ID: IT-23024

Topic Name : _____ Day : _____

Time : Date : / /

1. Write a Java program that reads a series of numbers from a file input.txt determines the higest number in the series, calculates the sum of natural numbers up to the highest number, and write the result anothe file 'output.txt. Use scanner to read from the file and PointWriter to write to the file Assume the numer in the input file separately.

Sample Input.txt:

10, 55, 1000, ... 100

Output.txt:

55, 1540, 500500 5050

The code is given below:

IT-23024

Topic Name : _____ Day : _____

Time : _____ Date : / /

Code:

```
import java.io.File;
import java.io.PrintWriter;
import java.util.Scanner;

public class ReadFile_sumseries {
    public static void main(String[] args){
        try{
            File inFile = new File("input.txt");
            Scanner in = new Scanner(inFile);
            PrintWriter out = new PrintWritter(new file("output.txt");
        if (in.hasNextLine()) {
        String line = in.nextLine();
        String [] nums = line.split(",");
        for (int i=0; i< nums.length; i++) {
            int n = Integer.parseInt(nums[i]);
        int sum = (n*(n+1))/2);
        out.print(sum);
        if (i! = nums.length -1){
            out.print(",");
        }
        }
        }
        in.close();
        out.close();
```

```
        System.out.println("File written successfully.");
    } catch (Exception e){
    System.out.println("File not found:");
    }
    }
}
```

2. what are the difference you have even found between static and final fields and method Exemplify what will happen if you try to access the static method/field with the object instead of class name?.

Ans:

The difference between static and final field is given below:

IT-23024

Topic Name : _____ Day : _____

Time : Date : / /

Feature	satatic	final
Definition	Belong the class, not instances	Once, assigned, cannot be change
Fields	Shared among all intances	Value cannot be modified once intialized
Methods	Belongs to the class, can be called with out an instance	Cannot be overridden by subclasses
Usage	Uses for constants utility methods and share data	Used to enforce immutability and prevent metuad overriding

Example of static and final field:

11-29024

Topic Name :

Day :

Time : Date : / /

```
class Example {
    static int staticVar = 10;
    final int finalVar = 20;

    static void staticMethod() {
     System.out.println("Static Method Called");
    }

    final void finalMethod() {
     System.out.println("Final Method called");
    }
}

public class Test {
    public static void main(String[] args) {
        Example obj = new Example();

        obj.staticMethod();
        System.out.println(Example.staticVar);
    }
}
```

IT-23024

Topic Name : _____ Day : _____

Time : Date : / /

If I Access Static Members Using an Object

1. It complie and run successfully but not recomanded. Java allows calling static member
2. Using and object neference, but it gives a warning because static member belong to the class not the object.

2. Final Field behave normally finalVar must be intialized eithen during deelanation or in the constructon.

3. Final method can be accessed normally but cannot be overnide in subclass.

IT-23024

Topic Name : _____ Day: _____

Time: Date: / /

3. Write a java program to find all factorion ~~um~~ numbers within a given range. A number is called a factorian if the sum of the factorials of its digits equals the number itself. The program should take user input for the lower and upper bound

Code:

```
import java.util.Scanner;

public class FactorionFinder {
    public static void main (String[] args) {
    Scanner scanner = new Scanner (System.in);
    System.out.print ("Enter lower bound; ");
    int lower = scanner.nextInt();
    System.out.print("Enter upper bound: ");
    int upper = scanner.nextInt();
    scanner.close();
    for (int i= lower; i<=upper; i++) {
    if (isFactorion(i)) System.out.print (i + " ");

    }

}
```

IT-23024

Topic Name :

```
private static boolean isFactorion(int num){
    int sum=0, temp=num;
    while(temp>0){
        sum+=factorial(temp%10);
        temp/=10;
    }
    return sum==num;
}
private static int factorial(int n){
    int[] fact={1,2,6,24,120,720,5040,
        40320,362880};
    return fact[n];
    }
}
```

IT-23024

Topic Name : _____ Day : _____

Time : Date : / /

4. Distinguish the difference among class, local, and instance variables. what is significance of this keyword?

Feature	Class variable	Instance variable	Local variable
Definition	A variable defined with the static keyword inside a class, shared by all instance of the class	A variable inside a class but outside any method unique to each object instance	A variable defined inside a method, construcion, or block.
Scope	Class-wide, accessible to all instances of the class	object-specific, exists as long as the object exists.	method or block-specific, exists only during the method execution
Lifetime	As long As class is loaded in memory	As long as the object instance exists	Limited to the execution time of the method block
memory	stored in the static memory	stored in the heap as part of the object instance	stored in the stack memory during method execution.
Initilization	Intitilize when the class is loaded	Intialize when the object is created	must be intilized before use.

11-3-2024

Topic Name : _____ Day : _____ Time : _____ Date : / /

The significace of the this keyword:

Definition: The this keyword is a keyword is a reference variable refering to the current instance of the class. It is often used inside instance method on constructors to refer to the current object.

Uses:

1. Distinguishing Instance variable: It helps distinguish instance variable for parameters or local variable that have the same name.

2. Accessing Instance Methods / Variables: It allows access to instance variables and methods.

3. Passing current object: It can be used to pass the current object as a parameter to another method or constructor.

Example:

```
class myClass {
    int instanceVar;
    Myclass (int instanceVar) {
        this.instanceVar = instanceVar;
    }
}
```

I I-23024

Topic Name : _____ Day : _____

Time : Date : / /

5. Write a java program that defines a method to calculate the sum of all elements in an integer array. The method should take an integer array as parameter and return the sum. Demonstrate this method by passing an array of integers from the main method:

code:

```
public class SumArray {
    public static int calculateSum (int[] arr){
        int sum = 0;
        ~~for( int sum =~~
        for ( int num : arr) {
            sum += sum;
        }
        return sum;
    }
    public static void main ( String[] args){
        int [] numbers = { 1, 2, 3, 4, 5, 6};
        int sum = calculatesum (numbers);
        System.out.println ("The sum of the array elements
                is: " + sum);
    }
}
```

IT-23094

Topic Name : _____ Day : _____

Time : Date : / /

6. What is called aecess modifiers? compare the aecesibity of Public, Private and proeted mat modifiers desenibe the different type of variable in Java with example.

An aecess modifiers in Java is a keyword used to specify the visibility or aceessibity of classes, methods, variables, on constructors.

The main aecess modifiers are:

1. public
2. private
3. proeted
4. default

Modifier	Aecessibility	Desenription
public	Accessable from anywhere	Member deeelaned as public and can be aecess from anywhere
private	Aecessable only within the same elass	Member declared as private not aceessable from any where. They can only be aceessed with in the class they are defined
Protected	Accessable within the same packadge and by subclass	member deelare as proteeted

IT-22024

Topic Name : _____ Day : _____

Time : Date : / /

Variable are :

1. Instance variable
2. Class variable
3. local variable
4. Parameter

1. Instance variable : A variable defined in a class but outside methods. Each object of the class has it own copy.

Code:

```
class car {
    String model;
    void displaymodel(){
        System.out.println("Model : " + model);
    }
}
public class Test {
    public class Test() static void main(String[] args){
        Car car1 = new Car();
        car1.model = "Toyota";
        car1.displayModel();
    }
}
```

2. Class variable:

A variable declared with the static keyword.

Shared by all instances of the class.

Example:

```
class car {
    static int count = 0;

    car()
      count++;

    }
}
public class Test{
    public static void main ( string[] args) {

      new car();
      new car();
      System.out.println ("Total cars: " + Car.count)

    }
}
```

IT-23024

Topic Name : _____ Day: _____ Date: / /

3. local variable: A variable declred inside a method or block. Only accesible with in that method or block.

Example:

```
class Example {
    void addNumbers (){
      int num1 = 5, num2 = 10;
      int sum = num1 + num2;
      System.out.println ("Sum: " + sum);
    }
}
public class Test{
    public static void main (String[] args){
      Example obj = new Example();
      obj.addNumbers();
    }
}
```

II-23024

Topic Name : _____ Day : _____

Time : Date : / /

4 Parameter: A variable passes to a method. Only accessble within the method

```
class Example{
    void printMessage( string message)
     System.out.println(message);
    }
 }
public class Test{
     public static void main (string[] args)
     {
     Example obj = new Example ();
     obj.printMessage ("Hello, Java!");
   }
}
```

IT-23024

Topic Name : _____ Day : _____

Time : Date : / /

7. Write a Java program to find the smallest positive root of a quadratic equation of the form.

$$ax^2 + bx + c = 0$$

Using the quadratic formula,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2 \cdot a}$$

The program should:

1. Take integer input ~~for~~ coefficient a, b and c,
2. Compute the root using Math.sqrt (double a).
3. Determine the smallest positive root using Math.min (double a, double b)
4. Print the smallest positive root if real root exist. otherwise print " No real roots".

Sample Input: a, b and c : 1, -3, 2

Sample output:

The smallest positive root is : 1.0;

IT-23024

Topic Name : _____ Day : _____ Time : Date : / /

Code:

```
import java.util.Scanner;

public class QuadraticEquation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the co-efficient a, b and
            c: ");
        int a = scanner.nextInt(), b = scanner.nextInt();
            c = scanner.nextInt();
        double discriminant = b*b - 4*a*c;
        if (discriminant >= 0) {
            double root1 = (-b + Math.sqrt(discriminant)) / (2*a)
            double root2 = (-b - Math.sqrt(discriminant)) / (2*a
        if (root1 > 0 && root2 > 0) {
            System.out.println("The smallest positive root is:"
                + Math.min(root1, root2)); }
        else if (root1 > 0) {
            System.out.println("The smallest positive root
                is: " + root1); }
```

```
else if (root2>0) {
  System.out.println("The smallest positive root is: "
                     + root2); }

  else {
    system.out.println("No real roots.");

  }
  scanner.close();
  }
}
```

8. write a program that can determine the letter, witespace and digit. How do we pass an array to fonetion? write an example:

Code:

IT-23024

Topic Name : _____ Day : _____ Time : Date : / /

```
public class CharacterType{
    public static void determin CharacterType
      (char[] chars){
    for( char c : chars){
    if ( character. isLetter (c)){
      System. out. println (c + " is a letter. ");
    }
   else if (Character. isDigit ((c)){
      system.out.println ( c + "is a digit.");
    } else if ( character. eisWhitespace ((c)){
      System.out. println ( "white space detected.");
      }
    }
  }
public static void main ( string [] args) {
      char[] charactors = { 'A', ' ', '5', 'b',
      determineCharacterType (charactors);
      }
}
```

IT-23024

Topic Name : _____ Day : _____

Time : Date : / /

9. In Java, explain how method ~~over~~ overriding works in the context of inheritance. what happens when a subclass override a method from it superclass. How does the super class help calling superclass method, what are the potential issus when overriding methods, especially when dealing with constructor.

Method Overriding allows a subclass to provide a specific implementation of a method already defined in it superclass. This is a core feature of inheritance in object oriented programming and is essential for achieving runtime polymorphism.

How super Helps:

The super keyword is used to call a method on constructor from the superclass.

super.methodName() is used to call the superclass's overridden method from subclass.

This is useful when the subclass wants to add extra behaviour to superclass method instead of completely replacing it.

11-29024

Topic Name : _____ Day : _____ Time : Date : / /

10. Differentiate between static and non-static members including necessary example. Write a program that able to cheak either a number or string is palingdom or not.

Aspect	static members	Non-static members
memory allocation	Allocate once for the class	Allocate separately for each object
Access	Accessed using class name	Accessed using an object
Scope	Shared across all instance	Unique to all instance
Usage	class-level functionally constants	Intance level functionality

Cheak a number for palingdom or no

IT-23024

Topic Name : _____ Day : _____

Time : Date : / /

11. what is called class abstraction and encapsulation? desenibe with example. what are the difference between abstract a class and Interface?

Ans:

Abstraction:

Abstraction is the concept of hiding the internal implementation details of a class and only exposing essential features. It allows the user to focus on what object does rather than how it does it.

Example of Abstraction using an abstract class.

```
abstract class Vehicle {
    abstract void start();
}
class Car extends Vehicle {
    void start() {
        System.out.println("Car starts with a key");
    }
```

TT-23024

Topic Name : _____ Day : _____

Time : Date : / /

```
public class main {
    public static void main ( String[] args) {
        Vechicle myCar = new car();
        myCar.start();
    }
}
```

Encapsulation:

Encapsulation is the practice of wrapping data variables and code (methods) into a single unit clas and restricting direct access to some details.

Example:

```
class person {
    private string name;
    public void setName
        this.name = name;
    }
public String getName (){
    return name;
}
}
```

IT-23024

Topic Name : _____ Day : _____

Time : Date : / /

```
public class Encap {
    public static void main( string[] args) {

        Person p= new person();
        P.setName ("John");

    System.out.println ( P. getName());

    }

}
```

Difference between Abstract a class and Interface

Points	Abstract Class	Interface
Definition	A class that contain abstract methods and may have concrete method	A collection of abstract methods that must be implemented by any class that uses it.
method	can have both abstract and non-abstract methods	can only have abstract method.
variable	Can have intance variable	Only public static and final variable
Access modifier	Can have any access modifier suel as private, proteted	methods are always public by default.

JT-23024

Topic Name : _____ Day : _____

Time : Date : / /

12. Create a Java program using inheritance to perform multiple numerical operations. Implement a 'BaseClass' with common functionalities and extend it into four speeilized classs. to handle diffenent tasks. Use a 'Mainelass' to exeeute all metuod.

Answen:

```
class BaseClass {
        void pnint ( stning msg. Objeet nes)
        {
        System.out. print ( msg + nes);
    }
}

class SumClass extends Baseeluss {
        double compute sum () {
            double sum = 0;
        for ( double i=1; i>=0.1; i--=0.1)
            sum += i;
            return sum;
    }
}
```

IT-23024

Topic Name : _____ Day : _____ Time : Date : / /

```
class DivisonMultipleclass extends BaseClass{
    int ged(int a, int b)
    {
    return b == 0 ? a: ged(b, a%.b);
    }
    int lem(int a, int b)
    {
    return (a*b)/ged(a,b);
    }
    }
Class NumberConversionClass extends BaseClass{
    stringo toBinary(int num)
    {
    return Integer.toBinaryString(num);
    }
    string toHtox(int num)
    {
    return Integer.toHexstrigng(num);
    }
    string toOct(int num)
    {
    return Integer toOctalstring(num);
    }
    }
```

IT-23024

Topic Name

```
class CustomPrintClass extends BaseClass {
    void pr(String msg) {
        System.out.println("***" + msg + "***");
    }
}

public class InheritanceToPerformNumericalOP {
    public static void main(String[] args) {
        SumClass sumObj = new SumClass();
        sumObj.print("Sum:", sumObj.computedSum());
        DivisorMultipleClass divMulObj = new DivisorMultipleClass();
        int a = 36, b = 60;
        divMulObj.print("GCD", divMulObj.gcd(a,b));
        divMulObj.print("LCM", divMulObj.lcm(a,b));
        NumberConversionClass numConObj = new NumberConversionClass();
```

II 29024

Topic Name : _____ Day : _____ Time : Date : / /

```
int num = 42;
numConvObj.print("Binary: ", numConvObj.toBinary(num));

numConvObj.print("Hex: ", numConvObj.toBinary(num));

numConvObj.print("Octal: ", numConvObj.toOctal(num));

new CustomPrintClass().pr("Execution Completed");

}

}
```