Scheme of valuation

ICT 1271 Introduction to Object Oriented Programming.

Date of exam 04/03/2025 10.45 to 12.15pm

1	What is the output of following code snippet? int Arr[][]=new int[3][]; Arr[0]=new int[2]; Arr[1]=new int[4]; Arr[2]=new int[3]; System.out.println(Arr[1].length); 1. 3 2. 4 3. 2 4. Compilation error
2	What happens when you compile the below code snippet? String s1="hello"; String s2=s2.substring(0,5); s2=s2.intern(); if (s1==s2) System.out.println("hello"); 1. No output. 2. Compilation error 3. Prints hello 4. Runtime error
3	Which of the following is NOT a Java primitive data type? a) byte b) char c) string d) float
4	Which of the following best describes Java's Just-In-Time (JIT) compiler? a) It converts Java source code directly to machine code. b) It compiles entire Java programs before execution. c) It compiles bytecode into native machine code at runtime. d) It does not affect Java's execution speed.

5	What will be the output of the following code? int a=0,b=8; if((a!=0)&& ((++b)/a >2)) System.out.println(a); else System.out.println(b); 1. 8 2. 9 3. Compilation error 4. None of the options
6	What will be the output of the following code? int a = 5, b = 10; if (a++ > 5) { if (b-- > 10) { System.out.println("X"); } else { System.out.println("Y"); } } else { if (++b > 10) { System.out.println("Z"); } else { System.out.println("W"); } } System.out.println("a: " + a + ", b: " + b); a) X, a: 6, b: 9 b) Y, a: 6, b: 9 c) Z, a: 6, b: 10 d) W, a: 6, b: 11
7	1) Choose which of the following code snippets will NOT compile? A. String s1 = "Object"; String s2 = s1.concat(" Oriented"); B. String s1 = "Object"; StringBuilder sb = new StringBuilder(s1); sb.append(" Oriented"); C. String s1 = new String("Object"); String s2 = s1.substring(0, 2); D. **String s1 = "Object";** **s1.setCharAt(0, 'P');**
8	public class Test { public static void main(String[] args) {

	String s1 = "abc"; String s2 = new String("abc"); String s3 = "abc"; System.out.println(s1 == s2); System.out.println(s1 == s3); } } D) true false **D) false true** D) false false D) true true
9	Which of the following statements about Java constructors is FALSE? a) A constructor can be private. b) A constructor cannot be overloaded. c) A constructor is called automatically when an object is created. d) If no constructor is defined, Java provides a default constructor.
10	What will be the output of the following Java program? class Demo { int x = 5; void changeValue(Demo d) { d.x += 10; } public static void main(String args[]) { Demo obj1 = new Demo(); Demo obj2 = obj1; obj2.changeValue(obj1); System.out.println(obj1.x); } } a) 5 b) 10 c) 15 d) Compilation error
11	Write a Java program to define a class called Matrix that has a 2D array of integers and the following methods. a) A parameterized constructor to receive two integers (m and n) as parameters and allocate memory basic for the 2D array of size m*n. b) Another parameterized constructor to receive an integer 2D array and allocate memory to the instance variable and initialize the array with the contents of the array received. c) Method to display the Matrix using for each loop d) Method find() to return a 1D array containing all prime numbers in the matrix.

Test class Matrix with class MatrixDemo having main method to test all the methods of the class appropriately.

Scheme:

Part a-1M part b-1M part c-1M Part d-1M

Rest -1M

```
class Matrix {
   private int[][] matrix;

   // Constructor to allocate memory for m x n matrix
   public Matrix(int m, int n) {
      matrix = new int[m][n];
   }

   // Constructor to initialize matrix with given 2D array
   public Matrix(int[][] arr) {
      int rows = arr.length;
      int cols = arr[0].length;
      matrix = new int[rows][cols];
      for (int i = 0; i < rows; i++) {
         System.arraycopy(arr[i], 0, matrix[i], 0, cols);
      }
   }

   // Method to display the matrix
   public void display() {
      for (int[] row : matrix) {
         for (int num : row) {
            System.out.print(num + " ");
         }
         System.out.println();
      }
   }

   // Method to find all prime numbers in the matrix without using ArrayList
   public int[] find() {
            int[] primes = new int[arr.length*arr[0].length];

      int index = 0;
      for (int[] row : matrix) {
         for (int num : row) {
            if (isPrime(num)) {
               primes[index++] = num;
            }
         }
      }
    int a[]=new int [index-1];
    for(int i=0;i<a,length;i++){a[i]=primes[i];}
    primes=a;
      return primes;
```

```
    }

    // Helper method to check if a number is prime
    private boolean isPrime(int num) {
        if (num < 2) return false;
        for (int i = 2; i * i <= num; i++) {
            if (num % i == 0) return false;
        }
        return true;
    }
}

public class MatrixDemo {
    public static void main(String[] args) {
        int[][] data = {
            {2, 3, 4},
            {5, 6, 7},
            {8, 9, 10}
        };

        Matrix matrix = new Matrix(data);
        System.out.println("Matrix:");
        matrix.display();

        int[] primes = matrix.find();
        System.out.println("Prime numbers in the matrix:");
        for (int prime : primes) {
            System.out.print(prime + " ");
        }
    }
}
```

12	Define a class called BillPayment with paymenttype,amount,details.For this given BillPayment class, demonstrate constructor overloading for three types of payment. That is for cash on delivery, card, and UPI payments. Instantiate the BillPayment class using each overloaded constructor. Call the processPayment() method on each instance to display the corresponding payment details. Scheme: **2 Mark:** Correct implementation of overloaded constructors for COD, card, and UPI payments with "this" keyword. **1 Mark:** Proper initialization of instance variables and setting of payment details. **2 Mark:** Correct implementation of the processPayment() method and instantiation in the main() method to produce the expected output. VM: 2.5 If function overriding done instead of overloading (rest all correct). 2 if no overloading is done (rest all correct). 2 if constants are passed to constructor (rest all correct). 4.5 if detail variable is handled differently. 3. if overloading of other function done (rest correct). Program: class BillPayment { // Instance variables to hold payment details String paymentType; double amount; String details; // Default Constructor for Cash on Delivery (COD)

```
BillPayment() {
    this.paymentType = "COD";
    this.amount = 0.0;
    this.details = "Cash on Delivery";
}

// Constructor for Card Payments (Credit/Debit)
BillPayment(String cardType, double amount) {
    this.paymentType = cardType + " Card";
    this.amount = amount;
    this.details = cardType + " Card Payment";
}

// Constructor for UPI Payments
BillPayment(String upiId, double amount, boolean isUpi) {
    if(isUpi) {
        this.paymentType = "UPI";
        this.amount = amount;
        this.details = "UPI Payment using ID: " + upiId;
    }
}

// Method to process and display payment details
void processPayment() {
    if(paymentType.equals("COD")) {
```

```
            System.out.println("Payment will be collected on delivery (COD).");
        } else {
            System.out.println("Processing " + details + " for amount ₹" + amount);
        }
    }
}

class MainPayment{
    public static void main(String[] args) {
        // COD Payment using the default constructor
        BillPayment bp1 = new BillPayment();
        bp1.processPayment();

        // Credit Card Payment using constructor overloading
        BillPayment bp2 = new BillPayment("Credit", 2500.50);
        bp2.processPayment();

        // Debit Card Payment using constructor overloading
        BillPayment bp3 = new BillPayment("Debit", 1500.75);
        bp3.processPayment();

        // UPI Payment using constructor overloading
        BillPayment bp4 = new BillPayment("user@upi", 500.00, true);
        bp4.processPayment();
    }
}
```

14	Design a Java program for an automobile manufacturing plant that tracks production. Create a class Car with instance variables for details such as **model** (a String) and **productionNumber** (an int). Use a static variable to count the total number of cars produced. Implement a static method **displayProductionSummary()** that prints the total production count. In the main class, simulate the production of multiple cars by creating Car objects and then display the production summary using the static method. **Scheme:** **2 Mark:** Correct class definition with instance variables (model and productionNumber) and a constructor that initializes these variables. **1 mark:** Correct use of a static variable (totalCarsProduced) that is incremented within the constructor. **1 Mark:** Correct implementation and invocation of the static method displayProductionSummary() in the main() method, demonstrating the overall production count. **Program:** class Car { // Instance variables to hold car details String model; int productionNumber; // Static variable to count total cars produced static int totalCarsProduced = 0; // Constructor to initialize car details and increment the static counter

```
    Car(String model, int productionNumber) {
        this.model = model;
        this.productionNumber = productionNumber;
        totalCarsProduced++;  // Increment the count for each new car produced
    }

    // Static method to display the total production summary
    static void displayProductionSummary() {
        System.out.println("Total Cars Produced: " + totalCarsProduced);
    }

    // Optional: Method to display individual car details
    void displayCarDetails() {
        System.out.println("Model: " + model + ", Production Number: " +
productionNumber);
    }
}

public class ManufacturingPlant {
    public static void main(String[] args) {
        // Simulate production of multiple cars
        Car car1 = new Car("Model S", 101);
        Car car2 = new Car("Model 3", 102);
        Car car3 = new Car("Model X", 103);

        // Optionally display details of each produced car
```

```
        car1.displayCarDetails();

        car2.displayCarDetails();

        car3.displayCarDetails();


        // Display the production summary using the static method

        Car.displayProductionSummary();

    }

}
```

13 Create a class Smartphone that contains an inner class Processor. Implement the following:
1.The Smartphone class should have an instance variable brand and a constructor to initialize it.
2.The inner class Processor should have instance variables cores (int) and clockSpeed (double, in GHz) with a constructor to initialize them.
3.The Processor class should have a method displayProcessorDetails() that prints the smartphone brand, number of cores, and clock speed.
Implement the necessary logic in the main() method to create a Smartphone object and associate it with a Processor object. Then, display the processor details.

Scheme:
Class & Inner Class Definition - 2 Mark
Implementation of displayProcessorDetails() Method - 2 Mark
Object Creation & Execution in main() -1 Mark

Program:

```
class Smartphone {
    private String brand;

    // Constructor for Smartphone
    public Smartphone(String brand) {
        this.brand = brand;
    }

    // Inner class Processor
    class Processor {
        private int cores;
        private double clockSpeed; // in GHz

        // Constructor for Processor
```

	`` public Processor(int cores, double clockSpeed) { this.cores = cores; this.clockSpeed = clockSpeed; } // Method to display processor details public void displayProcessorDetails() { System.out.println("Smartphone Brand: " + brand); System.out.println("Processor Cores: " + cores); System.out.println("Clock Speed: " + clockSpeed + " GHz"); } } public static void main(String[] args) { // Implement the necessary logic to create and display details Smartphone phone = new Smartphone("Apple"); Smartphone.Processor processor = phone.new Processor(6, 3.1); processor.displayProcessorDetails(); } } ```
15	A retail store needs a program to calculate the final bill for a customer purchasing multiple products. The program should: Store the prices of 5 products in an array of suitable type. Ask the customer to enter a discount percentage and a tax percentage. Find the total price of all products before applying any discount or tax. Apply the discount, then add tax, and compute the final amount the customer has to pay. Make sure that: The total price of products can be stored in a variable that allows decimal values.[type casting] The discount and tax percentages, given by the customer, are rounded to whole numbers before use. Display: The total price before applying any discount or tax. The amount after applying the discount. The final amount the customer must pay after adding tax. Scheme: Declaration & Initialization of Array – 1 Mark User Input Handling using Scanner - 1 Mark Type Conversion & Computation - 1 Mark Output Computation & Display -1 Mark

	Program: import java.util.Scanner; public class RetailBillingSystem { public static void main(String[] args) { // Step 1: Declare and initialize an array for product prices int[] productPrices = {100, 200, 150, 300, 250}; // Prices of 5 products int totalPrice = 0; // Step 2: Calculate total price of products for (int price : productPrices) { totalPrice += price; } // Step 3: Read discount and tax percentage from user Scanner scanner = new Scanner(System.in); System.out.print("Enter discount percentage: "); float discountPercent = scanner.nextFloat(); System.out.print("Enter tax percentage: "); float taxPercent = scanner.nextFloat(); scanner.close(); // Step 4: Type conversions double widenedTotalPrice = totalPrice; // Implicit widening (int → double) int discount = (int) discountPercent; // Explicit narrowing (float → int) int tax = (int) taxPercent; // Explicit narrowing (float → int) // Step 5: Compute final price double discountAmount = (widenedTotalPrice * discount) / 100; double priceAfterDiscount = widenedTotalPrice - discountAmount; double taxAmount = (priceAfterDiscount * tax) / 100; double finalAmountPayable = priceAfterDiscount + taxAmount; // Step 6: Display results System.out.println("Total Price before Discount & Tax: $" + widenedTotalPrice); System.out.println("Total Price after Discount: $" + priceAfterDiscount); System.out.println("Final Payable Amount after Tax: $" + finalAmountPayable); } }
16	List the conditions under which automatic type conversion takes place in Java. What are the type promotion rules in Java for expressions? Scheme:

	Two conditions ---0.5M each Type promotion rules --atleast 2 important rules—0.5M each Answer: Conditions for Automatic Type Conversion in Java: Compatible Data Types, Smaller to Larger Data Type Type Promotion Rules: Byte, Short, and Char are Promoted to Int, Wider Data Type Takes Precedence