ourdiu.com | Shared by Md Jakaria Hossain

Daffodil International University
Faculty of Science & Information Technology
Department of Software Engineering
Midterm Examination, Spring 2026
Course Code: SE 311; Course Title: Design Pattern
Sections & Teachers: A-D(MHN), E-H(MRN), I(ZH), J(AZ)

Time: 1:30 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)*	**Construct** a Java class StudentData with a method saveStudentData(String name, String id, String dept) that declares **throws** IOException and manually validates the input parameters. The method should check whether any of the provided fields are empty, and if so, it must explicitly **throw** an IOException. In main method of the code, call the saveStudentData method and handle the exception.	[Marks-5]	*CLO-1 Level-3*
	b)	You are designing a small e-commerce checkout system for an online shop. The current Xshop class handles **customer validation, price calculation, discount rules, shipping charges,** and **receipt generation** all inside a single method. The code works, but it is poorly structured: **public class Xshop {** **public static void main(String[] args) {** **XShop s = new XShop();** **System.out.println(s.go("Ayesha", "Laptop Bag", 2, 1200.0, true));}** **public String go(String n, String p, int q, double u, boolean vip) {** **if (n == null || n.isBlank() || q <= 0 || u <= 0) return "Checkout failed.";** **double sub = q * u;** **double disc = vip ? sub * 0.10 : 0.0;** **double ship = (sub - disc) >= 2000 ? 0.0 : 80.0;** **double total = sub - disc + ship;** **return "Receipt: " + n + " bought " + p + " x" + q** **+ " | Sub=" + sub** **+ " | Disc=" + disc** **+ " | Ship=" + ship** **+ " | Total=" + total; }}** **Identify** all the code smells present in the system and refactor it to remove the smells while keeping the functionality unchanged.	[Marks-3]	
	c)	**Demonstrate** key differences between Method Overriding and Method Overloading.	[Marks-2]	
	d)	You have to develop a Ride Sharing app named **Chalao**. Where the users can choose multiple ride options such as: • CNG • BIKE • CAR Each ride option can use the **startRide(String startLocation, String destination)** method. The main application should only request a RideService and must not directly create the specific ride objects. **Required Components:** • Rides (interface) • Cng, Bike, Car (concrete classes) • RideServices (responsible for object creation) • Chalao (main class)	[Marks-5]	

ourdiu.com | Shared by Md Jakaria Hossain

		Construct a proper Java code demonstrating this implementation so that the main class can request a ride service based on the user choice.		
2.	*a)*	You have to develop a University Examination Management System. The system should maintain a central configuration that stores important global settings such as: Exam Schedule and Grading Policy. Every module (ExamOffice, TeacherPanel, StudentPortal) should access the same configuration object. ExamOffice should **showConfig()**, TeacherPanel should **showGradingPolicy()**, StudentPortal should **viewInfo()**. **Required Components:** • ExamConfig (central configuration class) • ExamOffice, TeacherPanel, StudentPortal (object sharing classes) • UniversitySystem (main class) **Build** a Java code with necessary design implementation, which will ensure that only one instance of the configuration class will be created and all the other classes of the system can access that same instance.	[Marks-4]	*CLO- 2 Level-3*
	b)	You have to develop a software-based Hospital Management System where different types of reports can be generated, such as: **BloodReport**, **XrayReport** and **Prescription**. These different types of reports should not be created from scratch every time a new patient needs a report. Instead, the system should generate reports by copying a **BaseReport** template that contains common attributes such as PatientName, ReportType, and ReportDetails. The BaseReport template should provide the operations setPatientName(), updateDetails(), and showReport() by implementing an interface named **MedicalReport**. **Program** a Java code to build a system where each report type can clone the BaseReport template, so that new patient reports can be produced without needing to create a new report object every time.	[Marks-6]	