Retail Store Simulation

Project

Table of Contents

Retail Store Simulation

1. Introduction

You work for a local software development company, *Dunks Simulations Inc*, as a programmer. The company creates custom made simulators for a variety of industries. You have been given the job of creating a simulator for a local retail store, *Equipment Inc*. They want to use the simulator to improve their customer service and overall business model.

Your team has decided that an object-oriented programming language is the most suitable type for this project. The team chooses the Java programming language since the customer wishes to run the simulation on a variety of platforms such as Windows and Mac. The customer also wishes to reduce the maintenance costs of the simulation.

2. Objectives

This is an **individual** assignment.

The objectives are:

- Developing your object-oriented programming skills
- Develop your Java coding skills.
- Develop team building skills.

3. Objects

The following terms are used in this documentation.

Item	Description
Store	Represents the retail store being simulated. The store services a number of customers. The store also keeps track of the total number of items sold and the total amount of money made.
Customer	Represents a customer. Each customer has a name and a specific amount of money to spend. A customer may be a member of the store's Platinum Club.
Item	Generic term that refers to items for sale. Each item has a cost, a discount value and the total number that are available.

4. Specifications

In this project you are creating a simple retail store simulator which simulates sales that take place during the day. On each day several sales take place by a set of customers. At the end of the day, the retail store restocks the items for the next day. The store sells four types of items, stoves, fringes, tables and chairs.

Each cycle of the simulator represents a single day. The default number of cycles the simulation will run is 50.

The simulation process is as follows.

Initialise all objects as necessary. The initialisation process is as follows.

a. For the store, initialize the total number of items sold to 0 and the amount of money made to 0. The overall total number of items should be maintained. You do not need to track the total per item type.

b. Create 200 customers. The name of each customer will be "Customer#" where # is the position of the customer. For example, the name of the 50th customer will be "Customer50".

   Randomly determine if the customer is a member of the store's Platinum Club.

c. For each customer, randomly generate the amount of spending money between $1000 - $5000.

   a. To generate this number, add 1000 + random number between 0 – 4000.

d. Create 100 stoves, 500 fridges, 200 tables and 400 chairs for sale by the store.

   a. For each item created, randomly choose a price between $50 - $100 and a discount between 5% - 10%.

After the initialisation process, the total number of cycles is executed. During each cycle, a set of activities are performed, as described below.

1. For each customer, randomly determine how many of each item type they wish to buy. The random number lies between 0 – 20.

   i. For each item type, if the total number required is greater than what is left, sell all of the remaining items, of that type, to the customer.

      If there are no items, of that type, left, the customer will not be able to purchase that type of item.

   ii. Calculate the total cost (cost of all the item types purchased by the customer). Note that a discount is given to Platinum Club customers or customers purchasing 10 or more items in total (overall total).

   iii. If the customer does not have enough money, move on to the next customer.

   iv. If the customer has enough money, decrease the customer's total amount of remaining money by the total cost of the items purchased.

v. For each item type, reduce the total number available by the total sold to the customer.

vi. Update the store's total items sold and total amount of money made.

2. When all customers have made their purchase, do the following.

   i. If the total number of any item type is zero, randomly restock it with a value that lies between 100 – 500.

3. If the total number of iterations executed is less than 50, go to step 1, otherwise terminate the program.

Program Output

On each iteration, you will display the total number of items sold and the total amount of money made for that iteration.

At the end of the simulation, you will display the final total number of items sold and the final amount of money made by the store.

Log

Your program must keep track of each message outputted, by your program, on each iteration. At the end of the simulation, your program must output the contents of this log to the screen.

5. Code Templates

You must use the two Java classes provided with this assignment as your starting point.

6. Testing

As part of the testing of your program, the total number of iterations will be changed. The code provided with this assignment shows how the total number iterations can be changed.

7. Further Points to Note

Design

This project's primary design approach is object-oriented programming using Java. This means that the focus is the creation of the appropriate classes.

It is tempting to cram everything in a single class but it is important to note that it is better to maintain the principle that each class should focus on one area only. For example, a class called Bicycle focuses on bicycles and not vans.

In-Code Documentation

In-code documentation refers to the commenting of code to ensure that the functionality and purpose of the code is understood. The minimum comments that should be included are:

- At the top of each .java file, there should be a description of what the class does.
- At the top of each function/method in the .java file there should be a description of the purpose of the function, the values passed to the function and what is returned by the function.
- Next to each field there should be a description of its purpose.
- Utilise sensible names for classes, fields and functions/methods that reflect their purpose.

8. Grading Rubric

This project is worth 15% of the total course mark. The grading scheme is given below.

Area	Excellent	Good	Average	Unsatisfactory
Class Decomposition (20)	The responsibilities have been well subdivided between the classes. Each class has a single purpose. The methods and fields are appropriate for each class. **(16-20 marks)**	Good subdivision of responsibilities between the classes. Most of the classes have a single purpose. The methods and fields are appropriate for most of the classes. **(11-15 marks)**	Average subdivision of responsibilities between the classes (about half). About half of the classes, have a single purpose. The methods and fields are appropriate for half of the classes. **(6-10 marks)**	Very little subdivision of responsibilities between the classes. A few of the classes have a single purpose. The methods and fields are appropriate for a few of the classes. **(0-5 marks)**
Data structures and algorithms chosen (30)	Excellent use of the appropriate data structures and algorithms.	Good use of the appropriate data structures and algorithms.	Average use of the appropriate data structures and algorithms.	Poor use of the appropriate data structures and algorithms.

	Excellent coding practices have been used with no redundant or unnecessary code. Algorithms are efficient. No unnecessary data is used. **(24-30 marks)**	Good coding practices have been used with little redundant or unnecessary code. Algorithms are mostly efficient. Small amounts of unnecessary data is used. **(16-23 marks)**	Considerable redundant or unnecessary code is present. Algorithms are generally not efficient. Significant amounts of unnecessary data is used. **(8-15 marks)**	Redundant and unnecessary code is largely present. Algorithms are not efficient. Most of the data that is used is unnecessary. **(0-7 marks)**
Implementation and Execution (30)	Over 80% of the required functionality has been implemented and executes correctly. **(24-30 marks)**	Between 50% - 80% of the required functionality has been implemented and executes correctly. **(16-23 marks)**	Between 25%-49% of the required functionality has been implemented and executes correctly. **(8-15 marks)**	Less than 25% of the required functionality has been implemented and executes correctly. **(0-7 marks)**
Compiling (10)	The software compiles with no errors and less than 3 warnings. **(10 marks)**	The software compiles with 1-6 errors and 4-6 warnings. **(7-9 marks)**	The software compiles with 7-12 errors and 7-10 warnings. **(4-6 marks)**	The software compiles with greater than 12 errors and greater than 10 warnings. **(0-3 marks)**
Naming Conventions (5)	Over 80% of the code uses the correct naming conventions. **(5 marks)**	Between 51%-80% of the code uses the correct naming conventions. **(4 marks)**	Between 26%-50% of the code uses the correct naming conventions. **(3 marks)**	Between 0%-25% of the code uses the correct naming conventions. **(0-2 marks)**
In-Code Documentation (5)	Over 80% of the code has been fully documented with comments laid out in a readable and understandable form.	Between 50%-80% of the code has been fully documented with comments mostly laid out in a readable and understandable form.	Between 25%-49% of the code has been fully documented with some of the comments laid out in a readable and understandable form.	Less than 25% of the code has been fully documented with very few of the comments laid out in a readable and understandable form.

	(5 marks)	(4 marks)	(3 marks)	(0-2 marks)
Total (100)				

Deliverables

The final deadline for this assignment is 17th March 2023 at midnight. It should be submitted on eLearning.

The deliverables are:

- The Java code in the .java files. Do not include the .class files.