ourdiu.com | Shared by No Body

Daffodil International University
Faculty of Science & Information Technology
Department of Software Engineering
Final Term Examination, Fall 2025
Course Code: SE 311; Course Title: Design Pattern
Sections & Teachers: RJM(A,B), AH(C,D,M), NML(E,F,G,H), MRN(I,J,K,L), DB(N)
Time: 2:00 Hrs. **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)*	At DIU, students can register for courses. The process involves: Checking course availability (CourseCatalog), Verifying fee payment (FeeSystem), Allocating a seat (SeatAllocation). Students should be able to register by calling only one method, without interacting with the internal steps. Now, **Apply** the proper pattern to execute the classes with the given names: **CourseCatalog, FeeSystem, SeatAllocation, StudentRegistration, Test.** Ensure the student can register successfully through a single method call in the main method.	[Marks -8]	*CLO-2 Level-3*

	b)	Imagine you are designing a notification system for Hogwarts students.
		
		public interface MagicalObserver {
		void magicalAlert(String event); }
		public class HogwartsStudent implements MagicalObserver {
		private String name;
		public HogwartsStudent(String name) {
		this.name = name; }
		public void magicalAlert(String event) {
		System.out.println(name + " receives magical alert: " + event); }}
		public interface MagicalEvent {
		void enrollStudent(MagicalObserver s);
		void expelStudent(MagicalObserver s);
		void announceEvent(); }
		
		i) **Specify** the role of the HogwartsStudent classpublic .
		ii) Which method in the MagicalEvent interface sends updates to all students?
			Marks-2]	

2.	*a)*	At "DIU Smart Result System", the system accepts data only in JSON format. However, some old departmental modules still provide data in XML format. The university wants both systems to work together without modifying the old XML module. a) **Determine** and explain which Structural Design Pattern can solve this problem. b) **Draw** the complete class diagram of the pattern applied in this scenario. c) **Implement** Java code for this scenario using the following classes: • JSONData • OldXMLModule • XMLtoJSONAdapter - • DIUSmartResultSystem • Test Ensure that the client can retrieve results in JSON format from the old XML module through a single method call	[Marks - 1+4+7]	*CLO-3 Level-3*

ourdiu.com | Shared by No Body

	b)	You are designing a live sports score application. Multiple matches happen simultaneously. Fans receive notifications whenever a score changes. The class MatchManager handles updating scores, managing fan subscriptions, and sending notifications. All fans must implement all notification methods (Email, SMS, Push), even if they only want one. Current Implementation Problems: • MatchManager mixes different responsibilities. • Adding a new notification type requires modifying MatchManager. • Fans are forced to implement unnecessary notification methods. **Specify** the name of three SOLID principle violations in this design with explanation.	**Marks-3]**	
3.	*a)*	A Traffic Light can be in three states: Red, Green, and Yellow. Depending on the current state, calling **nextLight()** changes it to the next state in this order: Red → Green → Yellow → Red → ..... • **Produce** the State interface and concrete states: RedState, GreenState, and YellowState. • **Implement** the TrafficLight class that maintains the current state. • **Demonstrate** the traffic light changing states by calling nextLight() 5 times in a main method. **Expected Output:** Traffic Light is RED Traffic Light is GREEN Traffic Light is YELLOW Traffic Light is RED Traffic Light is GREEN	**[Marks -11]**	

	b)	**public interface Msg {**
		String show(); }
		public class Base implements Msg {
		public String show() {
		return "A"; } }
		public abstract class Wrap implements Msg {
		protected Msg m;
		public Wrap(Msg m) {
		this.m = m; }
		public String show() {
		return m.show(); } }
		
		public class X extends Wrap {
		public X(Msg m) {
		super(m); }
		public String show() {
		return "Exam"; } }
		public class Y extends Wrap {
		public Y(Msg m) {
		super(m); }
		public String show() {
		return super.show(); } }
		public class Test {
		public static void main(String[] args) {
		Msg m = new X(new Y(new X(new Base())));
		System.out.println(m.show()); } }
		
		Solve the code and
		i) Identify which design pattern is implemented in the given code?
		ii) Write the final output.
			[Marks 4]	*CLO-3 Level -6*