

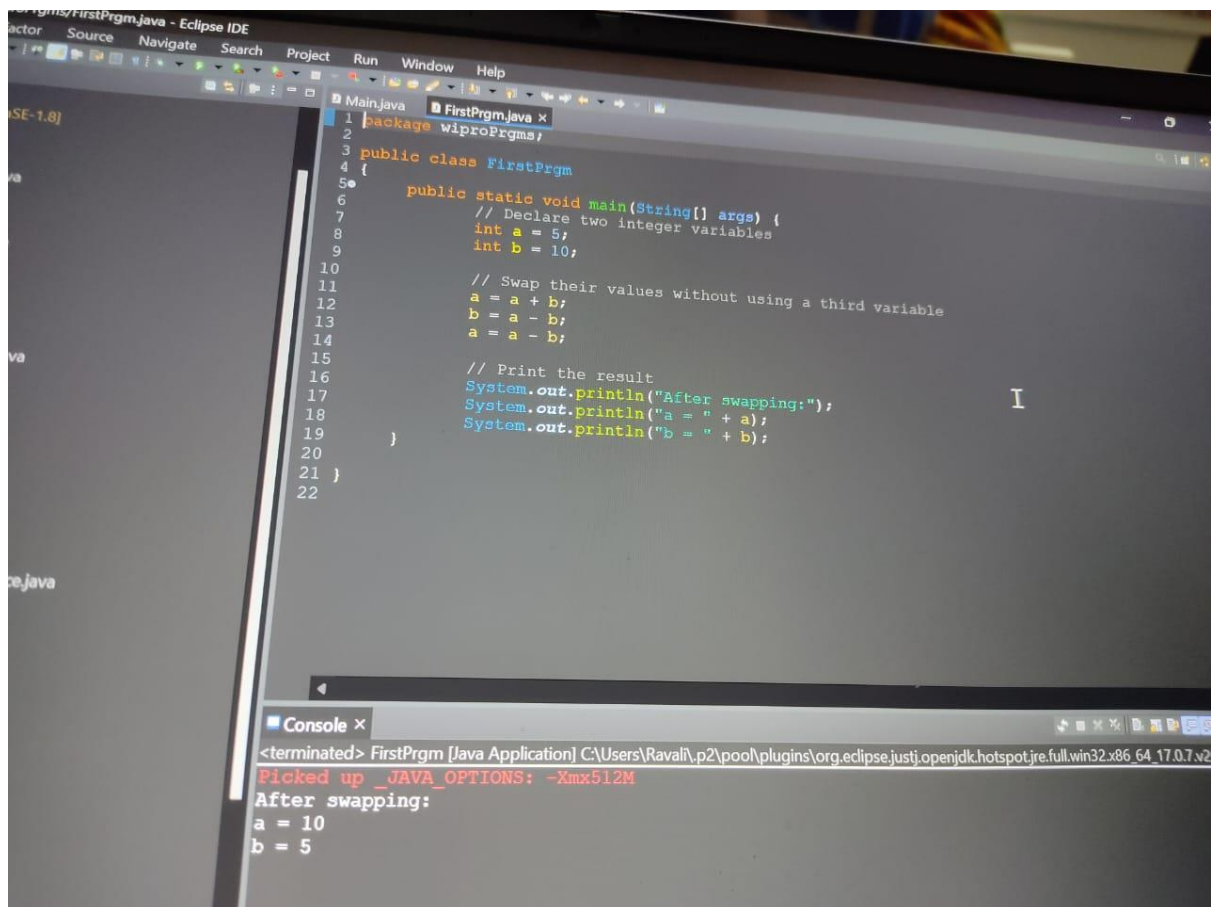
NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com

Day 1and 2

Task 1: Data Types/Variables

Write a program that declares two integer variables, swaps their values without using a third variable, and prints the result.



```
1 package wiproPrgrms;
2
3 public class FirstPrgm
4 {
5     public static void main(String[] args) {
6         // Declare two integer variables
7         int a = 5;
8         int b = 10;
9
10        // Swap their values without using a third variable
11        a = a + b;
12        b = a - b;
13        a = a - b;
14
15        // Print the result
16        System.out.println("After swapping:");
17        System.out.println("a = " + a);
18        System.out.println("b = " + b);
19    }
20 }
21
22
```

Console Output:

```
<terminated> FirstPrgm [Java Application] C:\Users\Ravali\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.7.v2
Picked up _JAVA_OPTIONS: -Xmx512M
After swapping:
a = 10
b = 5
```

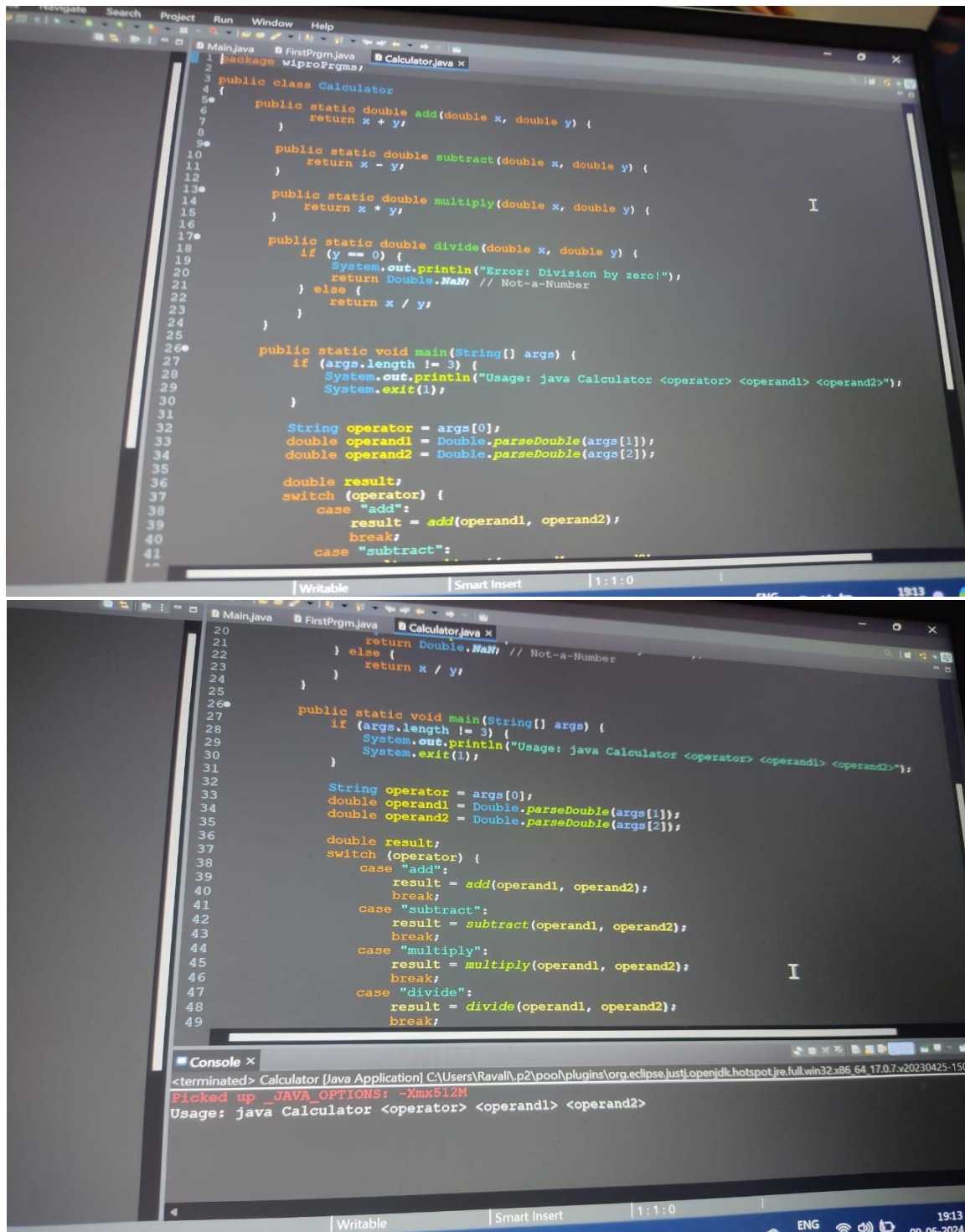
This Java program achieves the same result by swapping the values of a and b without using a third variable.

NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com

Task 2: Operators

Create a program that simulates a simple calculator using command-line arguments to perform and print the result of addition, subtraction, multiplication, and division..



```
1 package wiproPrjma;
2
3 public class Calculator
4 {
5     public static double add(double x, double y) {
6         return x + y;
7     }
8
9     public static double subtract(double x, double y) {
10        return x - y;
11    }
12
13    public static double multiply(double x, double y) {
14        return x * y;
15    }
16
17    public static double divide(double x, double y) {
18        if (y == 0) {
19            System.out.println("Error: Division by zero!");
20            return Double.NaN; // Not-a-Number
21        } else {
22            return x / y;
23        }
24    }
25
26    public static void main(String[] args) {
27        if (args.length != 3) {
28            System.out.println("Usage: java Calculator <operator> <operand1> <operand2>");
29            System.exit(1);
30        }
31
32        String operator = args[0];
33        double operand1 = Double.parseDouble(args[1]);
34        double operand2 = Double.parseDouble(args[2]);
35
36        double result;
37        switch (operator) {
38            case "add":
39                result = add(operand1, operand2);
40                break;
41            case "subtract":
42                result = subtract(operand1, operand2);
43                break;
44            case "multiply":
45                result = multiply(operand1, operand2);
46                break;
47            case "divide":
48                result = divide(operand1, operand2);
49                break;
50        }
51        System.out.println("Result: " + result);
52    }
53 }
```

Console Output:

```
<terminated> Calculator [Java Application] C:\Users\Ravali\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.7.v20230425-1502
Picked up _JAVA_OPTIONS: -Xmx512M
Usage: java Calculator <operator> <operand1> <operand2>
```

NAME :- Jangili Raval

EMAIL:- jangiliravali9@gmail.com

To compile and run this Java program, save it to a file named Calculator.java, then compile it using javac:

Javac Calculator.java

And run it using java with the desired operation and operands, like this:

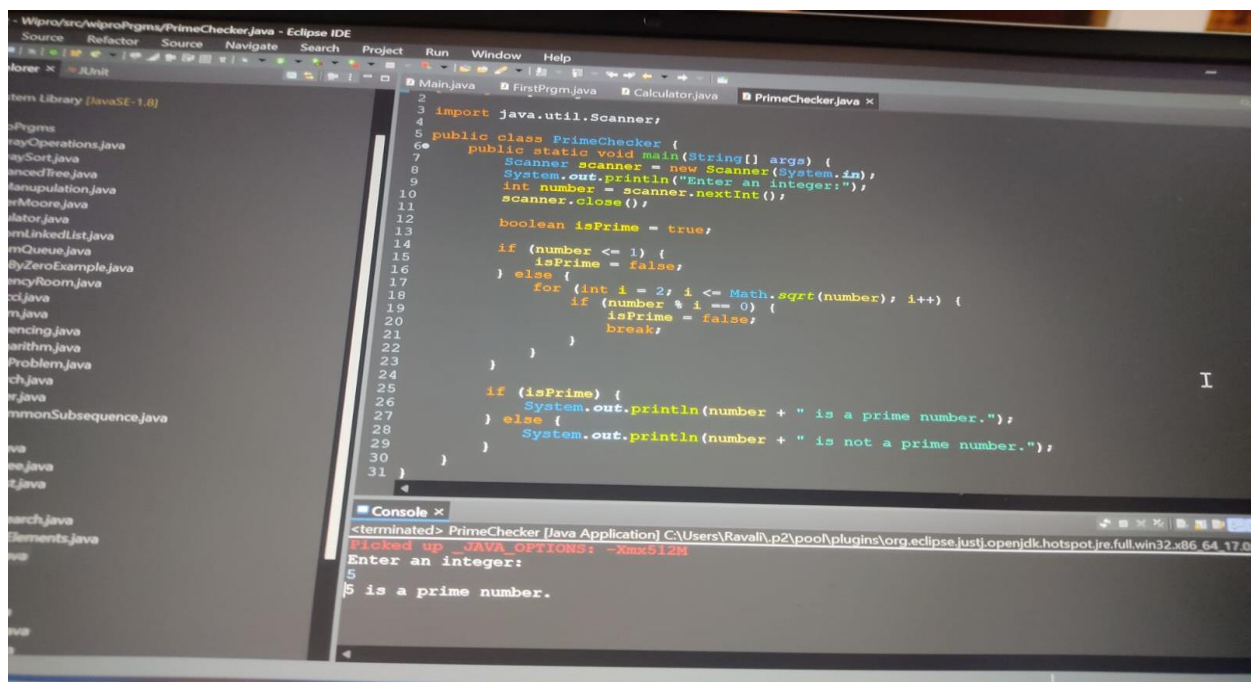
Java Calculator add 5 3

Output

Result: 8.0

Task 3: Control Flow

Write a Java program that reads an integer and prints whether it is a prime number using a for loop and if statements.



```
Wipro\src\wipro\Prims\PrimeChecker.java - Eclipse IDE
Source Refactor Source Navigate Search Project Run Window Help
Main.java FirstPrgm.java Calculator.java PrimeChecker.java x
1 import java.util.Scanner;
2
3 public class PrimeChecker {
4     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6         System.out.println("Enter an integer:");
7         int number = scanner.nextInt();
8         scanner.close();
9
10        boolean isPrime = true;
11
12        if (number <= 1) {
13            isPrime = false;
14        } else {
15            for (int i = 2; i <= Math.sqrt(number); i++) {
16                if (number % i == 0) {
17                    isPrime = false;
18                    break;
19                }
20            }
21        }
22
23        if (isPrime) {
24            System.out.println(number + " is a prime number.");
25        } else {
26            System.out.println(number + " is not a prime number.");
27        }
28    }
29 }
30
31 }

Console x
<terminated> PrimeChecker [Java Application] C:\Users\Ravali\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0
Picked up _JAVA_OPTIONS: -Xmx512M
Enter an integer:
5
5 is a prime number.
```

NAME :- Jangili Raval

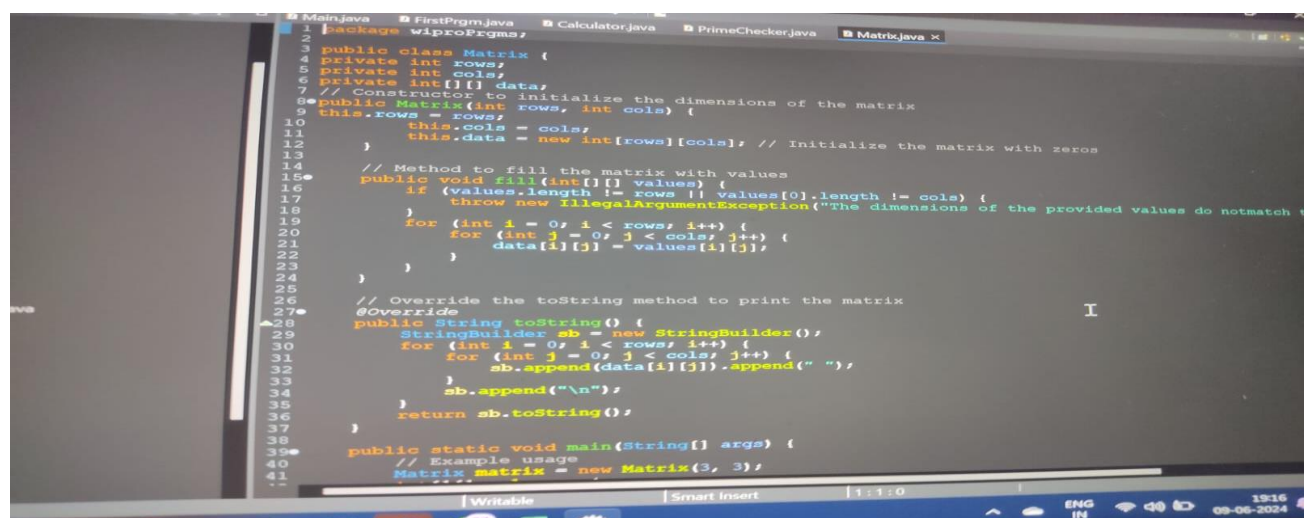
EMAIL:- jangiliravali9@gmail.com

How the program works:

1. It prompts the user to enter an integer.
2. It reads the integer from the user.
3. It checks if the entered number is less than or equal to 1. If it is, the number is not prime.
4. If the number is greater than 1, it iterates from 2 to the square root of the entered number.
5. Within the loop, it checks if the entered number is divisible by any number between 2 and the square root of the entered number.
6. If the entered number is divisible by any of these numbers, it sets the isPrime flag to false and breaks out of the loop.
7. After the loop, it checks the value of isPrime. If it's true, the entered number is prime; otherwise, it's not prime.
8. Finally, it prints the result accordingly

Task 4: Constructors

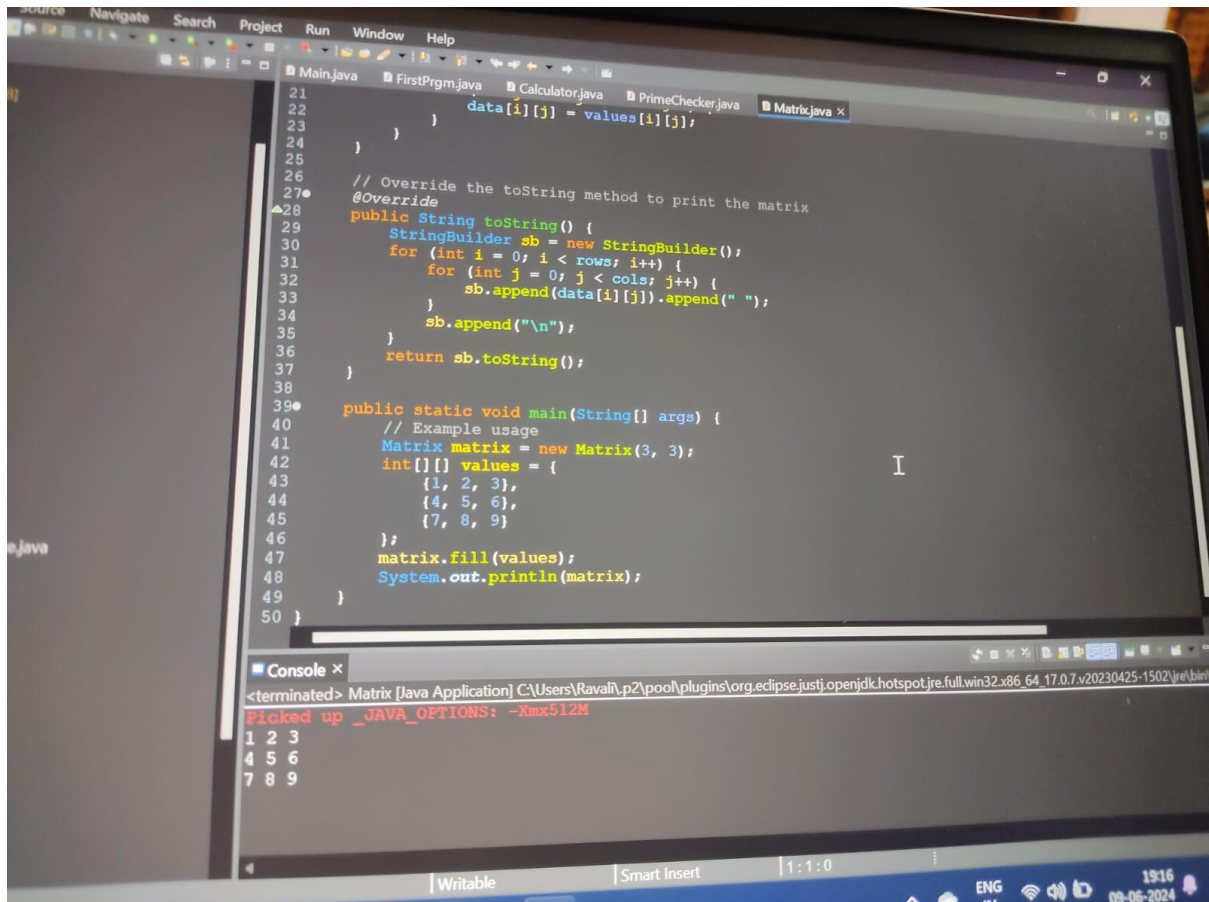
Implement a Matrix class that has a constructor which initializes the dimensions of a matrix and a method to fill the matrix with values.



```
1 package wiproPrjms;
2
3 public class Matrix {
4     private int rows;
5     private int cols;
6     private int[][] data;
7     // Constructor to initialize the dimensions of the matrix
8     public Matrix(int rows, int cols) {
9         this.rows = rows;
10        this.cols = cols;
11        this.data = new int[rows][cols]; // Initialize the matrix with zeros
12    }
13
14    // Method to fill the matrix with values
15    public void fill(int[][] values) {
16        if (values.length != rows || values[0].length != cols) {
17            throw new IllegalArgumentException("The dimensions of the provided values do not match");
18        }
19        for (int i = 0; i < rows; i++) {
20            for (int j = 0; j < cols; j++) {
21                data[i][j] = values[i][j];
22            }
23        }
24    }
25
26    // Override the toString method to print the matrix
27    @Override
28    public String toString() {
29        StringBuilder sb = new StringBuilder();
30        for (int i = 0; i < rows; i++) {
31            for (int j = 0; j < cols; j++) {
32                sb.append(data[i][j]).append(" ");
33            }
34            sb.append("\n");
35        }
36        return sb.toString();
37    }
38
39    public static void main(String[] args) {
40        // Example usage
41        Matrix matrix = new Matrix(3, 3);
42    }
43 }
```

NAME :- Jangili Raval

EMAIL:- jangiliravali9@gmail.com



```
21         data[i][j] = values[i][j];
22     }
23 }
24
25
26 // Override the toString method to print the matrix
27 @Override
28 public String toString() {
29     StringBuilder sb = new StringBuilder();
30     for (int i = 0; i < rows; i++) {
31         for (int j = 0; j < cols; j++) {
32             sb.append(data[i][j]).append(" ");
33         }
34         sb.append("\n");
35     }
36     return sb.toString();
37 }
38
39 public static void main(String[] args) {
40     // Example usage
41     Matrix matrix = new Matrix(3, 3);
42     int[][] values = {
43         {1, 2, 3},
44         {4, 5, 6},
45         {7, 8, 9}
46     };
47     matrix.fill(values);
48     System.out.println(matrix);
49 }
50 }
```

Console x

```
<terminated> Matrix [Java Application] C:\Users\Ravali\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.7.v20230425-1502\jre\bin\
Picked up _JAVA_OPTIONS: -Xmx512M
1 2 3
4 5 6
7 8 9
```

Explanation:

1. Constructor (__init__ method):

- The constructor takes two parameters, rows and cols, to define the dimensions of the matrix.
- It initializes the matrix with zeros using a list comprehension.

2. Fill Method (fill method):

- The fill method takes a list of lists (values) as input.
- It first checks if the dimensions of the provided values match the matrix dimensions. If not, it raises a ValueError.
- If the dimensions match, it assigns the provided values to the matrix.

3. String Representation (__str__ method):

NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com

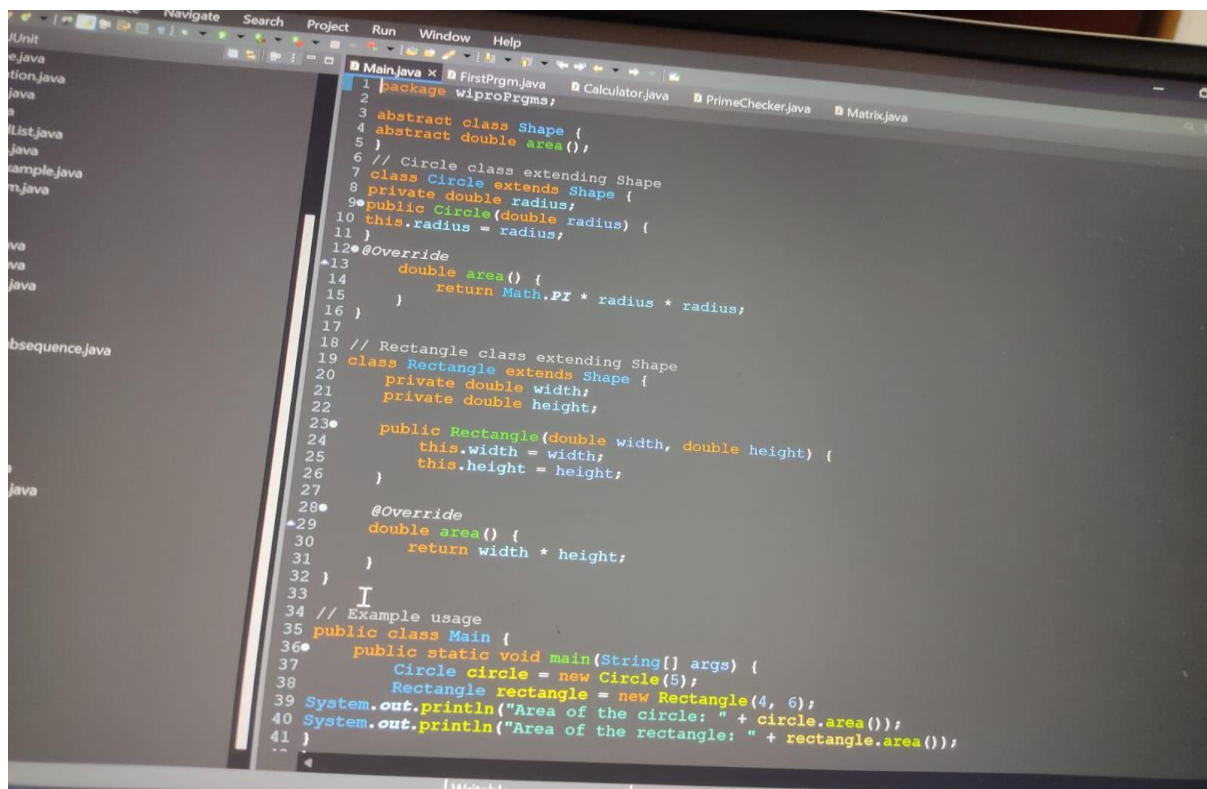
- The `__str__` method returns a string representation of the matrix for easy printing. It joins each row's elements with spaces and each row with newlines.

You can use this Matrix class to create a matrix of any dimensions, fill it with values, and print it out. The fill method ensures that the input values match the matrix's dimensions, raising an error if they do not.

Task 5: Inheritance

Create a Shape class with a method `area()` and extend it with Circle and Rectangle classes overriding the `area()` method appropriately.

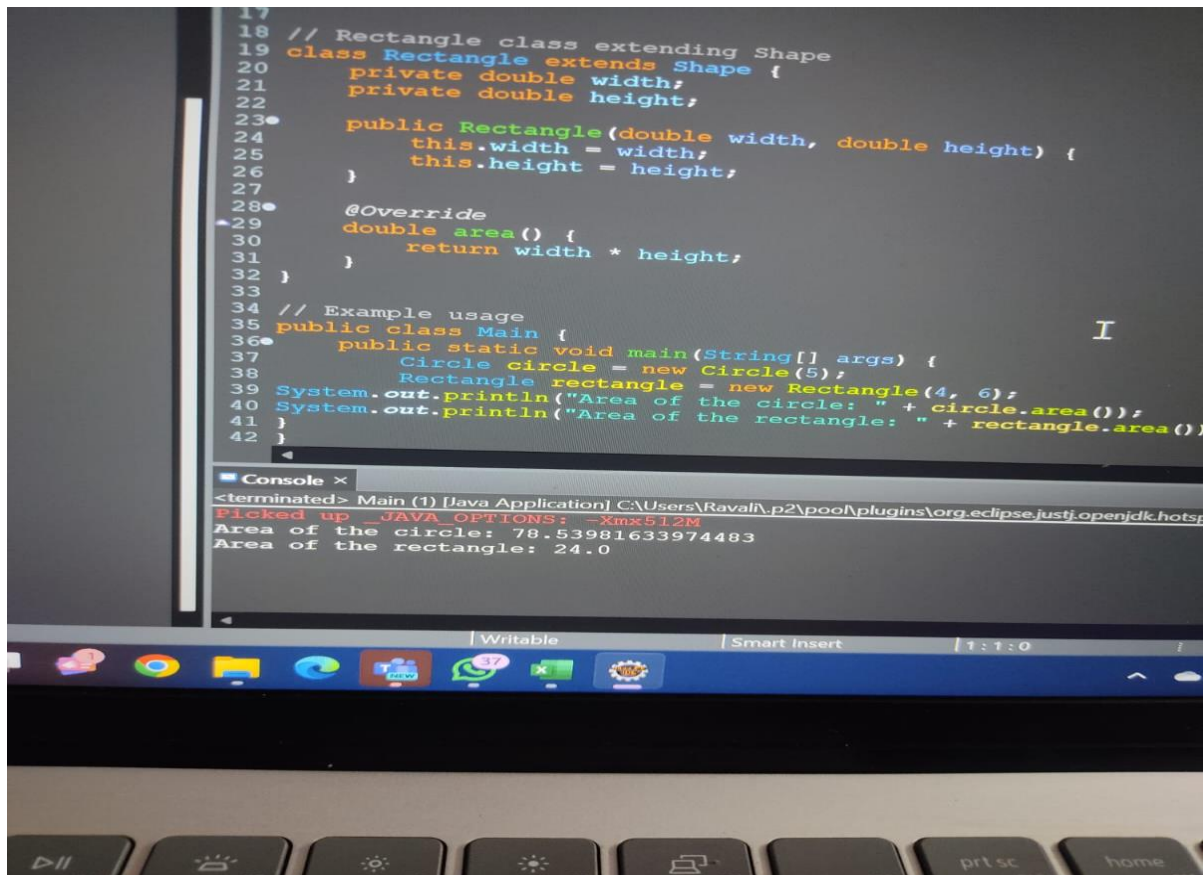
Java implementation of the task where we create a Shape class with an `area()` method and then extend it with Circle and Rectangle classes, each overriding the `area()` method appropriately.



```
1 package wiproPrgrms;
2
3 abstract class Shape {
4     abstract double area();
5 }
6 // Circle class extending Shape
7 class Circle extends Shape {
8     private double radius;
9     public Circle(double radius) {
10         this.radius = radius;
11     }
12     @Override
13     double area() {
14         return Math.PI * radius * radius;
15     }
16 }
17
18 // Rectangle class extending Shape
19 class Rectangle extends Shape {
20     private double width;
21     private double height;
22
23     public Rectangle(double width, double height) {
24         this.width = width;
25         this.height = height;
26     }
27
28     @Override
29     double area() {
30         return width * height;
31     }
32 }
33
34 // Example usage
35 public class Main {
36     public static void main(String[] args) {
37         Circle circle = new Circle(5);
38         Rectangle rectangle = new Rectangle(4, 6);
39         System.out.println("Area of the circle: " + circle.area());
40         System.out.println("Area of the rectangle: " + rectangle.area());
41     }
42 }
```

NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com



```
17
18 // Rectangle class extending Shape
19 class Rectangle extends Shape {
20     private double width;
21     private double height;
22
23     public Rectangle(double width, double height) {
24         this.width = width;
25         this.height = height;
26     }
27
28     @Override
29     double area() {
30         return width * height;
31     }
32 }
33
34 // Example usage
35 public class Main {
36     public static void main(String[] args) {
37         Circle circle = new Circle(5);
38         Rectangle rectangle = new Rectangle(4, 6);
39         System.out.println("Area of the circle: " + circle.area());
40         System.out.println("Area of the rectangle: " + rectangle.area());
41     }
42 }
```

Console x

```
<terminated> Main (1) [Java Application] C:\Users\Ravali\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot
Picked up _JAVA_OPTIONS: -Xmx512M
Area of the circle: 78.53981633974483
Area of the rectangle: 24.0
```

In this Java code:

1. Shape is an abstract base class with an abstract area() method.
2. Circle is a class extending Shape and implementing the area() method to calculate the area of a circle.
3. Rectangle is a class extending Shape and implementing the area() method to calculate the area of a rectangle.

NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com

The Main class contains the main method where instances of Circle and Rectangle are created and their areas are printed.

Task 6: Packages/Classpath

Create a package `com.math.operations` and include classes for various arithmetic operations. Demonstrate how to compile and run these using the classpath.

To create a package `com.math.operations` and include classes for various arithmetic operations, you can follow these steps:

Step 1: Create the Package and Classes

1. Create Directory Structure

- Create a directory structure to reflect the package name. For example:

`com/`

`math/`

`operations/`

2. Create Java Classes

- Inside the operations directory, create Java classes for different arithmetic operations.

Here are examples for addition and subtraction: `Addition.java` package `com.math.operations`;

```
public class Addition {  
    public int add(int a, int b) {  
        return a + b;  
    }  
}
```


NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com

```
}
```

Subtraction.java package

com.math.operations;

```
public class Subtraction {
```

```
public int subtract(int a, int b) {
```

```
return a - b;
```

```
}
```

```
}
```

Step 2: Compile the Classes

1. Open a terminal or command prompt and navigate to the parent directory of com (the root directory where com is located)
2. Compile the Java classes using the javac command:

```
javac com/math/operations/Addition.java com/math/operations/Subtraction.java
```

Step 3: Create a Main Class to Test the Operations

1. Create a main class to use the arithmetic operations. For example:

Main.java import com.math.operations.Addition; import

com.math.operations.Subtraction;

```
public class Main {
```

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

```
        Addition add = new Addition();
```

```
        Subtraction sub = new Subtraction();
```

NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com

```
int sum = add.add(5, 3);    int
difference = sub.subtract(5, 3);

    System.out.println("Sum: " + sum);
    System.out.println("Difference: " + difference);
}
}
```

Step 4: Compile and Run the Main Class

1. Compile the Main Class:

```
javac Main.java
```

2. Run the Main Class using the Classpath:

```
java -cp . Main
```

Directory Structure Example:

Your directory structure should look like this after creating the classes:

```
project_root/
├── Main.java
├── com/
│   └── math/
│       └── operations/
│           ├── Addition.java
│           └── Subtraction.java Full
```

Example:

Assuming the following file contents:

NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com

com/math/operations/Addition.java

package com.math.operations;

```
public class Addition {  
    public int add(int a, int b) {  
        return a + b;  
    }  
}
```

com/math/operations/Subtraction.java

package com.math.operations;

```
public class Subtraction {  
    public int subtract(int a, int b) {  
        return a - b;  
    }  
}
```

Main.java import

com.math.operations.Addition; import

com.math.operations.Subtraction;

```
public class Main {  
    public static void main(String[] args) {  
        Addition add = new Addition();  
        Subtraction sub = new Subtraction();
```

NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com

```
int sum = add.add(5, 3);    int
difference = sub.subtract(5, 3);

    System.out.println("Sum: " + sum);
    System.out.println("Difference: " + difference);
}
}
```

Compilation and Execution Commands:

Navigate to the project root directory where Main.java and com/ are located

```
javac com/math/operations/Addition.java com/math/operations/Subtraction.java
```

```
javac Main.java java -cp . Main
```

This should output:

Sum: 8

Difference: 2

By following these steps, you create a package com.math.operations, compile the Java classes, and run the main class using the classpath.

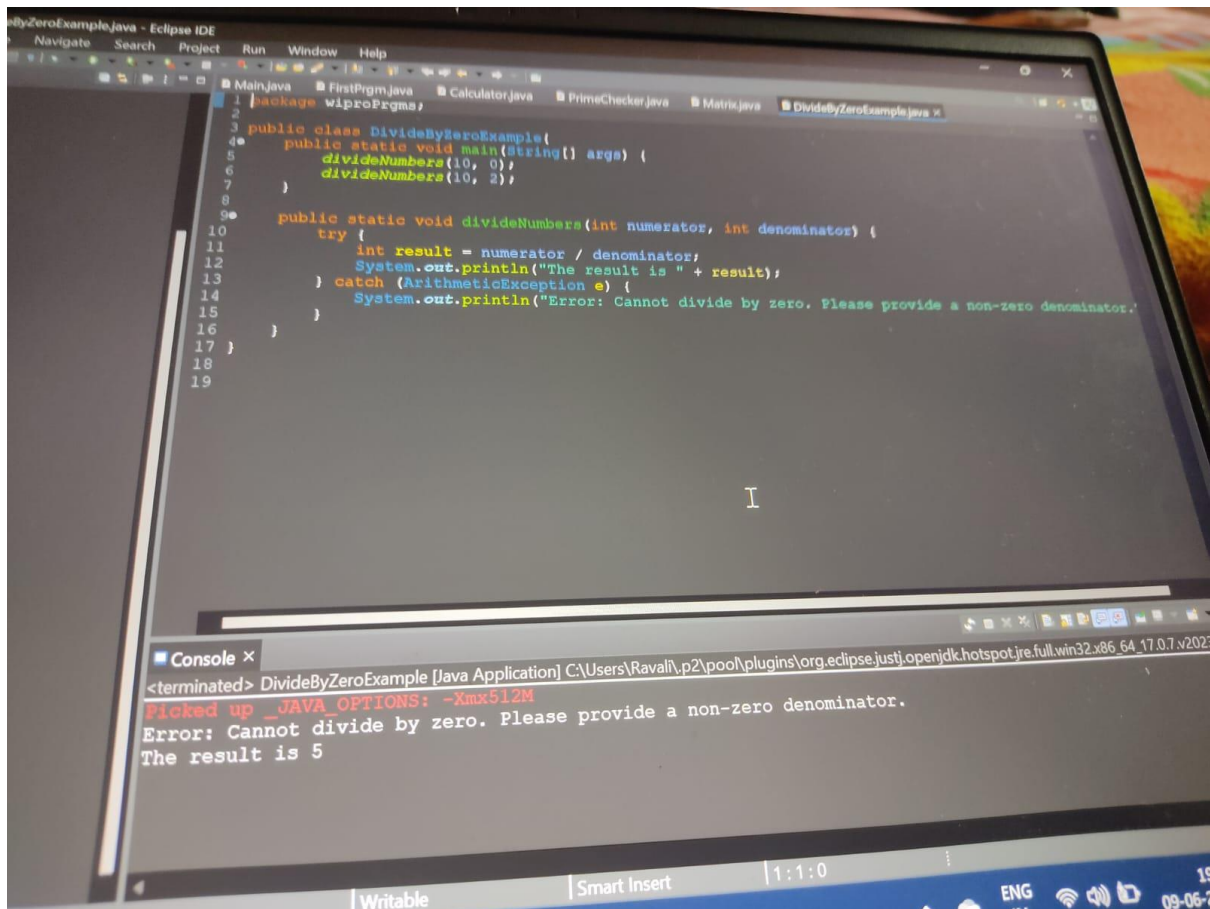
Task 7: Basic Exception Handling

Write a program that attempts to divide by zero, catches the `ArithmeticException`, and provides a custom error message.

Java program that demonstrates basic exception handling for a division by zero error, catching the `ArithmeticException`, and providing a custom error message:

NAME :- Jangili Ravali

EMAIL:- jangiliravali9@gmail.com



```
1 package wiproPrjms;
2
3 public class DivideByZeroExample {
4     public static void main(String[] args) {
5         divideNumbers(10, 0);
6         divideNumbers(10, 2);
7     }
8
9     public static void divideNumbers(int numerator, int denominator) {
10        try {
11            int result = numerator / denominator;
12            System.out.println("The result is " + result);
13        } catch (ArithmeticException e) {
14            System.out.println("Error: Cannot divide by zero. Please provide a non-zero denominator.");
15        }
16    }
17 }
18
19
```

Console x

```
<terminated> DivideByZeroExample [Java Application] C:\Users\Ravali\p2\poo\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.7.v2023-05-01\bin\java.exe
Picked up _JAVA_OPTIONS: -Xmx512M
Error: Cannot divide by zero. Please provide a non-zero denominator.
The result is 5
```

In this program:

- The main method demonstrates the usage of the divideNumbers method with two examples: one with a denominator of zero and one with a valid non-zero denominator.
 - The divideNumbers method attempts to divide the numerator by the denominator inside a try block
 - If an ArithmeticException occurs (which happens in case of division by zero), it catches the exception in the catch block and prints a custom error message.
-