Final Exam preparation

IT23051

Ans to the Qn No: 01

```
import java.io.*;
import java.util.*;
public class NumberProcessor{
    public static void main(String[] args) {
    String inputFile = "input.txt";
    String outputFile = "output.txt";

    try{
    Scanner scanner = new Scanner(new File(inputFile
    scanner.useDelimiter(",");

List<Integer> numbers = new ArrayList<>();
while (scanner.hasNext()) {
    String numStr = scanner.next().trim();
    if (!numStr.isEmpty()) {
        numbers.add(Integer.parseInt(numStr));
    }
}
scanner.close();
```

```
PrintWriter writer = new PrintWriter(new File(outputfile));

for (int num : numbers) {
    int sum = num * (num + 1) / 2;
    writer.print(num + ", " + sum + " ");
}

writer.close();
System.out.println("Processing completed, check output.txt
                    for results.");

} catch (FileNotFoundException e) {
    System.out.println("Error: Input file not found.");
} catch (NumberFormatException e) {
    System.out.println("Error: Invalid number
                        format in input file.");
}
}
}
```

Answer No: 02

Differences between static and final Fields and Methods

Static

1. Belongs to the class, not instances
2. Shared across all instances
3. Can be accessed without an object
4. Resolved at compile-time
5. common for constants and utility methods

final

1. Cann't be changed after initialization
2. Value cannt be reassigned once set
3. Cann't be overridden in subclasses
4. Enforced al compile-time
5. Used for constants and preventing method overriding

when accessing a static field/methods then,

1. The compiler does not prevent you for accessing static field/methods via an object

2. However, it's not recommended because static member belong to the class, not instances

3. The Java compiler issues a warning (not an error), suggesting the use of the class name

Answer NO: 3

```
import.java.util.Scanner;

public class FactorionNumbers {
    private static final int[] factorials = new int [10];

  static {
   factorials [0] = 1;
   for (int i=1; i<10; i++) {
      factorials [i] = factorials [i-1] * i;
   }
  }
  private static int sumOfFactorialOfDigits(int num){
  int sum =0, temp = num;
  while (temp >0){
      sum += factorials [temp%10];
      temp /= 10;
   }return sum;
  }
  private static void findFactorions (int lower, int upper){
     boolean found = false;
   for (int = lower; i<= upper; i++){
      if (i== sumOfFactorialOfDigits(i)){
        System.out.print(i+" ");
        found = true;
  }?
```

```
        if (!found){
            System.out.println("No factorion numbers
                found in this range.");
        }
    }
    Public static void main (String [] args){
        Scanner scanner = new Scanner (System.in);
System.out.print ("Enter the lower bound of the range:");
        int lower = scanner.nextInt();
System.out.print ("Enter the upper bound of the range:");
        int upper = scanner.nextInt();
System.out.println ("Factorion numbers in the range:");
        findFactorions (lower, upper);
        scanner.close();
    }
}
```

Answer no : 04

Difference among class, local, Instance variables in Java

Class variable:

1. Shared accross all instances (belongs to the class)
2. static keyword inside a class
3. stored in the method area
4. Can have public, private etc

Instance variable:

1. Unique for each object (belongs to the class)
2. Declared inside a class, but outside methods
3. stored in the heap (inside object)
4. Can have public, private, etc.

Local variable:

1. Exists only within the method/block where declare
2. Declared inside a method, constructor, or block
3. stored in the stack
4. Cann't have access modifiers

Answer : 05

```
public class ArraySumCalculator {
    public static int calculateSum(int [] array){
    int sum = 0;
      for(int num : array){
      sum += num;
    }
    return sum;
}
public static void main(String [] args){
    int [] numbers = {10, 20, 30, 40, 50};
    int result = calculateSum(numbers);
    System.out.println("The sum of all elements
    in the array is: " + result);
}
}
```

Answer - 6

Access modifiers define the visibility or scope of class, methods and variables in Java.

Comparison

1. public - Accesible from anywhere
2. Private - Accesible only within the same class
3. Protected - Accesible within the same package and subclasses
4. Default - Accessible only within the same package

Type of variable in Java

Java has three types of variable

1. class variables:
   * Belongs to the class rather than an instance
   * Shared among all objects of the class
   * Memory allocated once

2. Instance variable:

* Defined inside a class but outside methods
* Each object has its own copy
* Memory allocated in the heap

3. Local variables:

* Declared inside a method, constructor or block
* Accessible only within the method/block
* Must be initialized before use

Ans NO: 07

```
import java.util.Scanner;
public class QuadraticSolver {
    public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter coefficients a, b and c: ");
    int a = scanner.nextInt();
    int b = scanner.nextInt();
    int c = scanner.nextInt();
    double discriminant = b*b - 4*a*c;
if (discriminant < 0) {
    System.out.println("No real root.");
} else {
double root1 = (-b + Math.sqrt(discriminant)) / (2*a);
double root2 = (-b - Math.sqrt(discriminant)) / (2.0* a);
double smallestPositiveRoot = Double.MAX_VALUE;
```

```
if (root1 > 0 ) smallest positiveRoot = root1 ;
if (root2 > 0 && root2 < smallestpositiveRoot)
        smallestPositiveRoot = root2;
if (smallestPositiveRoot == Double.MAX.VALUE) {
  System.out.printIn ("No positive real roots");
 } else {
 System.out.printIn ("The smallest positive
        root is :" + smallestpositive Root);
  }
 }
 scanner.closel );
 }
}
```

Ans No: 8

```
import java.util.Scanner;
public class characterCounter {
    public static void main (String[] args) {
    Scanner scanner = new Scanner (system.in);
    System.out.print ("Enter a string:");
    string input = scanner.nextline ();
    int letterCount = 0, digitcount = 0, spaceCount = 0;
    for (char ch: input.to charArray ()) {
     if (character.isLetter (ch)) {
       digit Count++;
     } else if (character.isDigit (ch)){
       digit Count++;
     } else if (character.isWhitespace (ch)){
       spaceCount++;
     }
     System.out.println ("Letters: " + letterCount);
     System.out.println("Digits: " + digitCount);
     System.out.println(" Whitespaces: " + spaceCount);
     scanner.close();
}
```

Passing an array to a function

```
public class ArrayExample {
    static void printArray(int[] arr){
    for (int num: arr){
     System.out.print (num + " ");
    }System.out.println();
}
public static void main (String[] args){
int[] numbers = {10, 20, 30, 40, 50};
System.out.println("Array elements:");
printArray (numbers);
}
}
```

Answer:09:

Method overriding occurs in Java when a subclass provides a specific implementation of a method that is already defined in its superclass. The overriden method in the subclass must have method overriding. Subclass a new implementation of superclass method with the same signature. ~~dinamic~~ Dynamic Method Dispatch: Overridden methods are called based on the actual object type at runtime

using (super class) = calls the super class version of an overridden method or constructor

Access Modifiers Rule: cannot make the overridden method more restrictive than the super class

Checked Exceptions: subclass cannot throw broad exceptions than the overridden method in the subcla

final methods: A methods declared final cannot be overrudd

static method: cannot be overridden tg they are hidden instea

constructors: Not inherited but can call the superclo constructors using super();

Ans no: 10

Static methods

1. Declared using static keyword
2. stored in the class memory
3. can be accessed using the class name
4. Belongs to the class rather than an object
5. Used for common utilities like constaks and helper methods
6. Example : static int cant;

Non static methods

1. Dectared without static keyword
2. stored in the heap memory
3. Requires an instance of the class
4. Belongs to the individual objects
5. Used for instance spectic behavior
6. Example : int age;