OOP project 2025 Spring

Objective:

This is an individual project. You must work on this project on your own. The goal of this project is to design and implement a system using Java, with Java Swing for the graphical user interface (GUI). The system will be based on a real-world scenario. Select **one** of the five topics listed below. The project should demonstrate your understanding of core Java concepts, including object-oriented programming (OOP), file handling, and data structures.

Topics:

1. **Online Appointment Scheduling System**: A system for booking appointments (e.g., for doctors, salons) that allows users to create, view, modify, and cancel their appointments.
2. **Travel Agency Management System**: A platform for managing travel bookings, itineraries, and customer information, complete with functionalities to add trips, browse packages, and record customer reviews.
3. **Event Ticketing System**: Create an application for users to browse events, purchase tickets, and manage their bookings. Include features for event organizers to create and manage events.
4. **Inventory Management System for a Retail Store**: This system would help manage products, suppliers, and customer orders, with functionalities for tracking stock levels and generating sales reports.
5. **Real Estate Property Management System**: A platform for managing properties available for rent or sale, allowing realtors to list available properties and potential tenants or buyers to search for properties based on criteria.

Based on the topic you select, you need to design your system to provide meaningful features of the system. Any features not mentioned is welcome as long as they are valid scenarios.

Requirements:

1. System Scope

Your system should model a real-world scenario. The system should allow users to perform CRUD operations (Create, Read, Update, Delete) on data entities (e.g., customers, products, transactions, etc.).

2. OOP Concepts

- Use at least one interface to define a set of methods that must be implemented by multiple classes.
- Use at least one abstract class to provide common functionality for related classes.
- Demonstrate inheritance by creating a hierarchy of classes (e.g., a base class and at least two subclasses).
- Use at least 6 interconnected classes.
- Use method overloading.
- Use method overriding.
- Use polymorphism to allow objects of different classes to be treated as objects of a common superclass.

- Use Exception handling to handle error messages. There must be at least two user defined exception classes.

3. Data Storage

- Use file I/O to store and retrieve data persistently (e.g., save customer details, transaction history, or inventory to a file).
- The system should load data from a file when the program starts and save data back to the file when the program exits.
- You may also choose to use a database to store your data. No extra credit is provided in this case.

4. Data Structures

Use ArrayList or other Java Collections Framework classes to manage collections of objects (e.g., a list of customers, products, or orders). Therefore, a total amount of orders/deposits can be shown as a output on GUI.

5. GUI Requirements

Implement a Java Swing GUI with a main window that allows users to interact with the system.
The main window should include:

- Input fields (e.g., text fields, dropdowns, buttons) for users to perform CRUD operations.
- A display area (e.g., a JTextArea, JTable, or JList) to show output (e.g., error messages, lists of customers, transactions, or inventory, etc.).
- All primary functions should be operated within the main window. Temporary windows (e.g., dialog boxes) may be used for user input when necessary, but they must close immediately after serving their purpose.
- Ensure the GUI is user-friendly and intuitive (e.g., include labels, clear layouts, and error messages for invalid input).

6. Some features may make your project more attractive

- Implement sorting or searching functionality (e.g., sort customers by name, search for a product by ID).
- Add features to do data analysis according to the data the system collects.

Submission (on iSpace):

1. **Proposal** (**due date April 30th 11:59pm**):

Write a short proposal in a **PDF** file (2-3 pages) explaining:

- The real-world scenario you chose and the business logic for the scenario.
- Explain your design of the required OOP concepts (interface, abstract class, inheritance, etc.). You may use UML diagram or detailed explanation of your design.
- Draw your GUI window interface, and explain the features involved in the system.
- Explain how the IO is designed.

2. **Source Code** (**due date May 20th 11:59pm**):

Submit all Java source files (.java) with proper comments.

3. **Demo Video** (**due date May 20th 11:59pm**):
Record a video (less than 5 minutes) to demonstrate your system. While demonstrating your system, you must use your voice to explain the steps you are showing. You don’t have to show your face, but we will check if you are demonstrating it yourself. Plagiarism will be considered if we find the person demonstrating your system is not yourself.