University of Westminster
School of Computer Science & Engineering

5COSC019C	Object Oriented Programming – Coursework (2023/24)
Module leader	Saman Hettiarachchi
Unit	Coursework
Weighting:	50%
Qualifying mark	30%
Description	Object Oriented Programming and Design
Learning Outcomes Covered in this Assignment:	This assignment contributes towards the following Learning Outcomes (LOs): - LO1 Identify and justify good practices in the development of object oriented software; - LO2 Apply acquired knowledge of concepts, characteristics, tools and environments to adapt to new computational environments and programming languages which are based on object oriented principles; - LO3 Design, implement efficiently applications based on a OOP language, given a set of functional requirements; - LO4 Implement GUI interfaces using an OOP language; - LO5 Apply appropriate techniques for evaluation and testing and adapt the performance accordingly
Handed Out:	6th November 2023
Due Date	Thursday 13th January 2024 Submissions by 13:00
Expected deliverables	Submit on Blackboard a zip file containing: **1) A folder with all the UML documents, test case plan and report attached** **2) A folder with the developed project (NetBeans Solution with your Java code)** **3) A link to a video where you recorded a demonstration of the functionalities of the implemented system.**
Method of Submission:	Electronic submission on BB via a provided link close to the submission time.
Type of Feedback and Due Date:	Written individual feedback within 15 working days. **All marks will remain provisional until formally agreed by an Assessment Board.**

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or nonsubmission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/currentstudents/resources/academic-regulations**

Coursework Description

Objectives
The aim of this coursework is to assess the knowledge and skills that you have acquired about object-oriented programming during the module. You are asked to implement a program in which objects interact in order to fulfil a set of functional requirements.

Analyse the problem statement
An important skill that you will start to develop in this module is analysing a problem statement in order to identify the details needed to develop a solution.
In this assignment, the first task you should perform is a careful analysis of the problem statement in order to make sure you have all the information to elaborate a solution. If you have any questions, please write your queries on the forum on BB.

Problem description and requirement statement
You are required to develop a program to manage an online shopping system.

You should implement a **console system** from where the manager can add new products, delete if needed, print and save them as described in detail below.

You should implement a **Graphical User Interface** (GUI) from where a user can select different products, add them to the shopping cart and visualise the final price, etc., as described below.

For the user interface, you are not allowed to use drag-and-drop tools (such as the Designer in NetBeans), and you cannot use Java FX, but you can use some external API if you want to add graphs or some more professional components.

In this assignment, you will be required to address the following tasks:

1. Design and class implementation (Phase 1)
The design of your system should be consistent with the Object-oriented principles and easy to understand by an independent programmer.

You are required to design your program using UML diagrams. In particular, you have to draw:

- A UML use case diagram for the system *(6 marks)*.
- A UML class diagram *(6 marks)*

Read carefully the following requirements. It is important that you follow the specifications, and your design and implementation must comply with these.
According to the *Inheritance and encapsulation* principles, you have to design and implement a super-class **Product** and the subclasses **Electronics** and **Clothing**. Also, you will have to add at least one *constructor* for each class.

> The class **Product** should be **abstract** and include appropriate get/set methods and hold information about the *product ID* (mix of characters and numbers), *product name, number of available items* and the *price (4 marks).* (You can add any other information that you consider appropriate).

In particular:

- The **Electronics** subclass should hold specific information and methods. You should add the *brand* and *the warranty period* as instance variables, constructors, and the relative get/set methods *(3 marks)*.

- The **Clothing** subclass should hold specific information and methods. You should add the *size* and *colour* as instance variables (attribute), constructors and the relative get/set methods *(3 marks)*.

- You should implement a class **User** to represent the user account. The class should hold information about the *username and password*, constructors, and the relative get/set methods *(3 marks)*.

- You should implement a class **ShoppingCart** to represent the user's cart. The class should contain a *list of products* as instance variable, there should be methods to add, remove and calculate the total cost *(3 marks)*.

- Design and implement a class called **WestminsterShoppingManager**, which implements the *interface* **ShoppingManager** *(2 marks).* WestminsterShoppingManager maintains the list of the products in the system and provides all the methods for the system manager defined in the console menu.

2. Console Menu Implementation (Phase 2)

The class WestminsterShoppingManager should display **a menu in the console (not in the GUI)** containing the following management actions from which the manager can select one. – *You should have a menu with a list of options*

- **Add a new product** to the system. It should be possible to add either electronics or clothing, with all the relevant information (all the attributes defined in the relative class). You should consider that the system can have a maximum of 50 products *(5 marks).*

- **Delete a product** from the system, inserting the product ID. Display a message with the information of the product (if it is electronics or clothing) that has been deleted and the total number of products left in the system *(5 marks)*.

- **Print the list of the products** in the system. For each product, print on the screen all the information (attributes defined in the corresponding class) and say if it is electronics or clothing. The list should be ordered alphabetically according to the product ID *(6 marks)*.

- **Save in a file** the list of products that have been added to the system, with all the relative attributes. The next time the application starts, it should be able to **read back all the information** saved in the file and continue to use the system (*6 marks*).

3. Graphical User Interface (GUI) Implementation (Phase 3)

Whilst the manager interacts with the system through the menu console, the client will interact with the system through a graphical user interface.

Open a Graphical User Interface (GUI) from the menu console – *you will have an additional option that the user can select to open the GUI.*

Note: The GUI should look like the mock-up provided below. If your GUI appears different, no marks will be given.

You should implement the GUI according to the following requests:

- The user can select from a drop-down menu which type of product can be visualised (all, Electronics, or Clothes) *(4 marks).*
- The user can visualise the list of products with relative information as displayed below.

The user should be able to sort the list alphabetically. You should use a table to display this information on the GUI. *(6 marks)*.

- The user can select a product and add it to a Shopping Cart. When implementing these functionalities, you need to comply with the following requirements:
    - The items with reduced availability (less than 3 items available) should be in red on the table. *(2 marks).*
    - When the user selects the product, the product details (all the information related to the product) should appear in a panel below the table *(5 marks).*
    - The user can add the item to the shopping cart by clicking the relative button. The user can visualise the shopping cart by clicking the "Shopping Cart" button. The user can select another item and keep adding items to the shopping cart *(6 marks).*
    - The shopping cart will show the list of products and the final price *(6 marks).* To calculate the final price, the system will apply the following discounts:
- 20% discount when the user buys at least three products of the same category.
- 10% discount for the very first purchase. – *in order to implement this functionality, you will have to keep the history of the purchases made by each client. You can do this in different ways: if you would like to add extra classes, you can do so, and this will be reflected in your class diagram too.*

- If the discount is applied, the message will appear on the GUI showing the final cost as displayed below.

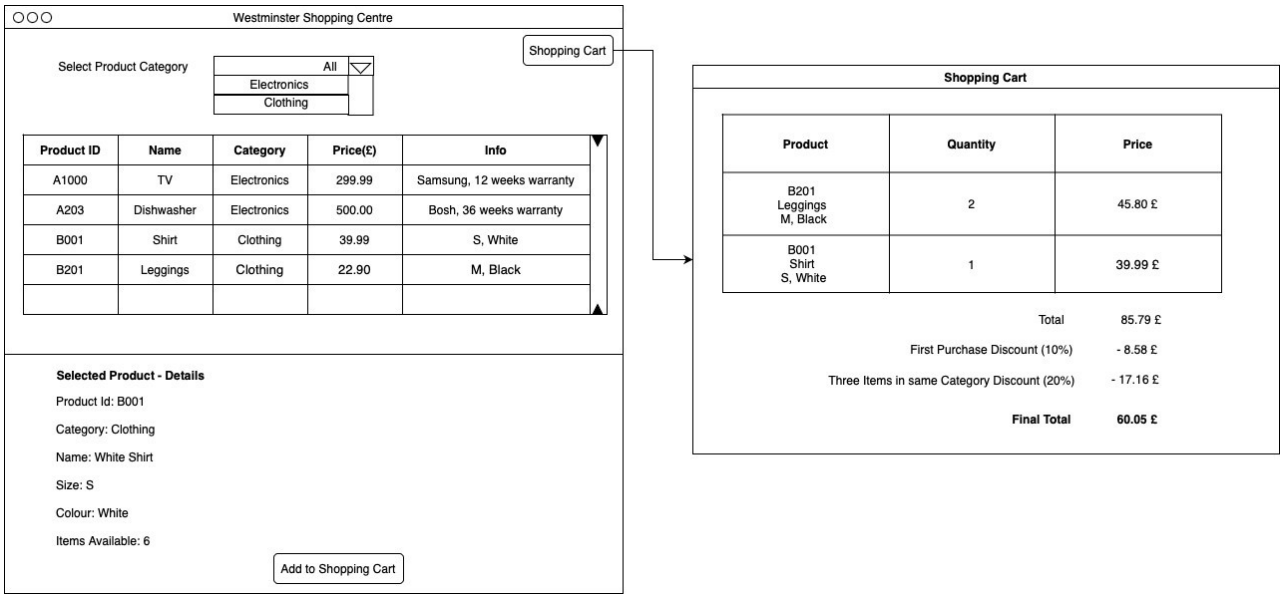

Figure 1 Mock-up for the GUI

4. Testing and system validation (Phase 4)

- **Write a test plan** designed to ensure that the coded solution works as expected. The test plan should include specific instructions about the data and conditions the program will be tested with *(4 marks).*

- Implement automated testing (you can use JUnit or feel free to use any other tool or scripts for unit testing) that runs scenarios of the main use cases you implemented in the console menu *(4 marks)*.

- The following will be evaluated: ○ The robustness of the code through the use of error handling and input validation *(3 marks).*
    - The quality of the code and the adherence to coding standards and conventions *(3 marks).*

Coursework Report
You are requested to fill out a report about your work. You should download the template from Blackboard and fill in the sections with your comments.
Note that if the Report is not submitted your mark will be capped to 30%.

VIDEO – (5 marks for the quality of your video)
You are requested to record a video of the system you implemented. You need to run the project and show all the functionalities you implemented in the console menu and in the GUI. Add at least 5 products during the demonstration. You should also record your voice explaining what you are demonstrating. **Note that if the video is not submitted, your mark will be capped to 30%.**

VIVA – Demonstration
You will have to demonstrate the understanding of the code implementation and your final mark will reflect this. Also, your tutor is entitled to request a misconduct case if there is doubt about the originality of your work!