



Daffodil International University
Faculty of Science & Information Technology
Final Examination, Fall 2022

Course Code: CSE 221/214, Course Title: Object Oriented Programming

Level: 2 Term: 1+2 Batch: 59, 60, OLD-SYL

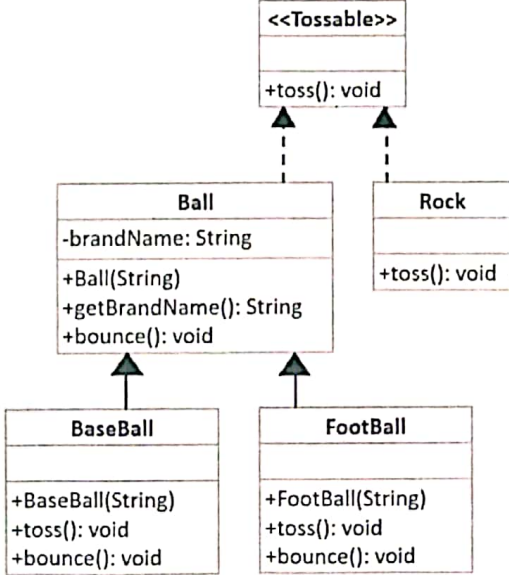
Time: 2:00 Hrs

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)	<p>What is the main difference between checked and unchecked exception? Find out the output of the following java program?</p> <pre> public class X { public static void main(String[] args) { try { badMethod(); System.out.print("A"); } catch (RuntimeException ex) { System.out.print("B"); } catch (Exception ex1) { System.out.print("C"); } } finally { System.out.print("D"); } System.out.print("E"); } public static void badMethod() { throw new RuntimeException(); } </pre>	[2.5]	CO1
	b)	Compare between Aggregation and composition with proper example.	[2.5]	
2.	a)	<p>"Multiple Inheritance is not supported through a class in Java, but it can be possible through the interface" – Explain your answer with example. Find out the output of the following java program.</p> <pre> interface GFG{ void learnCoding(); void learnProgramming(); void contribute(); } abstract class Student implements GFG { public void learnCoding(){ System.out.println("Let's make coding a habit with GFG"); } public void learnProgramming(){ System.out.println("Let's Learn Java"); } } class GEEK extends Student { public void contribute() { System.out.println("Let's help others to Learn java"); } public void learnCoding() { System.out.println("Let's make coding a habit with Java"); } } public class NewClass { public static void main(String[] args){ GEEK gfgStudent = new GEEK(); gfgStudent.learnCoding(); gfgStudent.learnProgramming(); gfgStudent.contribute(); } } </pre>	[5]	CO2
	b)	<p>Is there any Exception exists in the following program? If yes, explain that Exceptions that were causing problems and what approach have you taken to solve this problem.</p> <pre> public class Test { public static void main(String[] args) { int a = 20, b = 30, c = 10; int x = (a * b) / (a - b + c); System.out.println("Result: " + x); } } </pre>	[5]	

3.	<p>a) Let us consider a Company class with properties of name, reg_no and type and the function elements are getter and setter methods for each data field. It has a parameterized constructor to initialize the name, reg_no and type at the time of object creation. Company class also has a display method to show the company details.</p> <p>Company has a large number of employees of two different categories. Employee class has emp_id, emp_name, designation as data fields and getter and setter methods for these data fields. It defines a default constructor. Employee class also has a payment method but has no implementation.</p> <p>SalariedEmployee is one type of Employee with the basic properties of Employee with additional properties of salaryScale. It has a method to set salaryScale and it overrides the payment method from the Employee class which returns the salary of the employee.</p> <p>HourlyEmployee is another category or Employee which inherits the Employee class and has the data member of hours_worked and hourly_payment. It has getter and setter methods for data fields and it implements the payment method for salary payment returning hours_worked*hourly_payment.</p> <p>Now, develop a UML diagram for the above-mentioned company's class hierarchy maintaining all mentioned relationships.</p> <p>b) Assume that you have to create an interface called 'Machine' to represent any kind of machine. 'Machine' has two methods: start() and stop(). Now create two implementations of the 'Machine': a Car, and a WaterPump. Show that the same piece of code works fine for both machines. Also draw the UML of the above scenario.</p>	[6]	CO3
4.	<p>a) The abstract Fruits class has three subclasses named Apple, Banana and Jackfruit. It also has an abstract method named "display". Now develop a java program that demonstrates how to establish this class hierarchy. Declare an instance variables of type String that indicates the color of a fruit. Create objects of each class and display the name of the fruit and its color.</p> <p>b) Develop a Java Code to implement the following UML Diagram. You do not need to fill in the method bodies for the toss or bounce methods.</p>  <pre> classDiagram class Tossable { <<Tossable>> +toss() void } class Ball { -brandName: String +Ball(String) +getBrandName() String +bounce() void } class Rock { +toss() void } class BaseBall { +BaseBall(String) +toss() void +bounce() void } class FootBall { +FootBall(String) +toss() void +bounce() void } Tossable < .. Ball Tossable < .. Rock Ball < -- BaseBall Ball < -- FootBall </pre>	[6] [9]	CO4