REG.NO.:

SLOT: D2+TD2

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
FALL SEMESTER 2025-2026

Programme Name & Branch : **M. Tech. Integrated (CSE Core and Data Science)**
Course Code and Course Name : **CSI2008 & Programming in Java**
Faculty Name(s) : **Dr. V. SIVAKUMAR, Dr. LOKESH KUMAR R, Dr. BHUVANESWARI M S**
Class Number(s) : **VL2025260102506, VL2025260102498, VL2025260102501**
Date of Examination : **20-08-2025**
Exam Duration : **90 minutes** **Maximum Marks: 50**

CO1. Analyze the programs involving the fundamental program constructs.
CO2. Choose the appropriate OOP technique for solving the real world problem.
CO3. Demonstrate exception handling and use of threads in Java.

Q. No	Question	M	CO	BL
1.	**A)** What is the difference between **local and global (instance)** variables in Java? Give an example program to illustrate your answer. (5Marks)	10	CO1	BL3

Feature	Local Variable	Instance (Global) Variable
Where declared	Inside a method, constructor, or block	Inside a class, **but outside** any method or constructor
Scope	Accessible **only within** the method/block it is declared in	Accessible by **all methods** of the class (depending on access modifier)
Lifetime	Exists only while the method/block is running	Exists for the lifetime of the object
Default value	**No default value** — must be initialized before use	Automatically initialized with default values (e.g., `0`, `null`, `false`)
Storage	Stored in the **stack**	Stored in the **heap** (part of the object)

```
class VariableExample {
   // Instance (Global) variable
   int instanceVar = 10;

   void methodOne() {
      // Local variable
      int localVar = 5;

      System.out.println("Instance variable: " + instanceVar);
      System.out.println("Local variable: " + localVar);
   }
```

	void methodTwo() { // We can access instanceVar here as well System.out.println("Instance variable in methodTwo: " + instanceVar); // localVar is not accessible here — would cause a compile-time error // System.out.println(localVar); // Error } public static void main(String[] args) { VariableExample obj = new VariableExample(); obj.methodOne(); obj.methodTwo(); } } **B)** The following program has several errors. Modify it so that it will compile and run without errors. // Filename: Temperature.java (5Marks) PUBLIC CLASS temperature { PUBLIC void main(string args) { double fahrenheit = 62.5; */ Convert /* double celsius = f2c(fahrenheit); System.out.println(fahrenheit + 'F = ' + celsius + 'C'); } double f2c(float fahr) { RETURN (fahr - 32) * 5 / 9; } } **Answer:** // Filename: Temperature.java public class Temperature { public static void main(String[] args) { double fahrenheit = 62.5; /* Convert */ double celsius = f2c(fahrenheit); System.out.println(fahrenheit + "F = " + celsius + 'C'); } static double f2c(double fahr) { return(fahr - 32)*5/9; } } **Compile by: javac Temperature.java** **Run by: java Temperature** 62.5F = 16.944444444444443			
2.	Create a Java program using a 2D array to represent the **seating arrangement of a movie theater**. The theater has **5 rows and 6 columns**. Initialize all seats as "Available" (marked with 0). Book specific seats (change to 1). Print the seat matrix.	10	CO1	BL3

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
FALL SEMESTER 2025-2026

Answer:

```
public class MovieTheater {
   public static void main(String[] args) {
      // Create 2D array for 5 rows and 6 columns
      int[][] seats = new int[5][6];

      // Initialize all seats as 0 (Available)
      for (int i = 0; i < seats.length; i++) {
         for (int j = 0; j < seats[i].length; j++) {
            seats[i][j] = 0;
         }
      }
      // Book specific seats (set to 1)
      seats[0][2] = 1; // Row 1, Column 3
      seats[2][4] = 1; // Row 3, Column 5
      seats[4][0] = 1; // Row 5, Column 1
      seats[3][5] = 1; // Row 4, Column 6

      // Print the seating arrangement
      System.out.println("Movie Theater Seating Arrangement:");
      for (int i = 0; i < seats.length; i++) {
         for (int j = 0; j < seats[i].length; j++) {
            System.out.print(seats[i][j] + " ");
         }
         System.out.println(); // Move to next row
      }
   }
}
```

3.	Design a **Payment Processing System** in Java where: A. The system allows processing different types of payments (like **Debit Card**, and **UPI**) using method overloading based on different parameter combinations. Method name should be processPayment (5Marks) B. There are different payment providers like **BHIM** and **GooglePay** that process payments differently. Each of these should override a method authenticate() using method overriding. (5Marks) **Answer:**	10	CO2	BL3

```
// Base class
class Payment {
   // Method Overriding: to be overridden by specific providers
   String authenticate() {
      return "Authenticating using generic provider";
   }

   // Method Overloading: processPayment with different parameters
   // UPI Payment
   void processPayment(String upiId) {
```

```
        System.out.println("Processing UPI payment for ID: " + upiId);
    }

    // Debit Card Payment with PIN
    void processPayment(String cardNumber, String cvv, String pin) {
        System.out.println("Processing Debit Card payment with card: " +
cardNumber + " and PIN.");
    }
}
// Subclass 1
class BHIM extends Payment {
    @Override
    String authenticate() {
        return "Authenticating with BHIM API";
    }
}

// Subclass 2
class GooglePay extends Payment {
    @Override
    String authenticate() {
        return "Authenticating with GooglePay servers";
    }
}

// Main class
public class PaymentSystem {
    public static void main(String[] args) {
        // Method Overloading
        Payment payment = new Payment();
        payment.processPayment("user@upi");
        payment.processPayment("1234-5678-9012-3456", "123");
        payment.processPayment("1234-5678-9012-3456", "123", "0000");

        // Method Overriding
        Payment provider;

        provider = new BHIM();
        System.out.println(provider.authenticate());
                    provider = new GooglePay();
        System.out.println(provider.authenticate());
    }
}
```

4.	Create an abstract class named **VITEmployee** that contains variables, one non-abstract method and abstract method named **displayDetail** with parameters Designation and SchoolName, prints the employee details. Provide three	10	CO2	BL6

classes named Faculty, Instructor and Attender such that each one of the classes use the class **VITEmployee**.

Answer:

```
// Abstract class
public class Main {
    public static void main(String[] args) {
        Faculty f = new Faculty("Dr. Raj", 101);
        f.displayDetail("Professor", "School of Computer Science");

        Instructor i = new Instructor("Ms. Priya", 102);
        i.displayDetail("Lab Instructor", "School of Electronics");

        Attender a = new Attender("Mr. Kumar", 103);
        a.displayDetail("Attender", "School of Mechanical Engineering");
    }
}

abstract class VITEmployee {
    String empName;
    int empId;

    // Constructor
    VITEmployee(String empName, int empId) {
        this.empName = empName;
        this.empId = empId;
    }

    // Non-abstract method
    void printBasicInfo() {
        System.out.println("Employee Name: " + empName);
        System.out.println("Employee ID: " + empId);
    }

    // Abstract method
    abstract void displayDetail(String designation, String schoolName);
}

// Faculty class
class Faculty extends VITEmployee {
    Faculty(String name, int id) {
        super(name, id);
    }

    @Override
```

```
    void displayDetail(String designation, String schoolName) {
        printBasicInfo();
        System.out.println("Designation: " + designation);
        System.out.println("School Name: " + schoolName);
        System.out.println("------------------------");
    }
}

// Instructor class
class Instructor extends VITEmployee {
    Instructor(String name, int id) {
        super(name, id);
    }

    @Override
    void displayDetail(String designation, String schoolName) {
        printBasicInfo();
        System.out.println("Designation: " + designation);
        System.out.println("School Name: " + schoolName);
        System.out.println("------------------------");
    }
}

// Attender class
class Attender extends VITEmployee {
    Attender(String name, int id) {
        super(name, id);
    }

    @Override
    void displayDetail(String designation, String schoolName) {
        printBasicInfo();
        System.out.println("Designation: " + designation);
        System.out.println("School Name: " + schoolName);
        System.out.println("------------------------");
    }
}
```

Output:

```
Employee Name: Dr. Raj
Employee ID: 101
Designation: Professor
School Name: School of Computer Science
------------------------
Employee Name: Ms. Priya
Employee ID: 102
```

	Designation: Lab Instructor School Name: School of Electronics ---------- Employee Name: Mr. Kumar Employee ID: 103 Designation: Attender School Name: School of Mechanical Engineering ----------			
5.	Write a java program to **create custom exception for Numeric Value Exception** if the user enters the value. **TestCase-1:** Input number :: 45 Number 45 is Positive MyException **TestCase-2:** Input number :: -35 Number -35 is Negative MyException **Answer:** import java.io.BufferedReader; import java.io.IOException; import java.io.InputStreamReader; class MyException extends Exception { public MyException(String str) { System.out.println(str); } } public class SignException { public static void main(String[] args)throws IOException { BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); System.out.print("Input number :: "); try { int num = Integer.parseInt(br.readLine()); if(num < 0) throw new MyException(" Number "+num+" is Negative"); else throw new MyException(" Number "+num+" is Positive"); } catch (MyException m) { System.out.println(m);	10	CO3	BL6

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST - I
FALL SEMESTER 2025-2026

	} } }			