



Daffodil International University
Faculty of Science & Information Technology
Department of Software Engineering
Final Examination, Spring 2025
Course Code: SE216; Course Title: Object Oriented Programming
Sections: A-J & Teachers: RJM, MAAA, NAN, PC, KMH

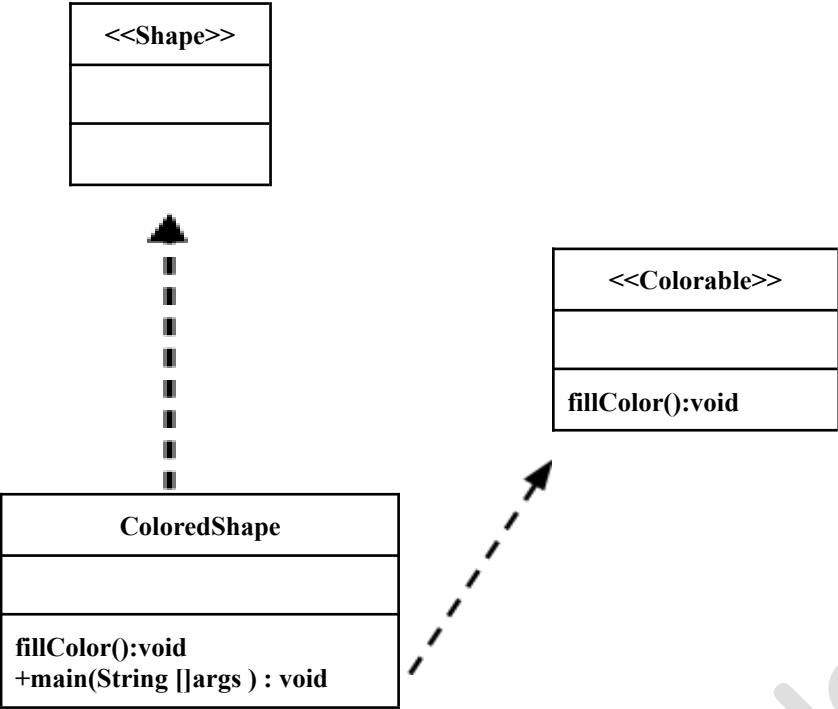
Time: 2:00 Hrs

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)	Solve the following scenario using appropriate UML diagrams, illustrating the association relationships. Alex is a Learner. Alex has a fullName, studentID of type String, int. Learner has a studentNumber of type int and performs an action showDetails() of type void. Alex must have a ReportCard, and without this, Alex cannot graduate. ReportCard has a recordID of type int.	2	CLO-3 Level-3				
	b)	You are organizing a tech fest with participants from two different departments: SWE and CSE., Your task is to manage and manipulate the list of participants using ArrayLists in Java. Customize a class called TechFest that performs the following: - Create two ArrayLists: One for SWE participants and one for CSE participants. Add the following participants to each list: - SWE participants: Alice, Bob, Charlie - CSE participants: David, Eva, Frank - Combine both lists into a single FinalList. - Replace the first occurrence of the participant "Alice" in the FinalList with "Grace". - Display the first and last participants in the FinalList. - Find the last occurrence of the participant "David" in the FinalList and display the index. If "David" is not found, display the message: "David is not in the list." - Remove all CSE participants from the FinalList. - Display the size of the FinalList. - Clear the FinalList. - Check if the FinalList is empty and display the message: "The list is empty!" if it is empty, otherwise display: "The list is not empty!"	7					
	c)	<table><tr><th>Code</th><th>Output</th></tr><tr><td>Class1: public class StudentTest{ public static void main(String [] args){ Student s1 = new Student(); System.out.println(s1.getName()); Student s2 = new Student("Elizabeth"); System.out.println(s2.getName()); Student s3 = new Student("Teylor"); System.out.println(s3.getName()); System.out.println(Student.numberOfStudents); } }</td><td>No Name Elizabeth Teylor 3</td></tr><tr><td>Class2: public class StudentSetNameTest { public static void main(String[] args) { Student s = new Student(); System.out.println(s.getName()); s.setName("Anabia"); System.out.println(s.getName()); } }</td><td>No Name Anabia</td></tr></table> You are given the following two Java programs that use a Student class. Construct the Student class so that both programs produce the expected outputs shown in the right column.	Code		Output	Class1: public class StudentTest{ public static void main(String [] args){ Student s1 = new Student(); System.out.println(s1.getName()); Student s2 = new Student("Elizabeth"); System.out.println(s2.getName()); Student s3 = new Student("Teylor"); System.out.println(s3.getName()); System.out.println(Student.numberOfStudents); } }	No Name Elizabeth Teylor 3	Class2: public class StudentSetNameTest { public static void main(String[] args) { Student s = new Student(); System.out.println(s.getName()); s.setName("Anabia"); System.out.println(s.getName()); } }
Code	Output							
Class1: public class StudentTest{ public static void main(String [] args){ Student s1 = new Student(); System.out.println(s1.getName()); Student s2 = new Student("Elizabeth"); System.out.println(s2.getName()); Student s3 = new Student("Teylor"); System.out.println(s3.getName()); System.out.println(Student.numberOfStudents); } }	No Name Elizabeth Teylor 3							
Class2: public class StudentSetNameTest { public static void main(String[] args) { Student s = new Student(); System.out.println(s.getName()); s.setName("Anabia"); System.out.println(s.getName()); } }	No Name Anabia							

	c)	<pre>classDiagram class Shape { <<Shape>> } class ColoredShape { fillColor():void +main(String []args) : void } class Colorable { <<Colorable>> fillColor():void } Shape < .. ColoredShape Colorable < .. ColoredShape</pre> Translate the above class diagram into a proper JAVA code. Expected Output: Fill the shape with color.	3	
2.	a)	Create an abstract class Instrument with two abstract methods: play() and adjust(). The class also has one concrete method: compose(). Then, create subclasses Guitar, Keyboard, and Violin that implement the necessary methods from the Instrument class. Finally, in a TesterClass Create instance(object) of every class and invoke(call) every method on each object. The output should display the corresponding messages for each method call, as follows: Guitar Class: In the playing method of Guitar Adjusting the strings of Guitar Composing music for Guitar Keyboard Class: In the playing method of Keyboard Adjusting the keys of Keyboard Composing music for Keyboard Violin Class: In the playing method of Violin Adjusting the strings of Violin Composing music for Violin	12	CLO-4 Level-6
	b)	Develop a Java program that asks the user to input two integers and divides the first number by the second. You must use nested try-catch blocks to handle the following cases: - If the user enters a non-integer input, catch the <code>InputMismatchException</code> and display: "Invalid input! Please enter integers only." - If the input is correct, print "Input is correct." - If the user enters zero as the divisor, throw and handle a custom exception named <code>DivideByZeroException</code> that displays: "Custom Exception: Division by zero is not allowed." - If both integers are valid and the divisor is non-zero, print the result of the division. - Finally, the program should always print: "Program terminated." regardless of any exception. All exceptions must be handled using a single pair of nested try-catch blocks — one outer try for input mismatch and one inner try for division and custom exception.	8	