23COA256: Object Oriented Programming Coursework Assignment

Dr Hossein Nevisi Semester 2

Task

Your client has requested you to design and develop a system in Java for their Computer Accessories Shop. The aim is to facilitate various shop's activities in an efficient and practical manner. The starting point for this project is a Javadoc documentation provided by the previous development team. This documentation outlines some essential classes and interfaces that you must use to build the system. However, you cannot modify these existing classes or interfaces by adding new methods, variables, constructors, or making any other changes.

The shop sells two categories of product: keyboard and mouse. The different types of keyboard that they sell are standard, flexible and gaming. The keyboards have the US layout or the UK layout. The different types of mouse that they sell are standard and gaming. The mice may have different number of buttons.

The system accommodates two user roles: admin and customer. The admin user has the right to add a new product to the stock and view all products alongside their all attributes. On the other hand, customers have the ability to view all product details except for the original cost, search/filter products, add items to their shopping basket, and complete their purchase using either PayPal or Credit Card. Credit Card transactions require a 6-digit card number and a 3-digit security code, while PayPal transactions require an email address. Additionally, customers should have the capability to cancel all items present in their shopping basket.
The stock should include, at a minimum, the products listed in Appendix B.

Functionality Two types of roles need to be created: "Admin" and "Customer". The system works with the list of users provided in a file (see Appendix A). Once the user logs in by selecting a username from the available list, they should gain access to the following functionalities in accordance with their respective roles.

- **Admin**
  - View all products alongside their attributes sorted ascending by retail price.
  - Add a product to the existing product list (stock).
- **Customer**
  - View all available products along with their attributes, excluding the original cost, sorted ascending by retail price.
  - Add items to their shopping basket.
  - View the contents of their shopping basket.
  - Complete the payment for all items in the basket by selecting one of the available payment methods:
    - ⋆ PayPal, requiring the entry of the customer's PayPal email address.
    - ⋆ Credit Card, requiring the entry of a 6-digit card number and a 3-digit security code.

    The pay function must update the stock, make the basket empty, and generate a receipt on the screen based on the selected payment method, as outlined below:

  * For PayPal, the receipt should display the content "[amount][1] paid by PayPal using [email][2] on [today][3], and the delivery address is [full address][4]".
  * For Credit Card, the receipt should display the content "[amount][1] paid by Credit Card using [card number][2] on [today][3], and the delivery address is [full address][4]".
  * Within the context provided:

  1 [amount] must be replaced by the numeric value of the total payment amount.
  2 [email] must be replaced by the customer's PayPal email address (for PayPal payment), and [card number] must be replaced by the customer's card number (for credit card payment).
  3 [today] must be replaced by today's date.
  4 [full address] must be replaced by the customer's full address, consisting of the house number, postcode, and city.

- Cancel their shopping basket which empties out the entire content of the basket.
- Search/Filter all available products in the stock using the following options and view the results (all product details must be displayed except original cost):
  A) Fast search/look-up for a given barcode.
  B) Search/Filter mice based on a given quantity of mouse buttons.

The program should be able to avoid and detect errors. Examples: (i) The customers should be notified if they want to buy an item that is out of stock. (ii) Products with a same barcode must not be added into stock more than once.

User Interface You should provide a graphical user interface (GUI) or command-line interface (CLI). A GUI generally receives more marks than a CLI. A good interface is intuitive, quick and easy to use.

Design and Implementation

Make object-oriented (OO) methodology a top priority throughout the design and implementation stages. Prior to coding, create your class diagram by identifying the required classes (including interfaces and enums), their attributes, and methods, as well as considering their interactions. Once you have a solid design, start coding by adhering to the structure defined in your class diagram. You must actively incorporate object-oriented programming principles and features into your code.

You should use Eclipse to develop the program. Your code needs to successfully compile and run within the Eclipse environment. The application should also be executable through your submitted JAR file, functioning as expected.

Note that this is an individual exercise and that you must not share any code.

Writing Understandable Code

Your code must be understandable for readers. For this purpose, you should use

- meaningful names for your classes, objects, methods and variables
- comments properly in your code
- indentation appropriately

Documentation and Submission

You are required to submit:

- Your **Eclipse coursework project folder** that should include all necessary files essential for independent compilation and execution of your program within Eclipse on any computer. This includes

- all source files (`.java` files)
- a text file names `UserAccounts.txt` that should be used for reading user details (see Appendix A), and
- a text file named `Stock.txt` that should be used for reading and writing product details (see Appendix B).

- An executable jar file of the application named "**cas.jar**".
- A representation of your object-oriented approach through a class diagram illustrating all classes, their attributes, methods, relationships, etc. This file should be named "**ClassDiagram**" and should be in either PDF or JPG format.
- A PDF document named "**NotCompleted**" that details the functionalities which have not been completed or are not functioning correctly within your application.

All files need to be zipped into a single zip file named "**YourStudentID.zip**" and submitted via Learn.

Marking

Your software will be marked based on its functionality, object-oriented design and implementation. The following serves as a general outline for how the assignment will be assessed.

- Object-oriented design and high-level definition (coding) of the classes: 38%
  - Class Diagram: 15%
  - High-level definition (coding) of the classes (including interfaces and enums) that effectively demonstrates your understanding of object-oriented principles and concepts: 23%
- User interface: 12%
- Functionality: 40%
  - Add a new product to the stock: 6%
  - View the list of all products: 6%
  - Add items to the shopping basket: 3%
  - View items in the shopping basket: 3%
  - Pay for all items in the basket and display a receipt: 10%
  - Search: A 5% and B 5%,
  - Cancel the shopping basket: 2%
  - If any of the above functions works fine but not implemented using object-oriented methodology, half of the mark will be deducted. Additionally, if their code is not understandable, one-third of the mark will be deducted.
- Validating data and detecting errors: 10%

The following requirements are absolutely essential. If any of them is not fulfilled, you will lose the main part of the mark.

- Your code must compile and run.
- There shouldn't be more than one class definition in each file.
- Object-oriented design principles must be used throughout the development process.
- Your code implementation must strictly adhere to the structure defined in your class diagram.
- You are not allowed to make any changes to the classes/interfaces specified in the provided Javadoc documentation. You must implement those exactly as described in the documentation.
- You should only use text files (.txt) to read/write data from/into.
- You should use platform-independent file paths when accessing files from your computer.
- The user interface cannot be a combination of CLI and GUI.
- The jar file must run either by double-clicking on it or by calling `java -jar cas.jar`.
- The implemented functionality must be usable via an interface.
- The interface is easy to work for the users work with the system for the first time.

Please be aware:

- All files essential for independent compilation and execution of your program within Eclipse must be provided. While you can develop your code on any operating system (including Mac operating system), the compilation and execution for assessment will be conducted on a Windows machine.
- Failure to submit the code or encountering compilation errors, along with the absence of a functional jar file, will prevent the program from being tested and consequently will not meet the necessary criteria for passing.

Plagiarism

Note that this is an individual exercise and that you must not share any code. Failure to do so would leave you open to prosecution for Academic Misconduct.

A User Accounts

The initial list of valid users of the system (which must be stored in UserAccounts.txt) should be as follows, where each user is given as a comma-separated list in the form: `user ID, username, name, house number, postcode, city, role`.

```
101, user1, Daniel, 12, LE11 3TU, Loughborough, admin
102, user2, Fen, 14, E20 3BS, London, customer
103, user3, Emma, 100, BN1 3XP, Brighton, customer
104, user4, Bowen, 57, PA3 2SW, Glasgow, customer
```

B Stock Data

The initial list of products in the stock (which must be stored in Stock.txt) should be as follows, where each product is given as a comma-separated list in the form: *barcode, product category, device type, brand, colour, connectivity, quantity in stock, original cost, retail price, additional information*. The additional information for keyboards is their layout (either UK or US) and for mice is number of their buttons.

```
112233, mouse, gaming, Logitech, black, wireless, 15, 7.50, 9.50, 3
123456, keyboard, gaming, Corsair, black, wired, 2, 30.0, 39.99, UK
124455, keyboard, standard, Advent, white, wired, 10, 3.50, 5.99, UK
124566, mouse, standard, Advent, grey, wired, 15, 2.50, 4.99, 2
125567, keyboard, flexible, Logitech, black, wireless, 4, 25.99, 30.0, US
221101, mouse, standard, Logitech, black, wired, 3, 3.0, 6.99, 3
221122, mouse, gaming, Razer, black, wired, 1, 28.0, 40.99, 7
223044, mouse, gaming, Anker, blue, wired, 3, 16.0, 18.50, 10
234555, keyboard, standard, Apple, white, wireless, 10, 75.50, 85.50, US
235066, keyboard, flexible, Microsoft, black, wireless, 5, 45.50, 60.0, UK
236677, mouse, standard, Asus, blue, wireless, 8, 10.0, 16.99, 5
237700, mouse, gaming, Anker, black, wired, 1, 8.0, 11.99, 7
237788, keyboard, standard, Logitech, grey, wired, 15, 6.50, 8.99, UK
237799, mouse, standard, Logitech, grey, wireless, 13, 7.0, 8.0, 2
238800, keyboard, flexible, Microsoft, black, wired, 10, 26.50, 30.0, US
```