MTU
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

OBJECT ORIENTED PRINCIPLES.

A03: PROJECT – A Java Application for Managing a Store.

Background.

In this project you are going to develop a Java Application for managing a store, so as to demonstrate your understanding of the OOP concepts seen during the semester.

As a difference with the previous labs, this is an open project: just some main requirements are given to you, which you must follow. After that, it is completely up to you to decide the design and implementation of your Java application.

To help you a bit, an Example Java Application is given. You can find the source code and Javadoc of this application in the folder A03_Example_Java_Application. Although it does not manage a store (but, instead, a public library) you can use its functionality for reference when designing and developing your own Java Application. Please see the Example Report provided in the folder A03_Example_Report, to see how the Example Java Application demonstrates the OOP concepts seen this semester.

Main requirements specification.

1. Your Java Application must manage a store.
    - It can be a store of any type and size you want (for example, a coffee truck, a veterinary shop, an art gallery, a grocery shop, an online store, etc.)
    - Given the example application provided, your own application cannot be a book store or a music store.
2. Your Java application must model (via Java classes) the concepts of:
    - A customer - representing a customer of the store.
    - A product - representing a product of the store.
    - An order - representing a customer purchasing a product of the store.
3. Your Java application must include a class MyMain.java testing its functionality.
4. Your Java application must include as many of the OOP concepts seen in class as possible, the more the better.
   The whole list of OOP concepts is described below:
    1. Primitive and Reference Variables.
    2. Classes and Objects.
    3. Encapsulation.
    4. Aggregation.
    5. Inheritance.
    6. Class Hierarchy.
    7. Static Polymorphism (overloading).
    8. Dynamic Polymorphism (overwriting).
    9. Abstract Class.
    10. Interface.
    11. User and Developer Isolation.
    12. Upcasting.
    13. Static Fields and Methods.
    14. Final Fields, Methods and Classes.
    15. Data Structures.
    16. Java Generics.
    17. Downcasting.
    18. Exception Handling.
    19. File Reading and Writing.
    20. Default Constructor and Copy Constructor.

   Please see the Example Report provided in the folder A03_Example_Report, which demonstrates the application of these OOP concepts to the Example Java Application.

5. You must submit a report of at most 1000 words describing:
    - A brief description of the store you are managing, and its main functionality.
    - The OOP concepts applied/demonstrated in your code.
    - A UML diagram of the different classes used in the application.
    - A brief description of how you are testing the functionality in the class MyMain.java

   Please see the Example Report provided in the folder A03_Example_Report for reference when writing your report.

Submission Details.

- Submission Deadline: Sunday 8th of December, 11.59pm (Week 12 of the semester).
- Please submit a zip file including:
    - The folder A03_Student_Java_Application, including the source code of your Java Application (subfolder scr) and its Javadoc documentation (subfolder javadoc).
    - The folder A03_Student_Report, including your 1000 words max report (file A03_Student_Report.docx).
- A 15 minutes video is to be recorded as part of the submission. On the video, you can demo the project, and present its main functionality, as well as any aspects of the implementation that you would like to highlight. Please upload the video to YouTube, and include a link to it in your A03 report. The video is mandatory for the project to be evaluated.

Rubric.

- Description of the type of store managed and its functionality (10%).
- Modelling, via Java classes, of the concepts customer, product and order (10%).
- Testing, via the class MyMain.java, of the application's functionality (10%).
- Code readability (10%).
- UML design (10%).
- Technical difficulty: specifically, the number of OOP concepts included in the code and explained in the report (50%).