**OOPs Lab file**

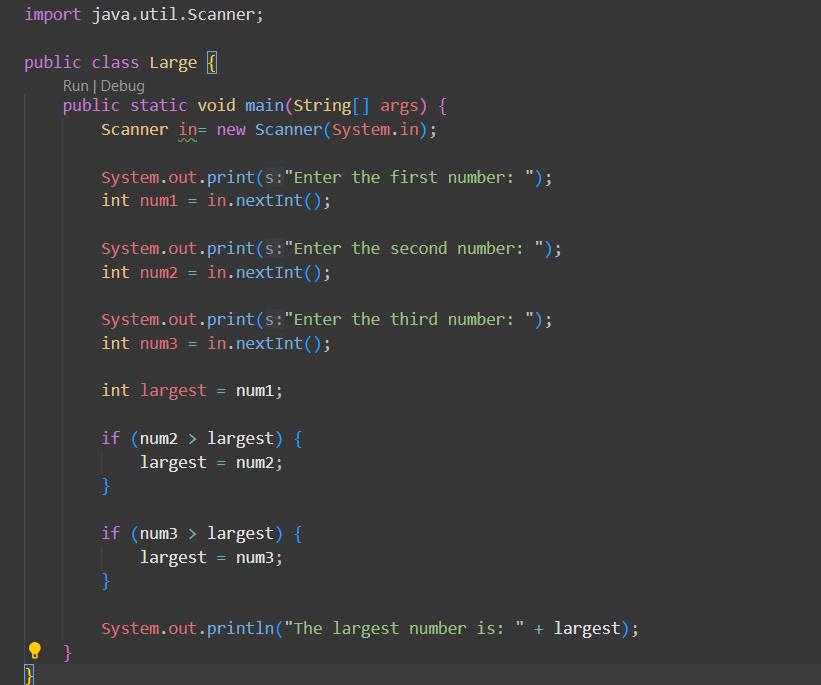
**Name : Mohammad Wasi**

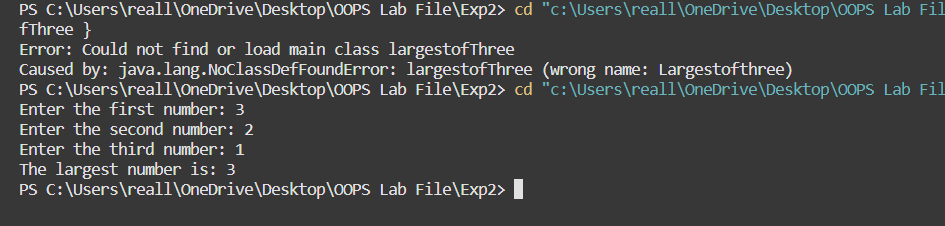
**SAP ID : 500110709**

**Submitted to : Amit Verma Sir**

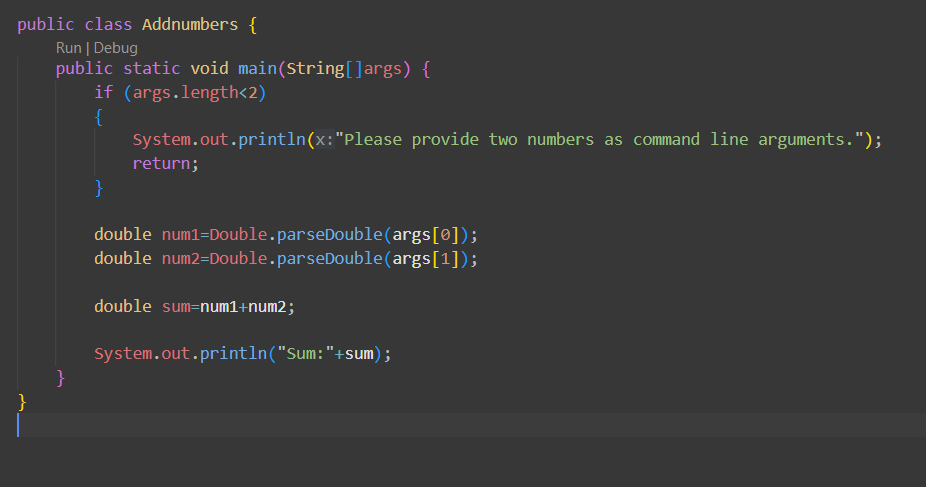
**EXPT 2.3**

**1)Write a program to find the largest of 3 numbers.**

****

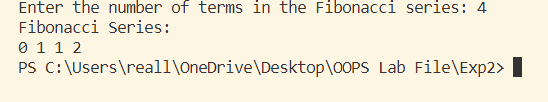
****

**2)Write a program to add two amber using command line arguments.**

****

**3) Write a program to print Fibonacci series using loop.**

****

****

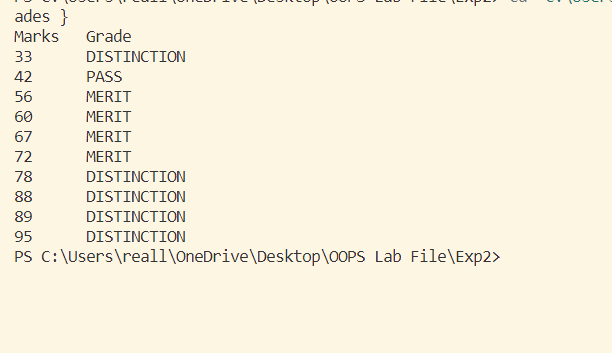
**5) Write a program to accept 10 student's marks in array, arrange it into ascending order, convert into the following grades and print marks and grades in the tabular form.**

**Between 40 and 50; PASS**

**Between 31 and 75; MERIT**

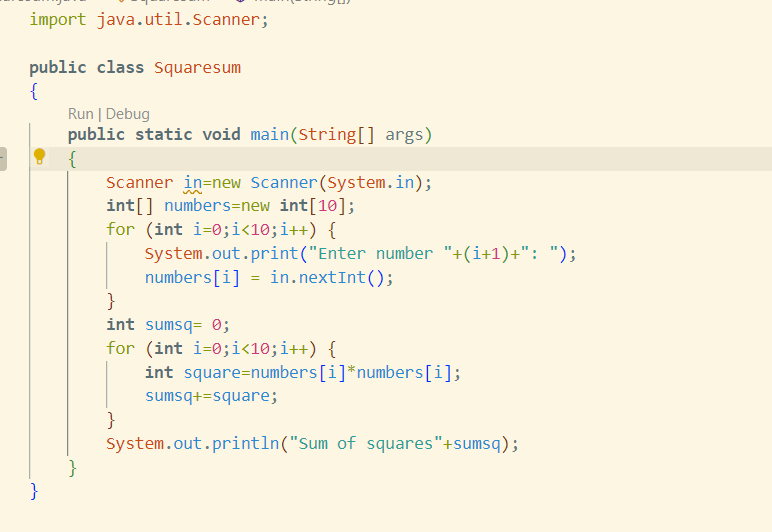
**and above: DISTINCTION**

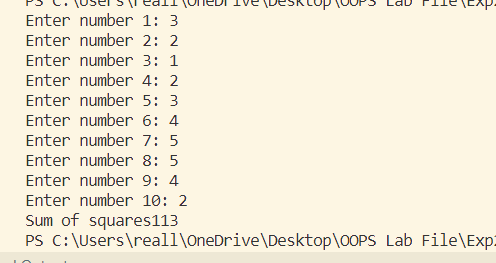
****

****

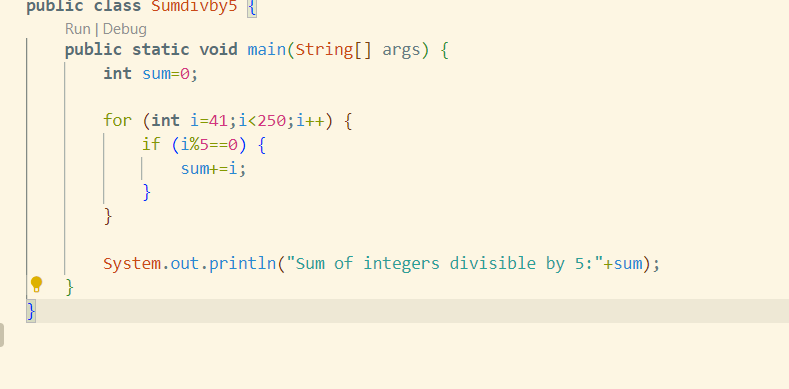
**6)Write a Java Program to accept 10 numbers in an array and compute the square of each number.**

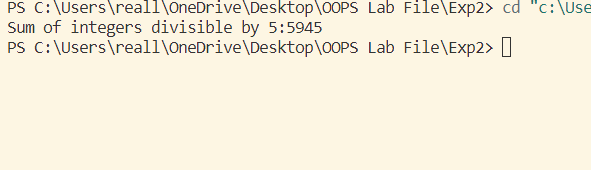
**Print the sum of these numbers.**

****

****

**7)Write a program to find the sum of all integers greater than 40 and less than 250 that are divisible by 5.**

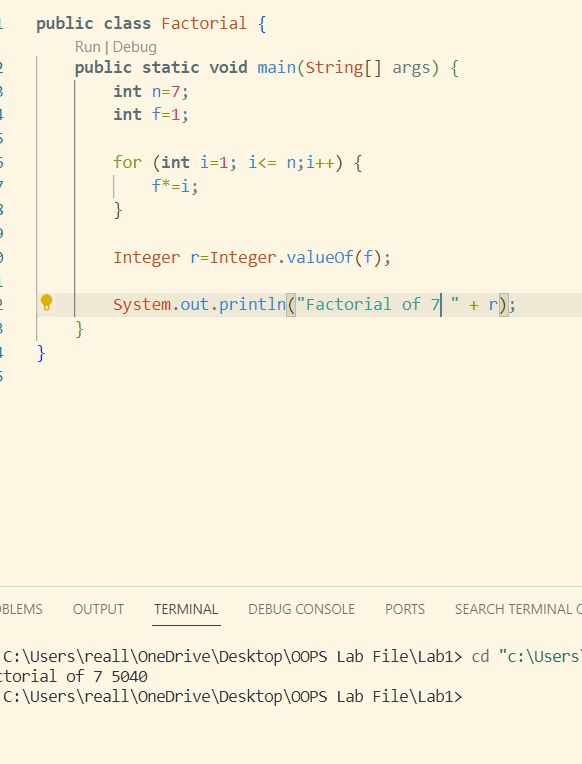
****

****

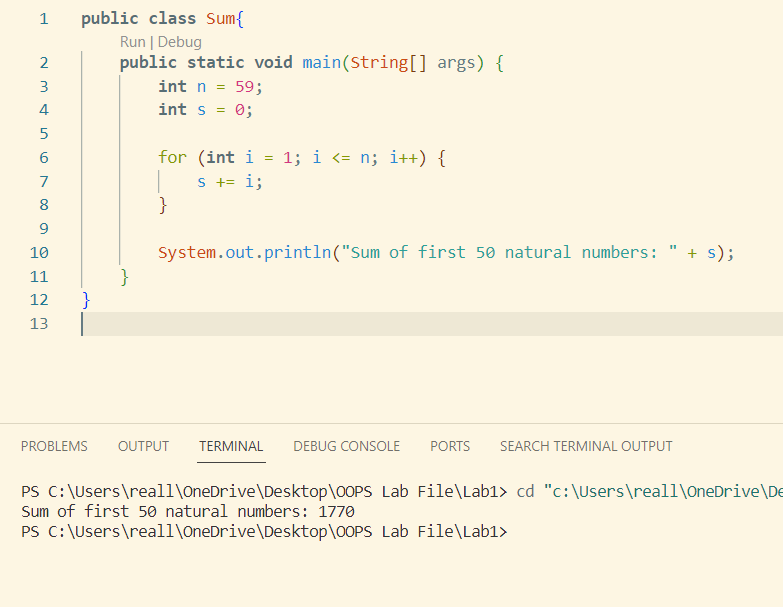
**LAB 1**

**Date – 21-8-23**

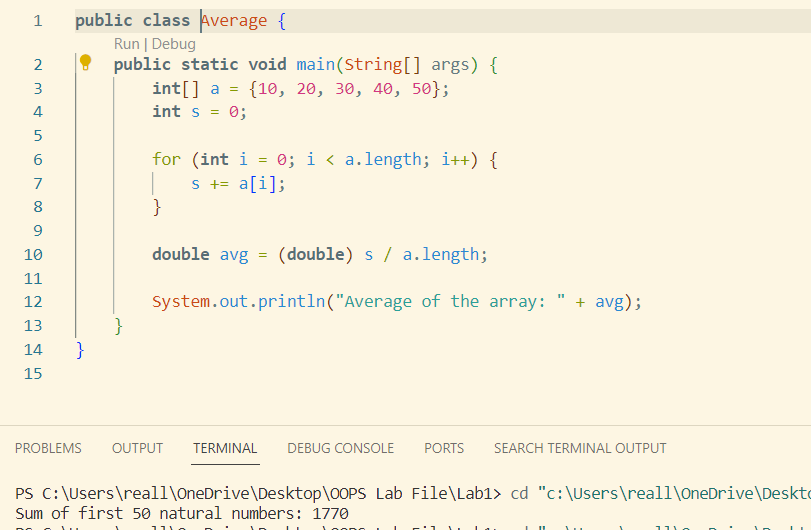
1. **WAP to calculate factorial of 5 using int variable and object of Integer wrapper class.**

****

1. **WAP to calculate the sum of first 50 natural numbers.**

****

1. **WAP to find the average of an array with 5 elements (10 to 50).**

****

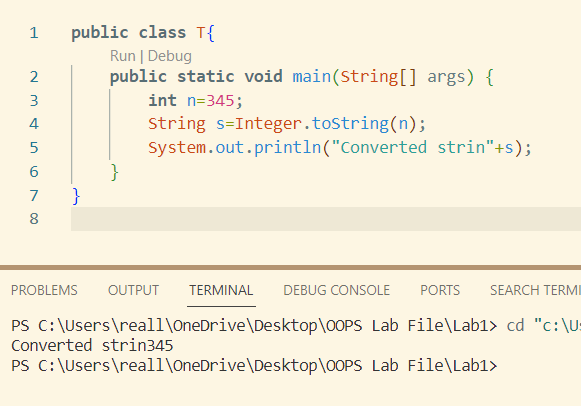
1. **WAP to parse a double from a string using parseDouble().**

****

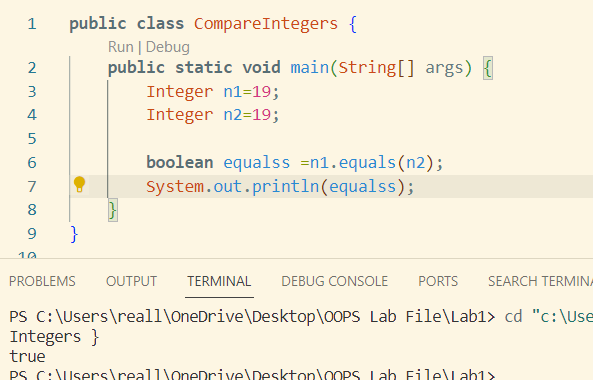
1. **WAP to parse an integer from a string using parseInt().**

****

1. **WAP to convert integer to string.**

****

1. **WAP to compare two objects of wrapper class Integer using equals.**

****

1. **WAP to print the following pattern**

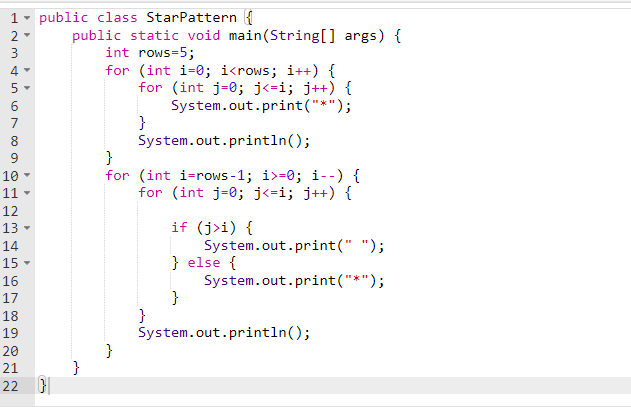
**\***

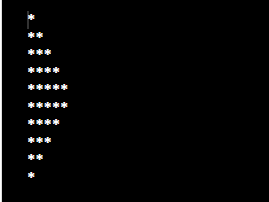
**\* \***

**\* \* \***

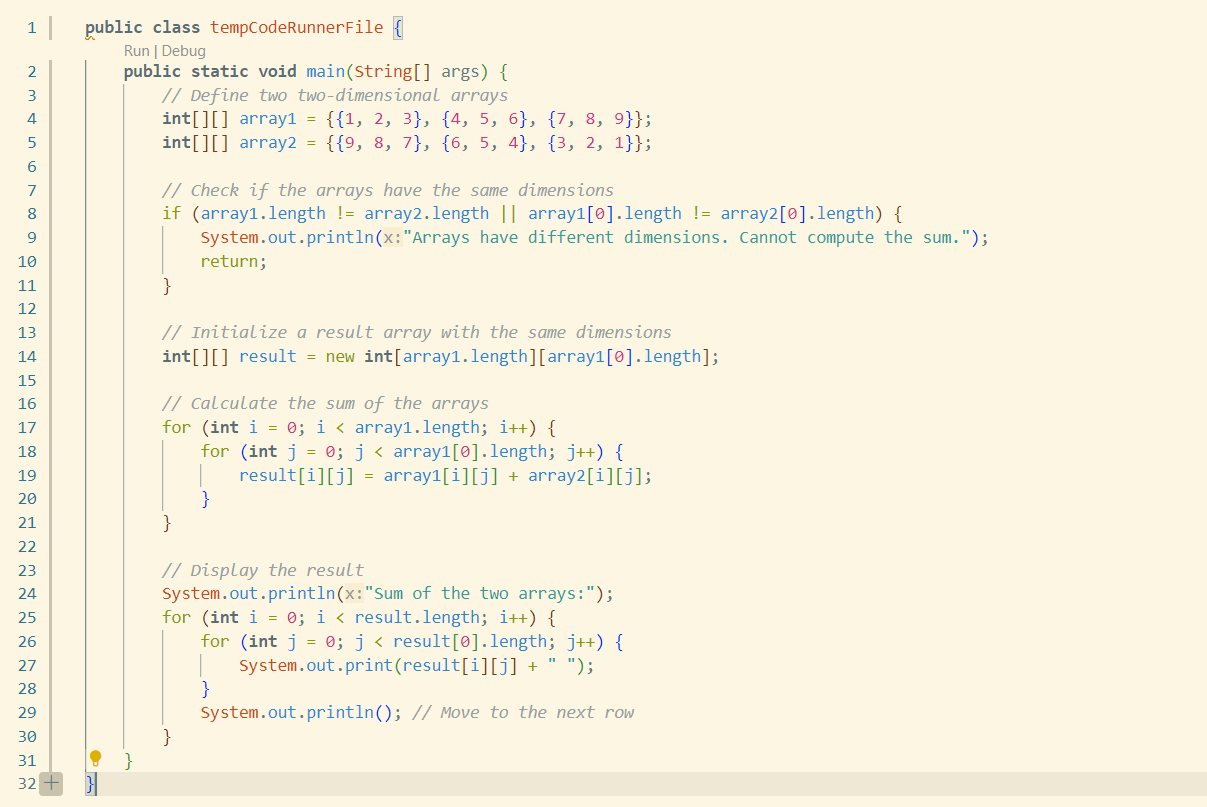
**\* \***

**\***

****

****

1. **WAP to create and initialize 2D array of 2x2 and find the sum of all elements of an array.**

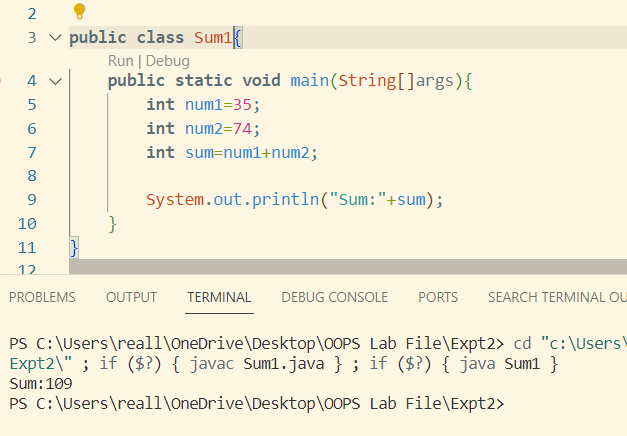
****

**Lab 2**

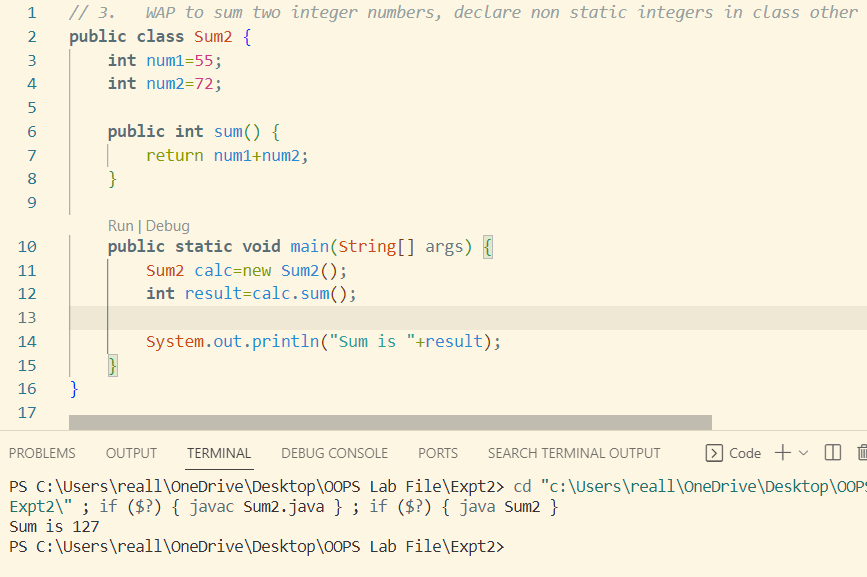
1. **WAP to print your name.**

****

1. **WAP to sum two integer numbers, declare non static integers in main function.**

****

1. **WAP to sum two integer numbers, declare non static integers in class other than the main class.**

****

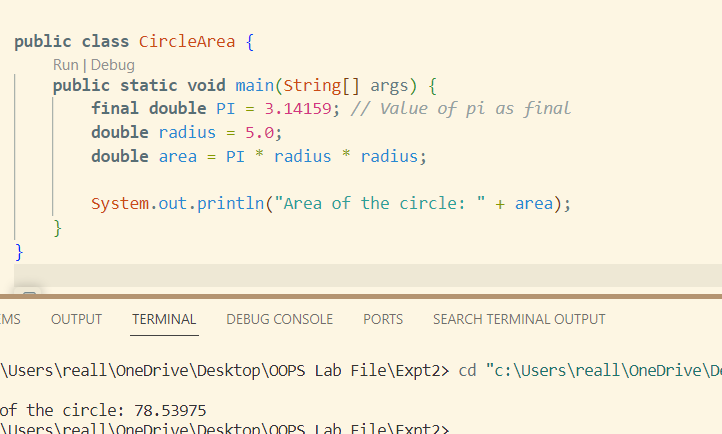
1. **WAP to sum two integer numbers, declare static integers in class other than the main class.**

****

1. **WAP to sum two integer numbers, declare non static integers in the member input function of the class other than the main class and display result using function display.**

****

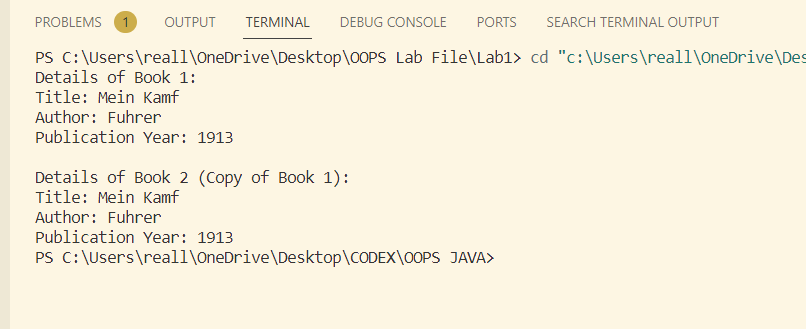
1. **WAP to calculate the area of circle (take value of pi as final).**

****

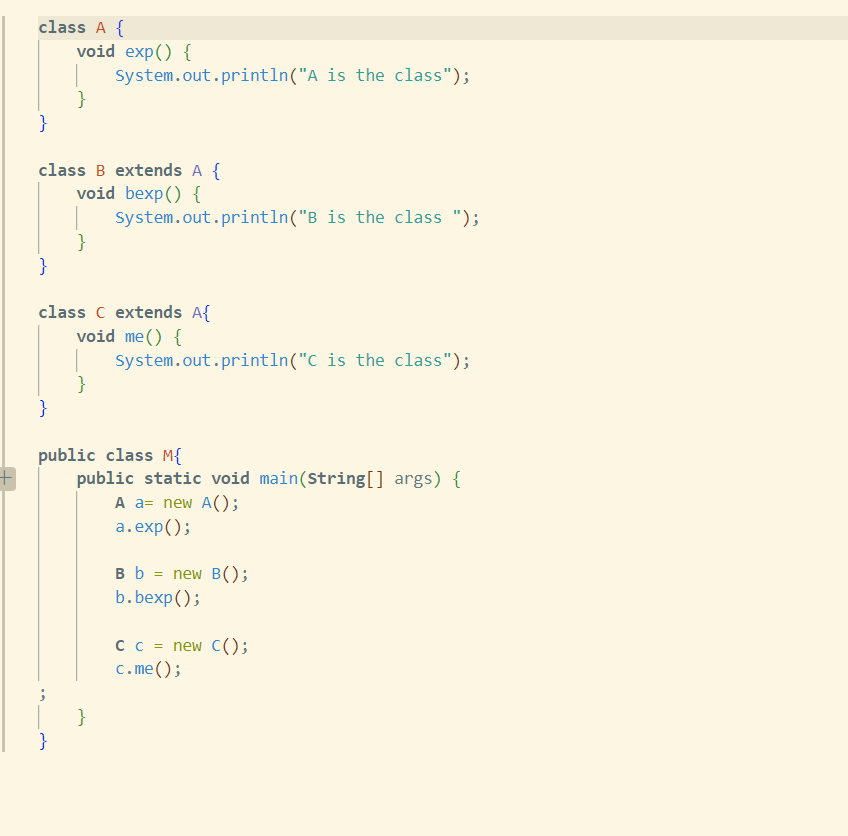
**Lab Date : 130923**

1. **Implement a class Book that contains a constructor to initialize title, author and print the details.**

****

****

**2. Show multi level inheritance**

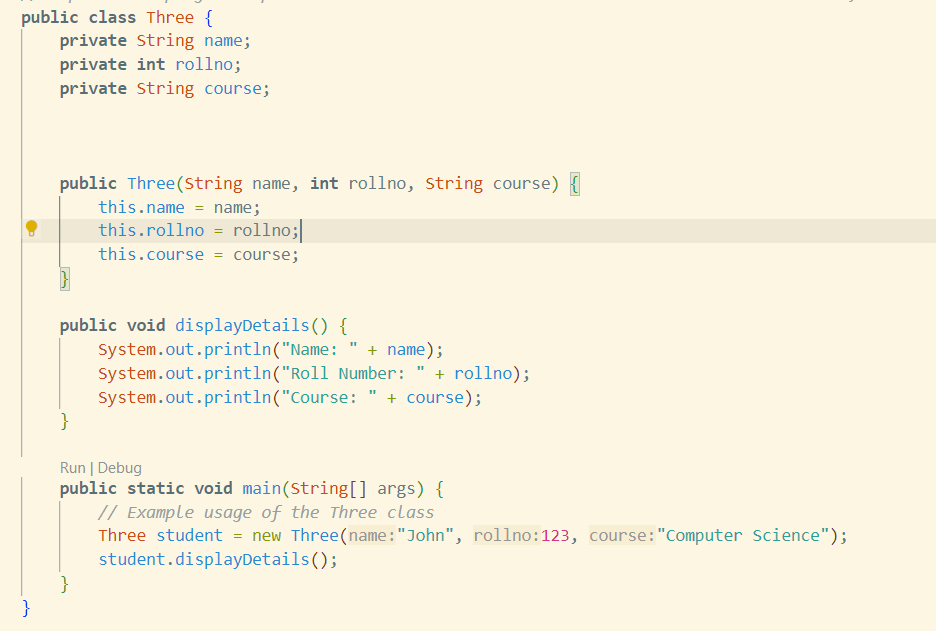
****

****

**3 *// implement a class Rectangle that contains constructor to initialize values of length and breath using this keyword, and compute its area using the function CalculateAreafunction and return and display area  in java***

****

**4 *// implement a program A parametrized constructor that initializes the student's name , roll number , course .A method called displayDetails that prints all the details i***

****

**Experiment Inheritance**

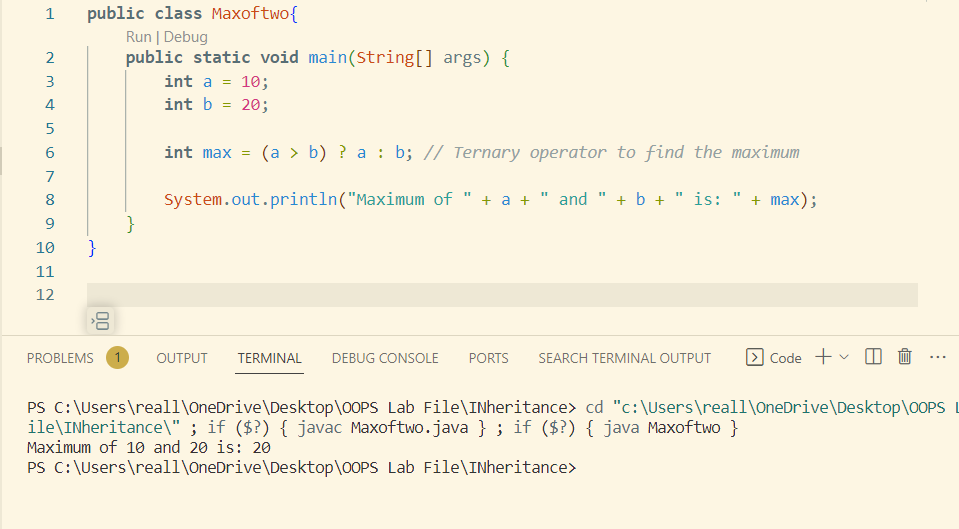
1. **Implement a program to left-shift and right-shift a binary number by a specified number of positions.**

****

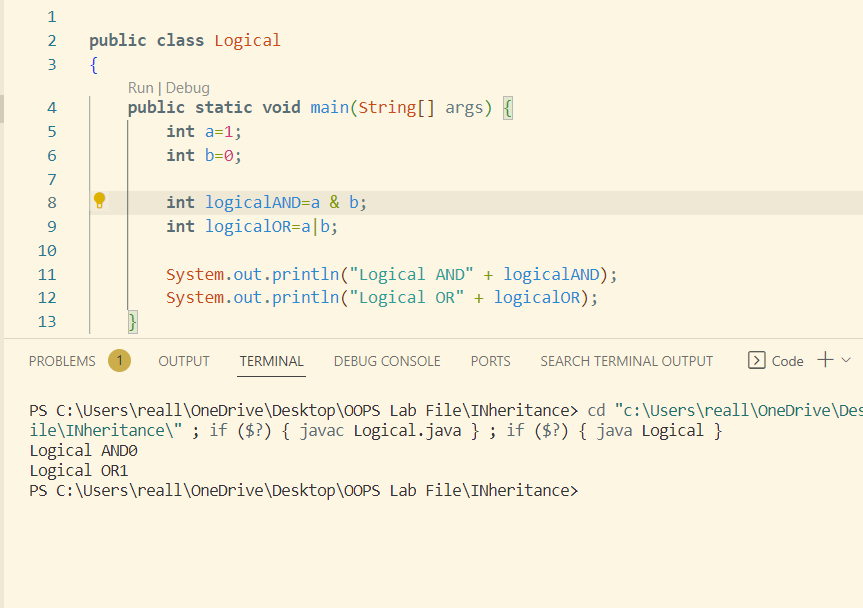
1. **Write a program to simulate logical AND and OR operations on boolean.**

****

1. **Write a program to find the maximum of two numbers using the ternary operator (a > b ? a : b).**

****

1. **Write a program to simulate logical AND and OR operations on Boolean**

****

**5,. Create a base class (superclass) and a derived class (subclass) with a few attributes and methods. Demonstrate inheritance by accessing superclass and subclass members.**

**class A {**

**private int numberA;**

**public A(int numberA) {**

**this.numberA = numberA;**

**}**

**public void displayInfoA() {**

**System.out.println("Number in A: " + numberA);**

**}**

**}**

**class B extends A {**

**private int numberB;**

**public B(int numberA, int numberB) {**

**super(numberA);**

**this.numberB = numberB;**

**}**

**public void displayInfoB() {**

**System.out.println("Number in B: " + numberB);**

**}**

**}**

**class C extends A {**

**private int numberC;**

**public C(int numberA, int numberC) {**

**super(numberA);**

**this.numberC = numberC;**

**}**

**public void displayInfoC() {**

**System.out.println("Number in C: " + numberC);**

**}**

**}**

**public class InheritanceDemo {**

**public static void main(String[] args) {**

**B myB = new B(10, 20);**

**C myC = new C(30, 40);**

**myB.displayInfoA();**

**myB.displayInfoB();**

**myC.displayInfoA();**

**myC.displayInfoC();**

**}**

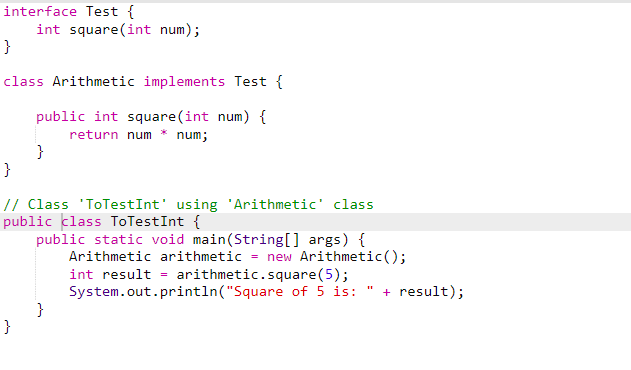
**}**

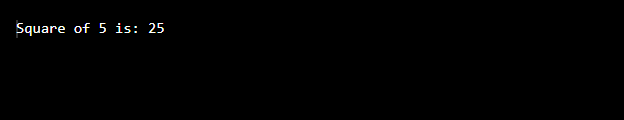
**Experiment**

**…Interface…**

**1) Write a program to create interface named test. In this interface the member function is square. Implement this interface in arithmetic class. Create one new class called ToTestInt. In this class use the object of arithmetic class.**

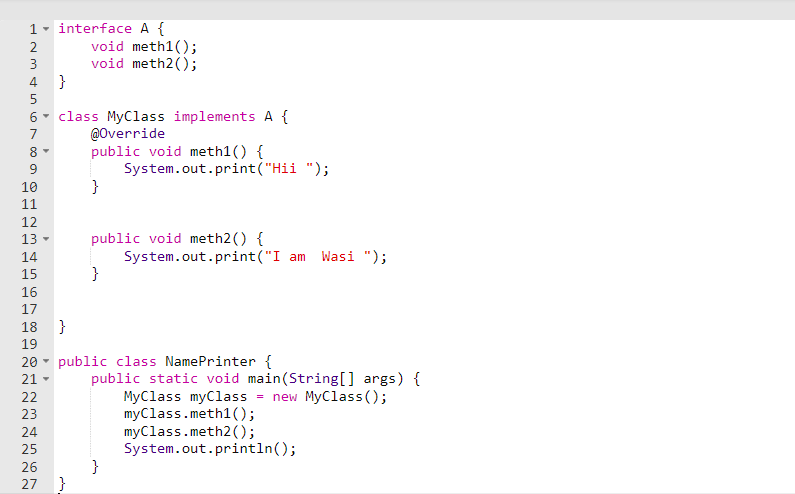
**Approach:** Define 'Test' interface with 'square' method. Implement it in 'Arithmetic' class for square calculation and use 'ToTestInt' to calculate and display the square.

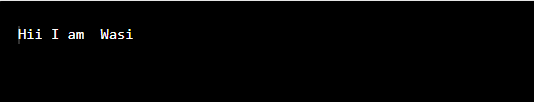
****

****

**2) Write a program to create interface A, in this interface we have two method meth1 and meth2. Implements this interface in another class named MyClass.**

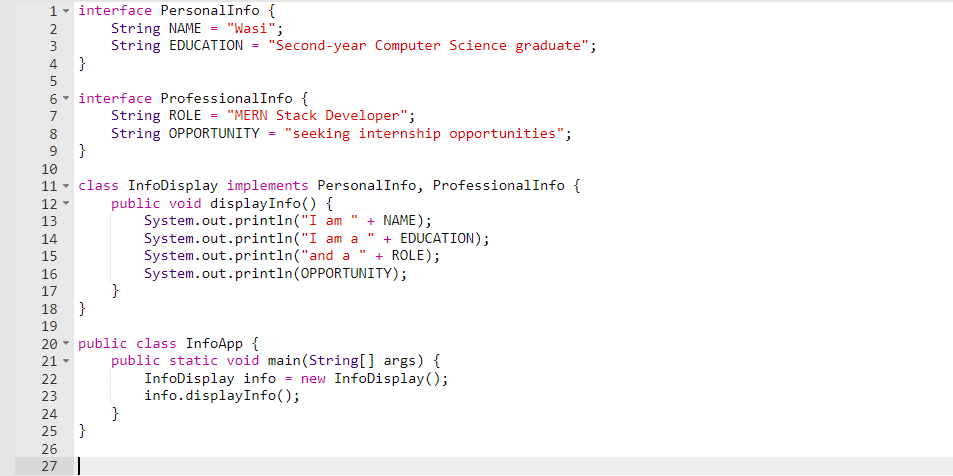
**Approach** : Create an interface 'A' with 'meth1' and 'meth2'. Implement it in 'MyClass' to print "Hii I am Wasi" using an object of 'MyClass'.

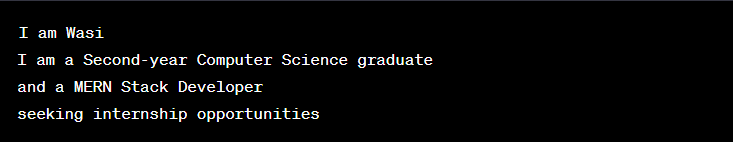
****

****

**3) Write a program in Java to show the usefulness of Interfaces as a place to keep constant value of the program**

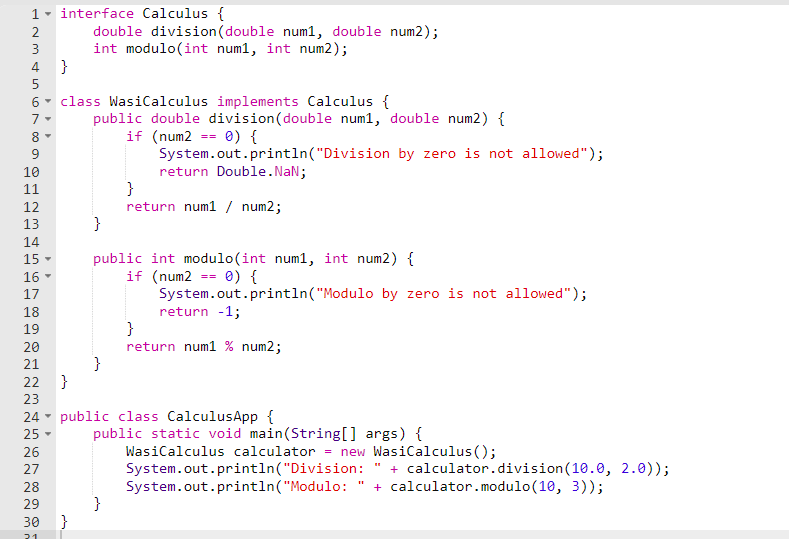
**Approach :** Define 'PersonalInfo' and 'ProfessionalInfo' interfaces to store constants. Implement 'InfoDisplay' to print personal and professional information.

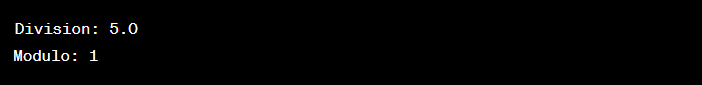
****

****

**4) Write a program to create an Interface having two methods division and modules. Create a class, which overrides these methods.**

**Approach** : Define an interface 'Test' with a 'square' method and implement it in 'Arithmetic' class to calculate the square of a number using the 'ToTestInt' class.

****

****

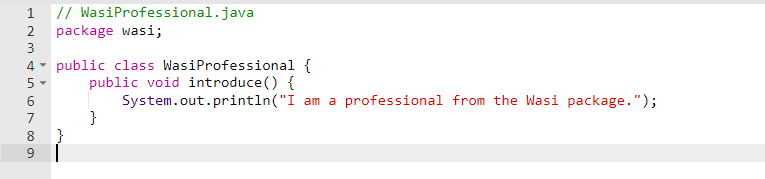
**Experiment**

**Packages**

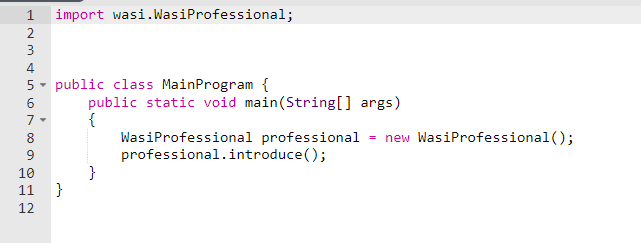
**1) Write a Java program to implement the concept of importing classes from user defined package**

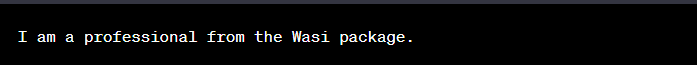
**and created packages.**

Creating Package WasiProfessional to be imported

****

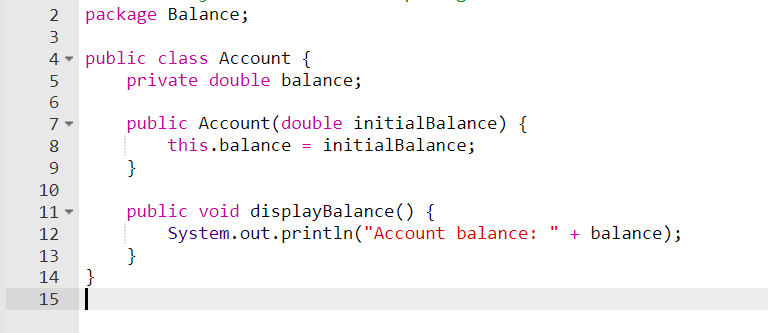
Importing the WasiProfessional class



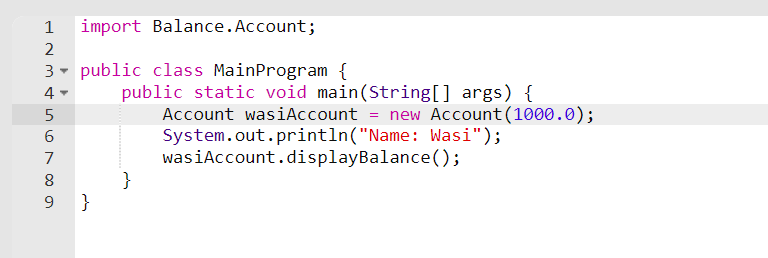
****

**2) Write a program to make a package Balance. This has an Account class with Display Balance method. Import Balance package in another program to access Display Balance method of Account class.**

Creating Package balance to be imported

****

Importing the Package

****

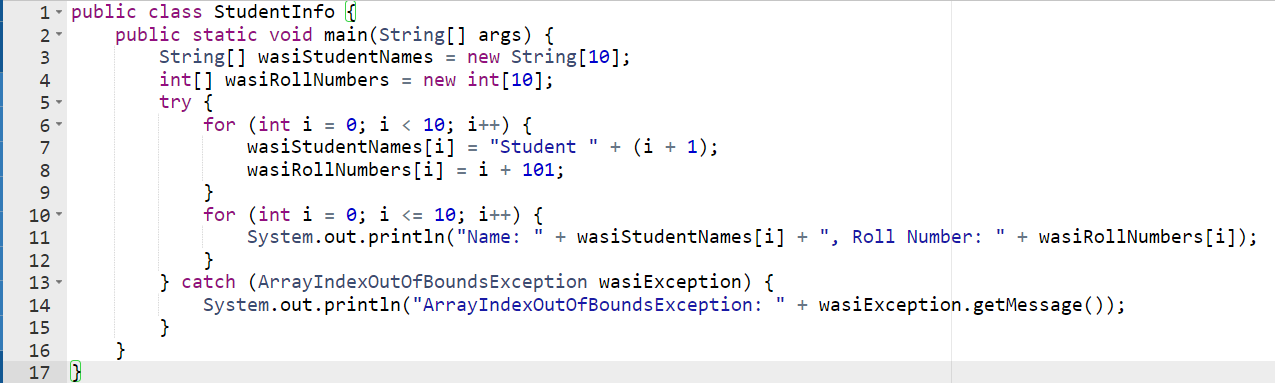
**Experiment**

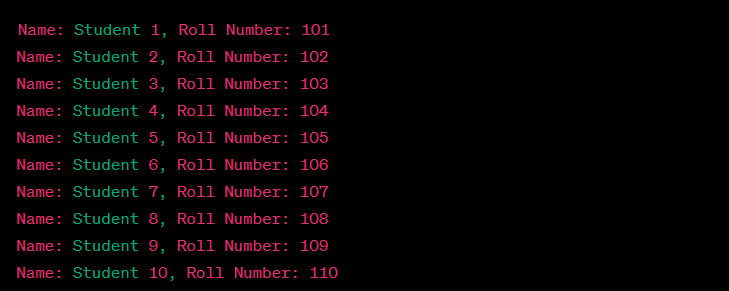
**Exception Handling**

1. Write a program in Java to display the names and roll numbers of students. Initialize respective array variables for 10 students. Handle ArrayIndexOutOfBoundsExeption, so that any such problem doesn’t cause illegal termination of program.

**Approach**

Initialize student name and roll number arrays. Use a try-catch block to handle ArrayIndexOutOfBoundsException while populating and displaying student details.

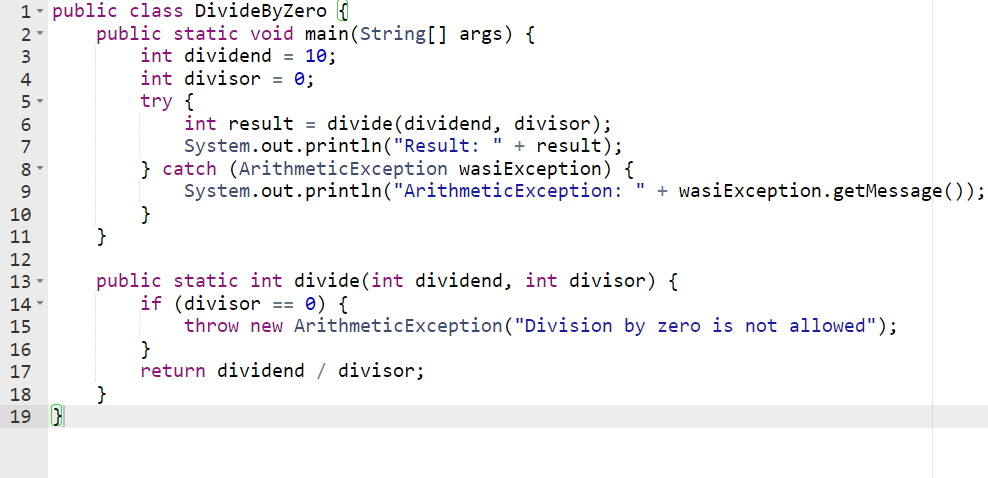


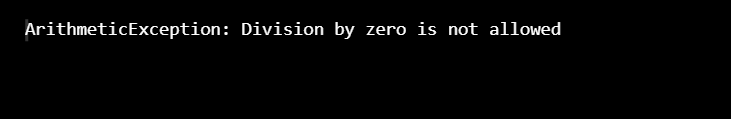


1. Write a Java program to enable the user to handle any chance of divide by zero exception.

**Approach**

Set dividend and divisor, use a try-catch block to catch ArithmeticException for division by zero, and display the result.

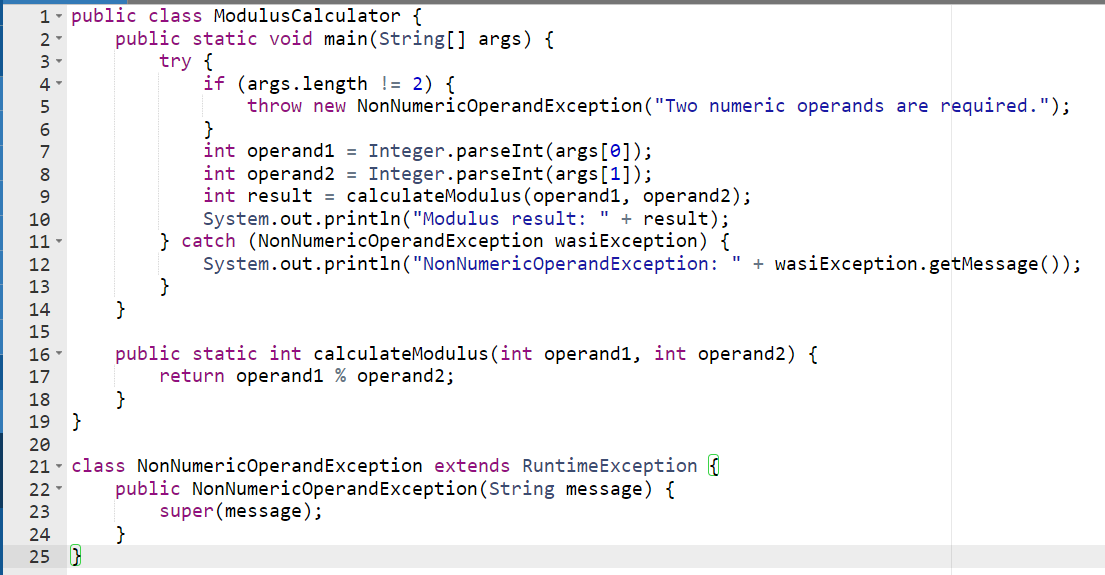




1. Create an exception class, which throws an exception if operand is nonnumeric in calculating modulus. (Use command line arguments).

**Approach**

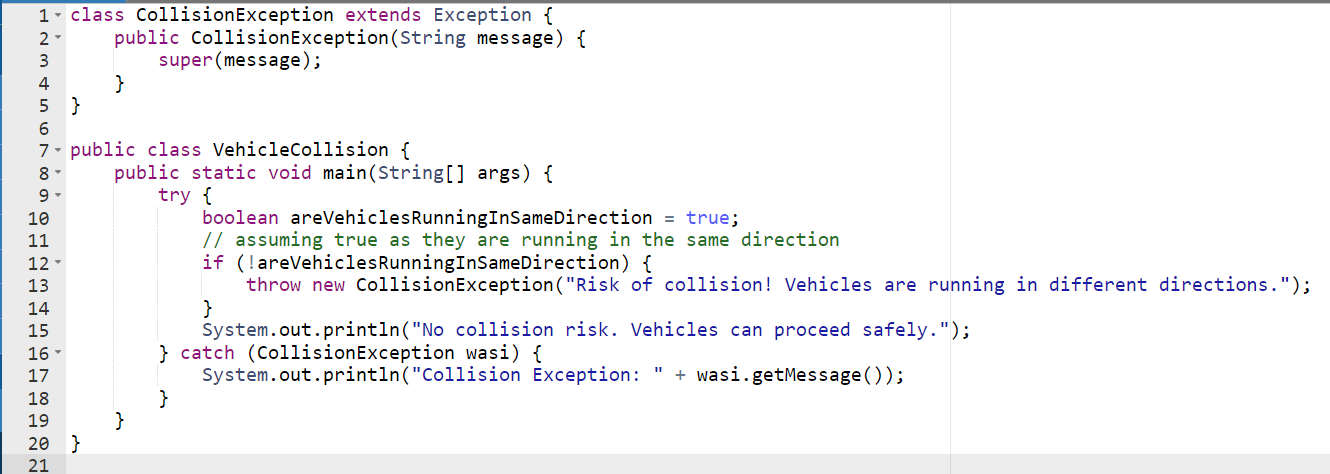
Check if two command-line arguments are provided. Handle a custom NonNumericOperandException when operands are non-numeric, calculate the modulus, and display the result.



1. On a single track two vehicles are running. As vehicles are going in same direction there is no problem. If the vehicles are running in different direction there is a chance of collision. To avoid collisions write a Java program using exception handling. You are free to make necessary assumptions.

**Approach**

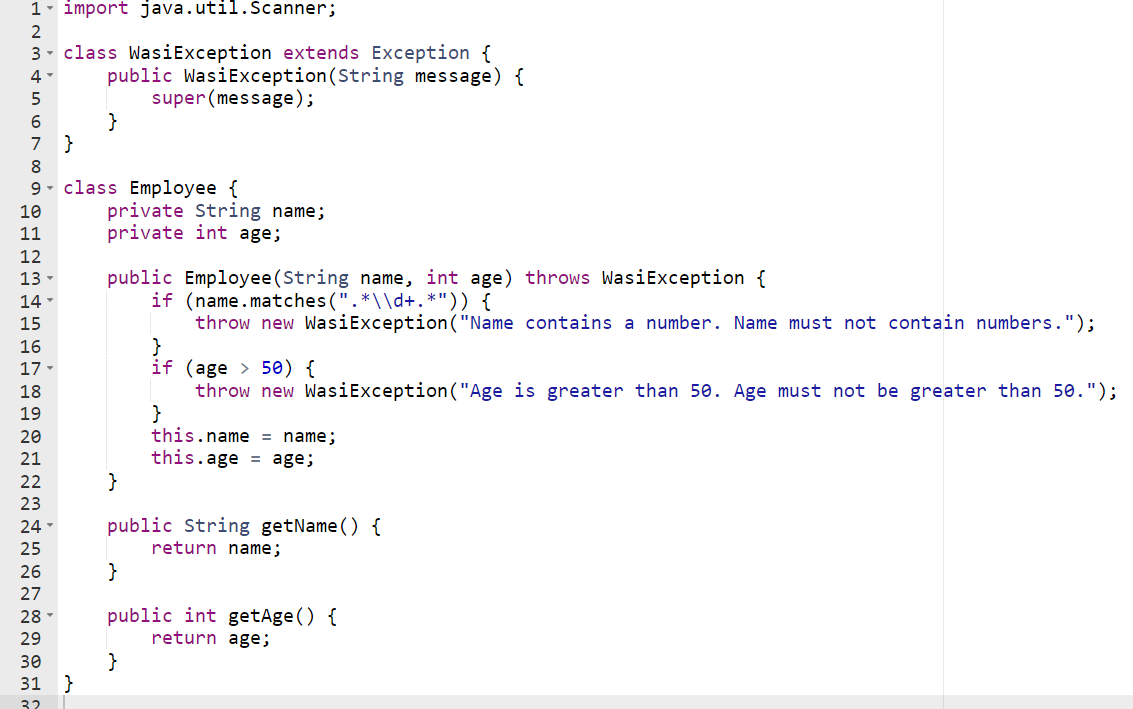
CollisionException is a custom exception class.We assume that initially, areVehiclesRunningInSameDirection is true to avoid collisions.If the vehicles are running in different directions, a CollisionException is thrown.

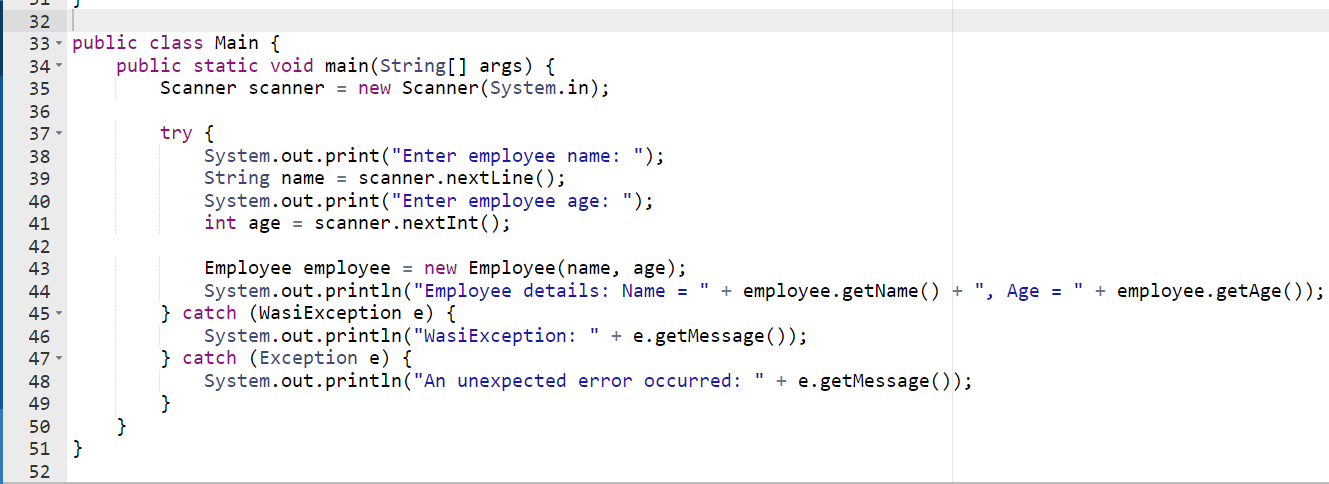


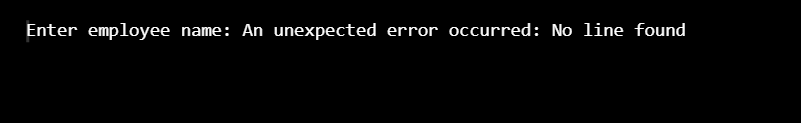


This is because we assumed that areVehiclesRunningInSameDirection is true, indicating that the vehicles are initially running in the same direction, and therefore, there is no risk of collision.

1. Write a java program to throw an exception for an employee details.
2. If an employee name is a number, a name exception must be thrown.
3. If an employee age is greater than 50, an age exception must be thrown.
4. Or else an object must be created for the entered employee details



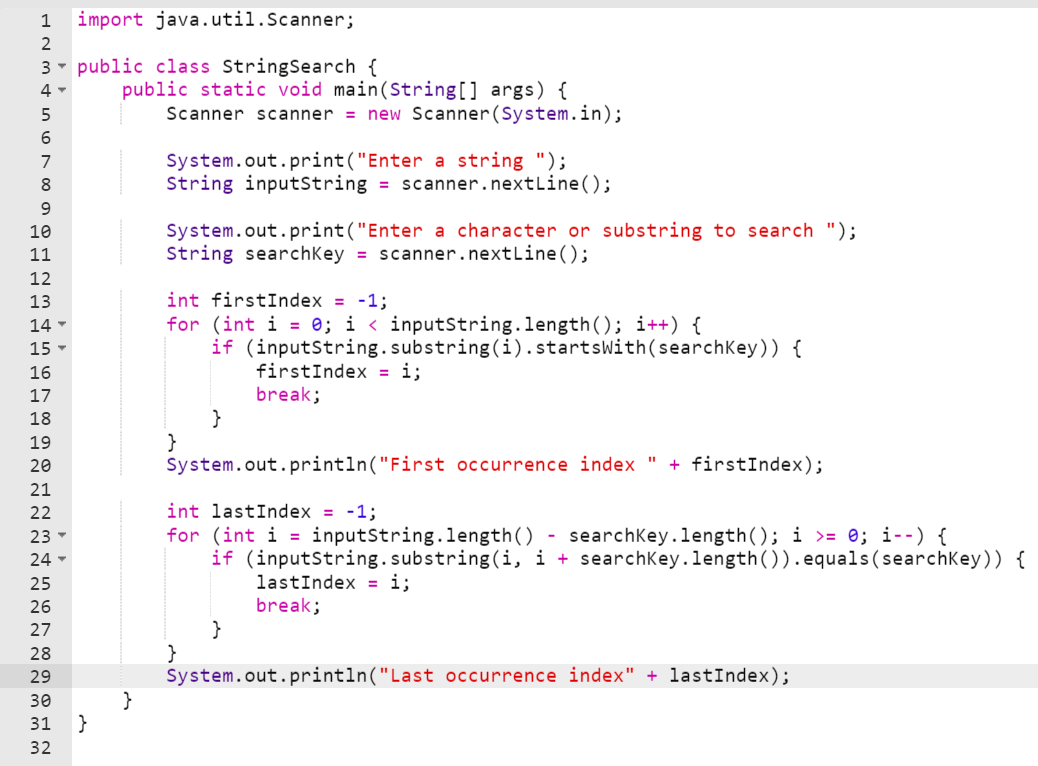


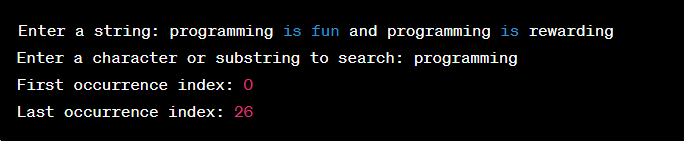


**Experiment**

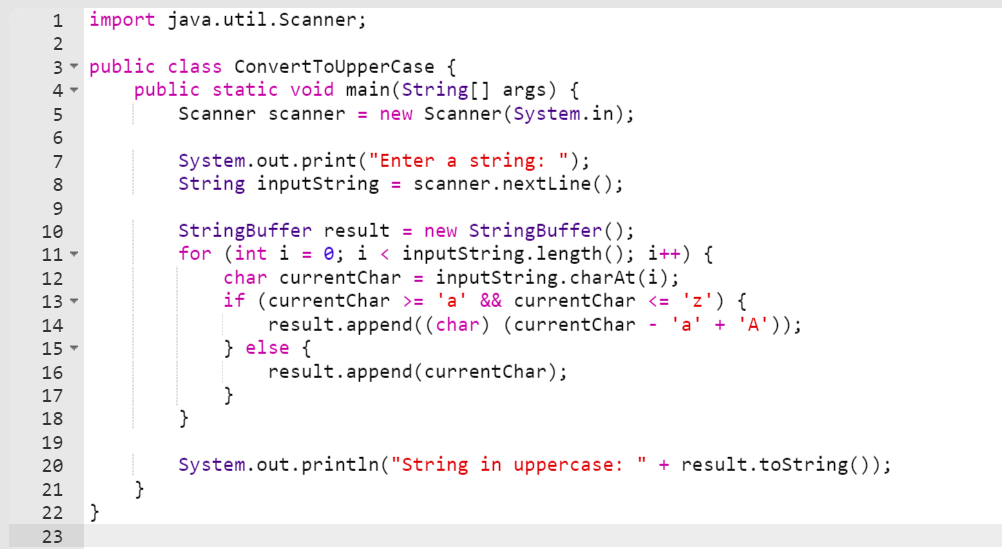
**Strings Handling and Wrapper Class**

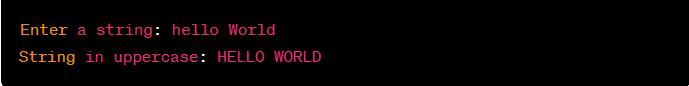
1. Write a program for searching strings for the first occurrence of a character or substring and for the last occurrence of a character or substring.



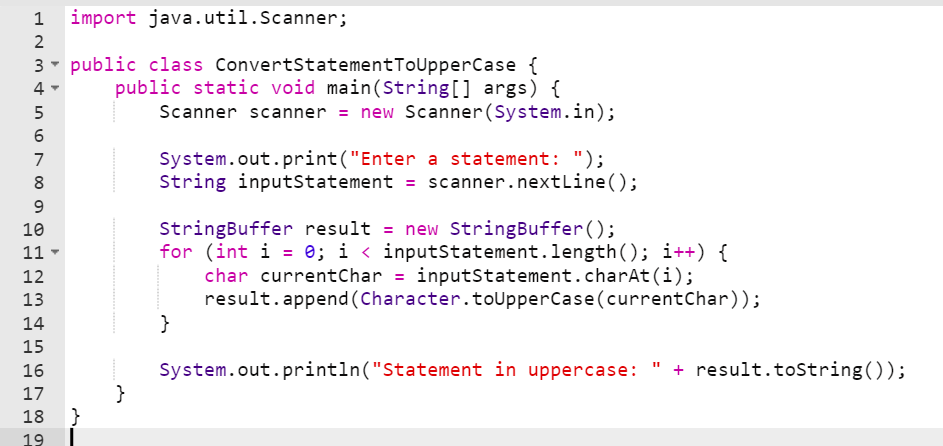


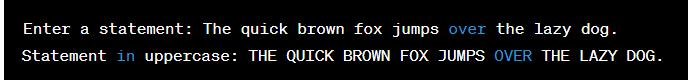
1. Write a program that converts all characters of a string in capital letters. (Use StringBuffer to store a string). Don’t use inbuilt function.



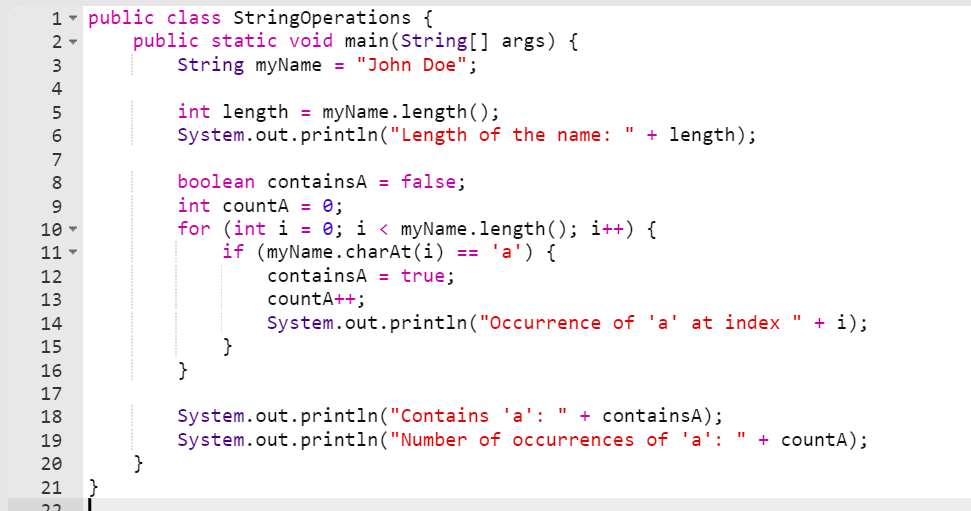


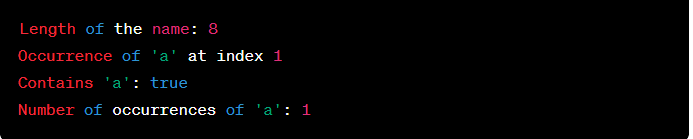
1. Write a program in Java to read a statement from console, convert it into upper case and again print on console. (Don’t use inbuilt function)





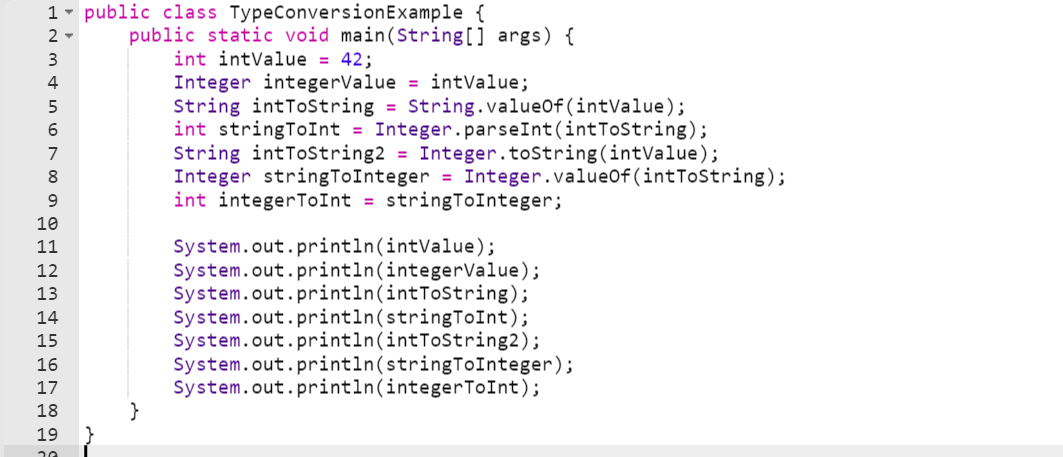
1. Write a program in Java to create a String object. Initialize this object with your name. Find the length of your name using the appropriate String method. Find whether the character ‘a’ is in your name or not; if yes find the number of times ‘a’ appears in your name. Print locations of occurrences of ‘a’ .Try the same for different String objects

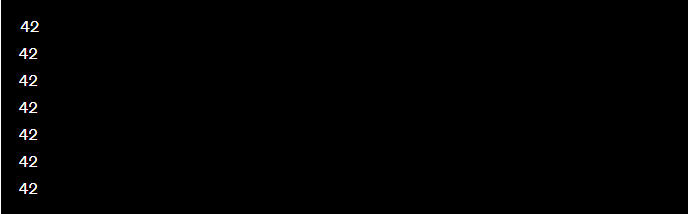




**TITLE:** Wrapper Classes

1. Write a Java code that converts int to Integer, converts Integer to String, converts String to int, converts int to String, converts String to Integer converts Integer to int.





1. Write a Java code that converts float to Float converts Float to String converts String to float converts float to String converts String to Float converts Float to float.

