



Daffodil International University

Faculty of Science & Information Technology

Department of Computer Science & Engineering

Midterm Examination, Fall 2026

Course Code: CSE221 , Course Title: Object Oriented Programming

Level: 2 Term: 2 Batch: 69

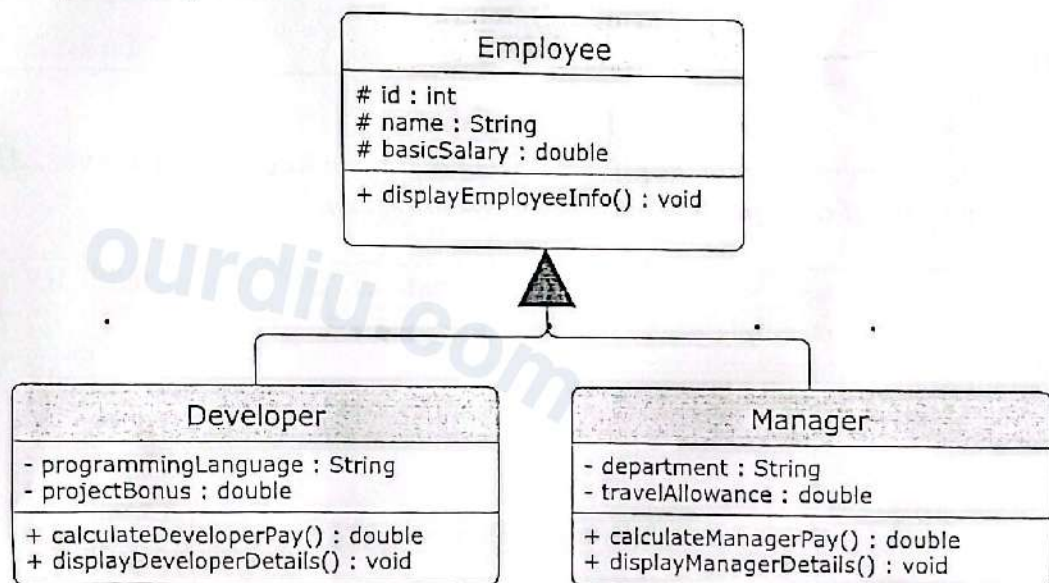
Time: 01:30 Hrs

Marks: 25

Answer ALL Questions

[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1.	a)	Explain instance and class variables in the context of Object-Oriented Programming in Java with a simple code example.	03	CO1
	b)	Trace the output of the following Java program. <pre> class Demo { Demo() { this(10); System.out.println("Default Constructor"); } Demo(int x) { this(x, 20); System.out.println("Single Parameter: " + x); } Demo(int x, int y) { System.out.println("Sum: " + (x + y)); } } public class TestOutput { public static void main(String[] args) { Demo obj = new Demo(); } } </pre>	02	
2.		A local grocery store is developing a software system to manage items on its shelves, and you are assigned to design the Product class by applying the fundamental OOP concept of encapsulation. The class tracks three details for every item: a product code (String), a product name (String), and its unit price (double). To maintain data security and avoid accidental tampering, these attributes remain hidden from direct outside access. The system allows registering an item in two ways: either by providing only the product code and name (leaving the initial price as 0.0), or by providing all three details (code, name, and price) together. To keep store records consistent, the product code is permanent once assigned—it can be viewed at any time, but cannot be modified. On the other hand, store staff can view and update both the product name and its unit price whenever necessary. The system also includes an execution entry point inside a Main class, which handles registering products into the system, updating their details, and retrieving their final information for display.		

a)	Based on the given scenario, build a complete UML class diagram representing both the Product and Main classes with accurate attribute types, visibility notations, overloaded constructors, and method signatures.	04	CO2
b)	Implement the Java program strictly following your diagram where the Main class creates products using both constructors, updates the price of the product, prints the details, and shows the expected console output.	06	
3. Carefully examine the following UML class diagram designed for an Employee Management System.			
 <pre> classDiagram class Employee { # id : int # name : String # basicSalary : double + displayEmployeeInfo() : void } class Developer { - programmingLanguage : String - projectBonus : double + calculateDeveloperPay() : double + displayDeveloperDetails() : void } class Manager { - department : String - travelAllowance : double + calculateManagerPay() : double + displayManagerDetails() : void } Employee < -- Developer Employee < -- Manager </pre> The diagram shows a class hierarchy where Employee is the superclass and Developer and Manager are subclasses. Employee has three private attributes: <code>id</code> (int), <code>name</code> (String), and <code>basicSalary</code> (double). It has one public method: <code>displayEmployeeInfo()</code> (void). Developer has two private attributes: <code>programmingLanguage</code> (String) and <code>projectBonus</code> (double). It has two public methods: <code>calculateDeveloperPay()</code> (double) and <code>displayDeveloperDetails()</code> (void). Manager has two private attributes: <code>department</code> (String) and <code>travelAllowance</code> (double). It has two public methods: <code>calculateManagerPay()</code> (double) and <code>displayManagerDetails()</code> (void).			
a)	Identify the type of inheritance represented in the provided UML class diagram and justify your answer.	02	CO3
b)	Apply Java programming language to implement the classes and their relationships based on the provided UML class diagram. Note that <code>calculateDeveloperPay()</code> must return the sum of <code>basicSalary</code> and <code>projectBonus</code> , while <code>calculateManagerPay()</code> must return the sum of <code>basicSalary</code> and <code>travelAllowance</code> . Finally, create a driver class named Main containing the <code>main()</code> method to instantiate objects of both subclasses, assign appropriate sample values to their attributes, and invoke their respective display methods to show the final output on the console.	08	