COMSATS University Islamabad
Sahiwal Campus

Department of computer science
Assignment # 3

Subject: Object Oriented Programming **Couse code:** CSC241 **Due Date:** 1/12/2025
Instructor: Anam Khan **Class:** SP25-BCS-A **Total marks:** 20

CLO3: Apply event handling model to develop event driven programs that respond to user events.

Instructions:

- This will be a group assignment. Group leaders of semester project will not do this assignment and will get marks equal to max marks of other group members.

Question 1: Implement following scenarios using core OOP and Swing GUI concepts:

Scenario 1: Shipping Cost Calculator

Part 1: Coding (OOP Concepts)

Design and implement a Java application for a shipping cost calculation system. Your solution must demonstrate the following OOP concepts:

- **Inheritance:** Create an abstract base class `ShippingType` with an abstract method `calculateCost(double distanceKm, double itemWeightKg)` and a method `getType()`. Then, create two concrete subclasses:
    - `StandardShipping`: This class should inherit from `ShippingType`. Its `calculateCost` method will calculate the cost based solely on the distance, using a fixed rate per kilometer (e.g., Rs 50 per km).
    - `ExpressShipping`: This class should also inherit from `ShippingType`. Its `calculateCost` method will calculate the cost based on both distance (using the same fixed rate per km) and an additional factor related to the item's weight (e.g., $0.25 per kg).
- **Polymorphism:** Ensure that the `calculateCost` method is used polymorphically, allowing the system to calculate costs for different shipping types through a common interface.

Part 2: GUI Design and Functionality

Develop a Java Swing GUI application that interacts with the OOP classes designed in Part 1. The GUI should include:

- Input fields for the user to enter the **item's weight (in kg)** and the **distance (in kilometers)** between the source and destination.
- Radio buttons or a dropdown menu to allow the user to select the **shipping type** (Standard or Express).
- A "Calculate Cost" button that, when clicked, will:
    - Read the user inputs.
    - Create instances of the appropriate `Item`, `ShippingType` (based on selection)
    - Call the `getTotalCost()` method to get the calculated cost.
    - Display the total shipping cost clearly in a designated area on the form.

Scenario 2: Amenity Cost Calculator

Part 1: Coding (OOP Concepts)

Design and implement a Java application to manage and calculate costs for various amenities offered in a system (e.g., a hotel or resort). Your solution must demonstrate the following OOP concepts:

- **Inheritance:** Create an abstract base class `Amenity` with attributes for `name` (e.g., "Wi-Fi", "Breakfast") and `baseCost`. It should have an abstract method `calculateCost(int daysStayed)`. Then, create the following concrete subclasses:
    - `BasicAmenity`: Inherits from `Amenity`. Its `calculateCost` method should return the fixed `baseCost`, as days stayed do not affect its price.
    - `PackagedAmenity`: Inherits from `Amenity`. In addition to `name` and `baseCost`, it should have a `description` attribute (e.g., "Includes pool access, gym, and sauna"). Its `calculateCost` method should also return the fixed `baseCost`.
    - `LuxuryAmenity`: Inherits from `Amenity`. Its `calculateCost` method should calculate the cost based on its `baseCost` multiplied by the `daysStayed` and an additional per-day charge factor (e.g., `baseCost + daysStayed * 1.2`).
- **Polymorphism:** Ensure that the `calculateCost` method is used polymorphically, allowing the system to calculate costs for different amenity types through a common interface.

Part 2: GUI Design and Functionality

Develop a Java Swing GUI application that allows users to interact with the amenity system. The GUI should include:

- Input fields for the **amenity name** and **base cost**.

- A dropdown menu or radio buttons to select the **amenity type** (Basic, Packaged, Luxury).
- An input field for **days stayed** (this field should only be relevant and perhaps enabled/disabled based on the selected amenity type, especially for `LuxuryAmenity`).
- A text area or input field for the **description** (this field should only be relevant and perhaps enabled/disabled when "Packaged Amenity" is selected).
- A "Calculate Amenity Cost" button that, when clicked, will:
    - Read the user inputs.
    - Create an instance of the appropriate `Amenity` subclass based on the selected type.
    - Call the `calculateCost()` method on the created `Amenity` object.
    - Display the calculated total amenity cost in a dedicated label or text area on the form.

The End