



Daffodil International University
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Final Examination, Spring 2025

Course Code: CSE221, Course Title: Object Oriented Programming

Level:2 Term:2 Batch: ALL

Time: 2 Hours

Marks: 40

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 the concept of Inheritance, Abstract Class, and Polymorphism in Java. How do they work together to create reusable and maintainable code?	5	CO1
	b)	Scenario: A company manages its employees using an Employee Management System. The system consists of: • Base Class (Employee) with attributes: name, ID, salary. • Abstract Class (PermanentEmployee) which extends Employee and has additional attributes: bonus. • Concrete Class (ContractEmployee) which extends Employee and has an attribute contractPeriod. • Implement Polymorphism by overriding a method calculateSalary() for both PermanentEmployee and ContractEmployee. Task: - Implement the Inheritance & Abstract Class structure in Java. (6) - Implement calculateSalary() method using polymorphism. (2) - Demonstrate the use of method overriding. (2) <i>(Full correct implementation: 10 marks, Partial correctness: 6-8 marks, Minor issues: 3-4 marks)</i>	10	CO2
2.	a)	How does Java achieve multiple inheritance using Interfaces? - Explain with an example how an interface can be used to achieve multiple inheritance in Java. (3) - Discuss advantages and limitations of using interfaces for multiple inheritance. (2)	5	CO1

	b) Scenario: A Library System consists of: • A Library that manages multiple Books. • Each Book has an Author and belongs to a Category. • Each Book is issued to a Member. Task: I. Design a UML Class Diagram covering Inheritance, Abstract Class, Polymorphism, and Association between Library, Book, Author, Category, and Member. (6) II. Clearly define relationships such as One-to-Many, Many-to-Many. (2) III. Justify the use of Abstract Classes and Interfaces if applicable. (2) (Full correct implementation: 10 marks, Partial correctness: 6-8 marks, Minor issues: 3-4 marks)	10	CO3
3.	Scenario: A Ride-Sharing Application needs a system to manage: • Drivers (Driver class) • Passengers (Passenger class) • Rides (Ride class) • Vehicles (Vehicle class) The system should: • Allow a Passenger to book a Ride. • Assign an available Driver to the ride. • Ensure each Ride is associated with a Vehicle and a Driver. • Implement calculateFare() as an abstract method that varies based on ride type (Economy, Premium). Task: a) Design a UML Class Diagram for the system. (7) b) Identify Associations (One-to-Many, Many-to-Many). (3)	10	CO3