National Institute of Business Management School of Computing and Engineering Course work | Assesement Announcement Sheet	

Course Name	KIC-DCSAI24.1F/PE		
Module Name	Object Oriented Programming with Java		
Batch	24.1F/PE		
Learning Outcomes Covered (Mention according to the Module Descriptor)			
Assesement | CW No	1		
Assesement Mode	**Group**	**Group (if it is group mode only)**	
		Group Size	**Grouping Criteria**
		-	Lecturer will decide.
Assesement Type	**Practical Test | ~~Report~~ | Software | ~~Presentation~~ | VIVA | ~~MCQ~~**		
	If other specify		
Hand in Date | Time	12/5/2024		
Hand out Date | Time	12/5/2024		
Submission Details (Format and Location)	LMS-Git hub Link		
Plagiarism Criteria			

Assesement | CW Description

Part A

Scenario 1

You are tasked with developing a simple employee management system for a small company. The company wants to keep track of its employees' information securely. The following requirements must be addressed:

1. Each employee has an ID, name, department, and salary.
2. The company does not want to expose employees' salaries directly to anyone. Instead, they want a method to provide a calculated bonus (e.g., 10% of the salary) without exposing the salary value.
3. Employee details such as ID, name, and department can be viewed but should be updated only through specific methods to maintain data integrity.
4. Implement validation to ensure:

- Employee ID is a positive integer.
- Salary cannot be less than zero.

Task:
Write a Java program that demonstrates the concept of encapsulation. Implement the following:

1. A class Employee with private fields for ID, name, department, and salary.
2. Public getter and setter methods to:
    - Access and update the employee's name and department.
    - Update salary (with validation to ensure it is not negative).
    - Access a calculated bonus without exposing the actual salary value.
3. A test class EmployeeManagement to create employee objects, set their details, and demonstrate the functionality of the encapsulated fields.

Scenario 2

A company, Global Freight Services, operates in the logistics industry. They handle both domestic shipments and international shipments. The company uses a software system to manage shipment details.

The base class is Shipment, which has the following attributes:

- shipmentId (unique identifier for the shipment)
- weight (weight of the shipment in kilograms)
- destination (city/country the shipment is being sent to)

The company handles two types of shipments:

1. DomesticShipment
    - Additional attribute: region (the local region where the shipment is going, e.g., "North", "South").
    - Tax for domestic shipments is calculated as 5% of the shipment cost.
2. InternationalShipment
    - Additional attributes: customsFee (fee for crossing international borders) and insuranceFee (fee for insuring the shipment).
    - Tax for international shipments is calculated as 15% of the shipment cost.

Task:
Write a Java program using inheritance to represent the above scenario. Perform the following:

1. Create the base class Shipment with necessary constructors, getters, and setters.

2. Create the derived classes DomesticShipment and InternationalShipment.
3. Override a method calculateTotalCost() in each subclass to calculate the total cost of the shipment, including the tax.
4. In the main method:
    - Create instances of DomesticShipment and InternationalShipment.
    - Display the total cost for each shipment.

Scenario 3

A logistics company, GlobalTrans, manages various types of vehicles for transporting goods. Each vehicle type (e.g., trucks, ships, and airplanes) has a unique way of calculating the transport cost based on distance and weight. The company wants to create a Java application that utilizes polymorphism to calculate transport costs for different vehicles.

Question:

1. Design a class hierarchy to represent the above scenario. The base class Vehicle should define common properties such as distance and weight. It should also define a method calculateCost() to be overridden by the subclasses Truck, Ship, and Airplane.
2. Implement the following in Java:
    - A base class Vehicle with attributes distance (in km) and weight (in kg).
    - The calculateCost() method in Vehicle that returns 0 (to be overridden).
    - Subclasses Truck, Ship, and Airplane that override the calculateCost() method with specific cost formulas:
        - Truck: Cost = distance * 5 + weight * 2
        - Ship: Cost = distance * 3 + weight * 1.5
        - Airplane: Cost = distance * 10 + weight * 5
3. Write a Java main method that demonstrates polymorphism by:
    - Creating a list of Vehicle objects, including Truck, Ship, and Airplane.
    - Calculating and displaying the transport cost for each vehicle using a loop that calls the calculateCost() method.

Part B

The purpose of this Part B is to develop a standalone Java Swing application, focusing on GUI development and application flow without database connectivity. Students are encouraged to use their creativity while adhering to the requirements.

Requirements:

1. **Business Scenario Selection:**
    - Students must select a business scenario of their choice (e.g., a library management system, ticket booking application, inventory management system, etc.).
    - The selected scenario should be realistic and manageable within the given timeline.
    - The chosen scenario must be approved by the lecturer before proceeding with the development.
2. **Application Features:**
    - **Login Screen:**
      The application should begin with a login screen that requires users to enter valid credentials to access the application. Hardcoded credentials can be used since database connectivity is not required.
    - **Menu Screen:**
      After successful login, the user should be navigated to a menu screen. This screen should provide options for accessing different features of the application, relevant to the selected business scenario.
    - **Business-Specific Screens:**
      Develop multiple screens to cover the functional requirements of the chosen scenario. Examples include adding, editing, and viewing records or generating simple reports.
    - **Completion of Business Scenario:**
      Ensure that all core functionalities of the selected scenario are implemented fully and demonstrate a complete workflow.
3. **Design and Usability:**
    - Use Java Swing components to create a user-friendly graphical interface.
    - Follow proper application design principles, including modularity and reusable components where applicable.
4. **Documentation:**
    - Provide brief documentation (maximum 2 pages) including:
        - Description of the business scenario.
        - Application features and navigation flow.
        - Any challenges faced during development and how they were addressed.

Evaluation Criteria:

- **Functional Completion:**
  The application should fulfill all requirements and demonstrate a complete workflow based on the selected business scenario.
- **User Interface Design:**
  The UI should be intuitive, clean, and functional.

● **Code Quality:** Code should follow proper conventions and demonstrate the principles of object-oriented programming. ● **Creativity:** Innovative approaches to implementing the business scenario and user interface design will be rewarded.