files must be read when the application starts up, and updated whenever needed (e.g. when a new member is added, etc). The system should have the current state persisted at all times, so that even anomalous termination of the application does not result in loss of data.

External Hardware

The system will normally use bar-code readers to obtain the product code (from the product label) and the member id (from the membership card). However, for the time being, these bar-code readers are unavailable. Operators will therefore be prompted for the codes at the console.

Similarly, there will be a receipt printer and an adhesive label printer (for printing barcode labels). For this version of the software, you should build placeholder classes for these devices, that output text to the console.

Data Requirements

Categories

The products are classified into categories (“clothing”,”mugs”, “stationary”, “diaries”, etc.). Each category is identified by a “category id”, which is a three-letter code (e.g. “CLO” for “clothing”). The list of categories is not hardcoded, but rather stored in a data file named

`Category.dat` as specified in a later section; categories can therefore be added by the store keeper.

Products

Each physical product in the store has a “product id” that identifies it uniquely. The product id is assigned automatically by the system, and consists of the category code, followed by a slash character, and a number which is assigned sequentially to products in the same category, starting from 1. For example, the twelfth product to be registered in the “clothing” category would be assigned a product id “CLO/12”. The list of products is stored in a data file named **`Products.dat`** as specified in a later section.

Vendors

The University keeps a list of vendors who supply the product. Vendors related information are stored in a data file named **`Vendors.dat`** as specified in a later section.

Transactions

Every billing transaction should be logged in a data file named **`Transactions.dat`** as specified in a later section.

Discounts

Discounts and other promotional offers are stored in a data file named **`Discounts.dat`** as specified in a later section.

Members

Staff and students can become members of the store and avail discounts. This requires submission of an application form accompanied by a document of identity (student or staff card).

Members are identified by their “member id”, which is identical to their student/staff card number. The list of members is stored in a data file named **`Members.dat`** as specified in a later section.

> **Note:** The general public are not eligible for members-only discounts. However, there may be other discounts for which they may qualify e.g. on special occasions. Furthermore, they are also not eligible to become members.

Store Keepers

User names and passwords of all store keepers are stored in a file called **`StoreKeepers.dat`**

Functional Requirements

Payments during checkout

During checkout, the system will request for payment to be effected. The system will display an itemised list of charges, with a total, and will prompt the store keeper to enter the amount tendered and/or points to be redeemed (see the section on “**Discounts, offers and loyalty points**” below). If points are redeemed during checkout, the Members data file will be updated to reflect the latest points remaining. Customers can pay using their loyalty points as well as cash or a combination of these for a single transaction.

The store keeper receives the payment and enters the payment amount and/or the number of points to redeem. The total tendered, cash as well as dollar-equivalent of the loyalty points, cannot be less than the total amount. At this point, the store keeper may decide to cancel the payment by selecting the “Cancel” option.

If a valid amount is tendered, using a combination of cash and/or loyalty points, the system will compute the change to be returned and print a receipt using the receipt printer.

Should the quantities of any product purchased by the customer result in the inventory levels falling below a specified threshold (reorder level) , an alert message is automatically printed for the store keeper indicating which products are affected and the present quantities in the inventory. Note that this message is not printed on the customer’s receipt and is strictly for the store keeper only. The reorder levels are specified in the Products data file.

Discounts, offers and loyalty points
Discounts may be offered to new members when they make their first purchase, say, 20%. The discount amount should be configurable. For subsequent purchases, members are entitled to a members-only discount of, say, 10% (also configurable).

The store will also offer discounts on special occasions such as Centenary Celebration, University President’s birthday, National Day and Orientation Day, which all customers are eligible to enjoy, not just members. The store may offer different discounts for different occasions. Note that discounts are not cumulative (e.g. 10% + 5% = 15%) – only one discount can be offered at any given time; the highest of the offered discounts will apply.

Furthermore, members may accumulate loyalty points (stored in the Members data file) and redeem the points after accumulating sufficient points. The dollar-to-point value (e.g. $10 spent = 1 additional point) and the points-to-dollar redemption value (e.g. 100 redeemed points = $5 worth) can be hard-coded if you so wish.

Check and replenish inventory
The store keeper should be able to generate a list of products with inventory quantities below a specific threshold. The threshold for each product is configured in the Products data file. The store keeper should be given an option to generate a purchase order for all items below their thresholds.

Member Registration

The new member fills out a (paper) membership application including their name and student/staff card number. The clerk requests the computer system to display the “New member“ screen and enters the form information.

The system then

- uses the student/staff card number as the membership number
- stores the member in the members database.

New Products Entry

1) When new products arrive, the store keeper requests the system to display the “New Product” screen

2) For each Product, the store keeper enters
    - the Product’s category
    - the Product’s title, in addition to other information

3) The system then
    - automatically assigns a “product id” to this product, according to the rules above
    - enters the product in the product data base

- displays an option for continuing to enter information about another product

Category Addition

1) The store keeper can add a new category requesting the system to display the “New Category” screen.

2) The system displays a list of all categories currently available in the system.

3) The store keeper is then prompted to enter
   - ♦ A three-letter code for the new category
   - ♦ A name for the category

4) The system then checks if the category code already exists, in which case it will print out an error message and terminate the function, returning to the main menu. If the category code does not already exist, the new category is added to the data file, and the user is returned to the main menu.

Buying a product

1) The store keeper requests the system to display the “Check Out” screen, and reads the bar-code on the student/staff card to determine the identity of the member.

2) The store keeper reads the product id from the bar-code sticker on each Product being purchased.

3) The member is required to make immediate payment. If payment is not received, the check-out operation is cancelled, and the user is returned to the main menu.

4) On receiving the payment, the system registers the purchase for each Product, in the Transactions data file, as specified in a later section. The user is returned to the main menu.

Reporting

The system must be able to print out to the screen some simple reports; in these reports, each record should be displayed as a brief one-liner. The following reports must be provided:

- ♦ A list of all categories, and their category id
- ♦ A list of all Products available, and their description (id, quantity, price, etc)
- ♦ A list of all transactions for a given date range sorted by product id. You should also include the product name and description in this report.
- ♦ A list of all members in the system and their details

FILE FORMATS AND SAMPLE DATA

All data files are currently text-based, so that all values are stored as strings rather than binary data. Each file contains a sequence of records of the same type.

Categories

Categories are stored in the file **`Category.dat`**.
Each line comprises comma-separated fields described as follows:
Field 1 – Three letter code of category
Field 2 – Name of category

The following is a sample **`Category.dat`** file:

```
CLO,Clothing
MUG,Mugs
STA,Stationary
DIA,Diary
```

Your category data file should contain no less than 5 categories in total.

Members

Members are stored in the file **`Members.dat`**.
Each line comprises comma-separated fields described as follows:
Field 1 – Name of member
Field 2 – Member id
Field 3 – Loyalty points accumulated so far (-1 indicates a new member)

The following is a sample **`Members.dat`** file:

```
Yan Martel,F42563743156,150
Suraj Sharma,X437F356,250
Ang Lee,R64565FG4,-1
```

Your member data file should contain no less than 10 members in total.

Products

Each line comprises comma-separated fields described as follows:
The format for each record is:
Field 1 – Product id
Field 2 – Name of product
Field 3 – Brief description
Field 4 – Quantity available
Field 5 – Price
Field 6 – Bar code number
Field 7 – Reorder Quantity (threshold)
Field 8 – Order Quantity (when inventory below reorder level)

The following is a sample **`Products.dat`** file:

```
CLO/1,Centenary Jumper,A really nice momento,315,21.45,1234,10,100
MUG/1,Centenary Mug,A really nice mug this
time,525,10.25,9876,25,150
STA/1,NUS Pen,A really cute blue pen,768,5.75,123459876,50,250
STA/2,NUS Notepad,Great notepad for those
lectures,1000,3.15,6789,25,75
```

Your product data file should contain no less than 10 products in total.

Transactions

Each line comprises comma-separated fields described as follows:
The format for each record is:
    Field 1 – Transaction id (a monotonically increasing number)
    Field 2 – Product id
    Field 3 – Member id or "PUBLIC" for non-member
    Field 4 – Quantity purchased
    Field 5 – Date

The following is a sample **`Transactions.dat`** file:

```
1,CLO/1,F42563743156,2,2013-09-28
1,MUG/1,F42563743156,3,2012-09-28
2,STA/1,PUBLIC,1,2013-09-29
3,STA/2,R64565FG4,2,2013-09-30
```

Discounts and offers

Each line comprises comma-separated fields described as follows:
The format for each record is:
    Field 1 – Discount code
    Field 2 – Description
    Field 3 – Start date (where applicable or "ALWAYS")
    Field 4 – Period of discount in days (where applicable or "ALWAYS")
    Field 5 – Percentage discount without the percentage symbol e.g. "10" for 10% discount
    Field 6 – Applicable to Member (M) or All (A)

The following is a sample **`Discounts.dat`** file:

```
MEMBER_FIRST,First purchase by member,ALWAYS,ALWAYS,20,M
MEMBER_SUBSEQ,Subsequent purchase by member,ALWAYS,ALWAYS,10,M
CENTENARY,Centenary Celebration in 2014,2014-01-01,365,15,A
PRESIDENT_BDAY,University President’s birthday,2014-02-01,7,20,A
ORIENTATION_DAY,Orientation Day,2014-02-02,3,50,A
```

Your discounts data file should contain no less than 5 discounts in total.

Vendors

Vendor list for each category of products will be kept in a separate file. A vendor may supply many categories of products. The entries for each vendor appear in the order of preference in the file. Each line comprises comma-separated fields described as follows:
The format for each record is:
    Field 1 – Vendor Name
    Field 2 – Description

The following is a sample **`VendorsMUG.dat`** file:

```
Nancy’s Gifts,Best of the best gifts from Nancy’s
Office Sovenirs,One and only Office Sovenirs
Pen’s and Such,Sovenirs and gifts for all occasions
ArtWorks Stationary Store,All kinds of Stationary and Gifts
```

Each Vendor data file should contain no less than 3 vendors in total.

Storekeepers

For this version, you do not need to store encrypted passwords.

Field 1 – Storekeeper's Name
Field 2 – Storekeeper's Password

The following is a sample **`Storekeepers.dat`** file:

```
Stacy,Dean56s
Johny,A12ysd45
```

Your Storekeepers data file should contain no less than 2 Storekeepers in total.

All the Best… Hope you enjoyed the journey of learning OOP….