Daffodil International University

Department of Computer Science and Engineering

Faculty of Science & Information Technology

Midterm Examination, Summer 2025

Course Code: CSE221, Course Title: Object Oriented Programming

Level:2 Term:2 Batch: ALL

Time: 1.5 Hours **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 **Food-Delivery Platform** connects **Customers** with **Restaurants**. • A **Customer** has custid, name and email • Each Customer places many **Orders**. • An Order has an *orderID*, *orderDate*, and *totalAmount*. • Each Order contains one or more **MenuItems** (name, price). A class-level attribute is required in **Order** to keep the **running total of orders placed** on the platform.		
		a) Business-Case Analysis a) List **three attributes** (with Java data types) for **Customer** and **Order**. b) Identify **two behaviours** (methods) relevant to classes in the case. c) Describe the **relationships** among **Customer**, **Order** and **MenuItem**, including *multiplicity*.	5 6	CO1 CO3
		b) UML Class Diagram Draw a UML diagram comprising **Customer**, **Order** and **MenuItem** showing: • attributes with visibility and data types • methods you identified in part a) • association links with multiplicities • the **static attribute** that tracks total orders	0	CO4
		c) Java Implementation Implement the UML model: (code)		

		1. Create the three classes with constructors and the members from UML model. 2. Declare and update the static attribute in **Order**. 3. In a `main()` method, instantiate **one Customer** who places **two Orders**, each containing at least one **MenuItem**. 4. Print a summary showing customer name, order IDs, item names and **total orders so far**. **d) Reasoning on Design Choices** Why is the *totalOrders* counter best declared **static**? Write 3 points which may further enhance your model.	3	CO2
2.		**Problem Solving** **Scenario:** A **Car-Pooling Service** matches **Drivers** (driverID, rating) with **Rides** (rideID, distance, fare) requested by **Riders** (riderID, name). A Rider can join many Rides; a Ride can include many Riders; a Driver drives many Rides but each Ride has one Driver. Tasks 1. Identify the classes and principal attributes (no methods needed). 2. Sketch a high-level UML class diagram with correct multiplicities (show *Driver–Ride* and *Rider–Ride* separately).	5	CO3

Good Luck