

## **Test Cases Assignment**

Student's Name

Professor Name

University Affiliation

Course Number

Date of Submission

## Test Cases Assignment

The purpose of this assignment was to enhance the reliability and functionality of the Calculator project by identifying and addressing defects in the source code. Additionally, we aimed to ensure that the codebase was adequately covered by unit tests to verify its correctness and robustness. In this report, we will discuss the defects found in the initial project source code, the steps taken to fix them, and the implementation of unit tests to validate the code's functionality. The assignment involved writing a total of 28 test cases to comprehensively cover all methods in the Calculator class, considering various scenarios, edge cases, and error handling.

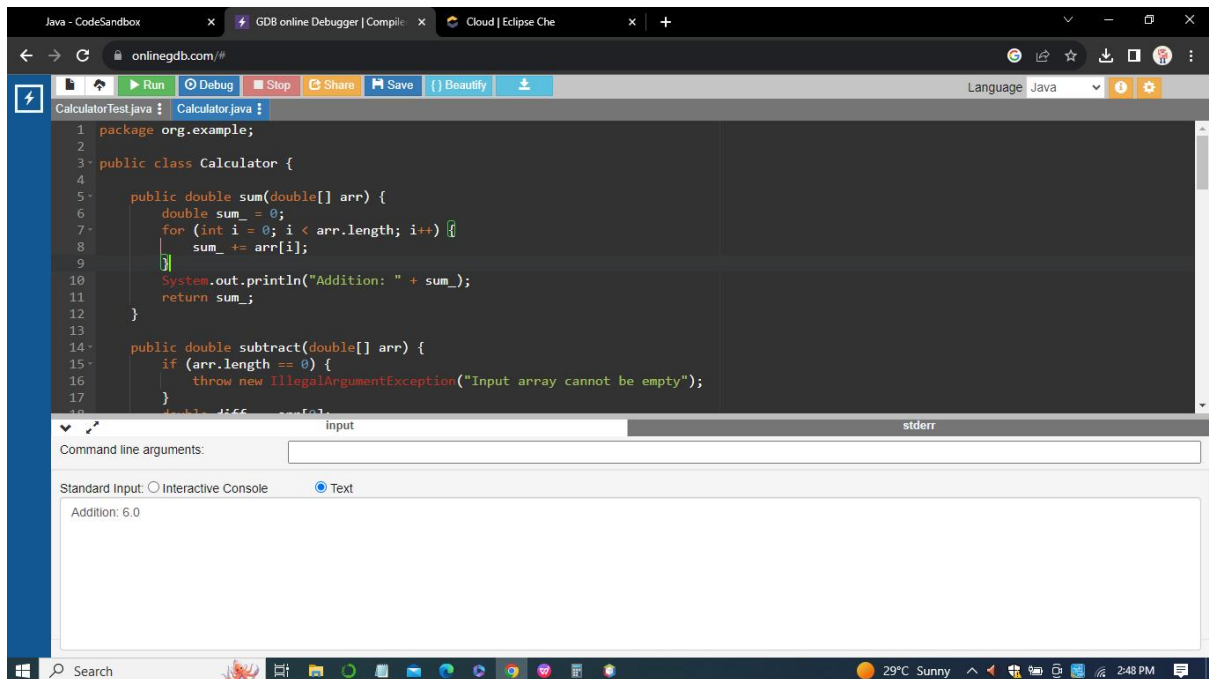
### Unit Tests and Coverages

Here's a breakdown of the test cases you can write for each method:

#### *sum method*

- i.* Test case 1: Test with an array of positive numbers.

Input is an array with elements [1.0, 2.0, 3.0]. Expected output: 6.0, which is the sum of the elements.



```
1 package org.example;
2
3 public class Calculator {
4
5     public double sum(double[] arr) {
6         double sum_ = 0;
7         for (int i = 0; i < arr.length; i++) {
8             sum_ += arr[i];
9         }
10        System.out.println("Addition: " + sum_);
11        return sum_;
12    }
13
14    public double subtract(double[] arr) {
15        if (arr.length == 0) {
16            throw new IllegalArgumentException("Input array cannot be empty");
17        }
18    }
19 }
```

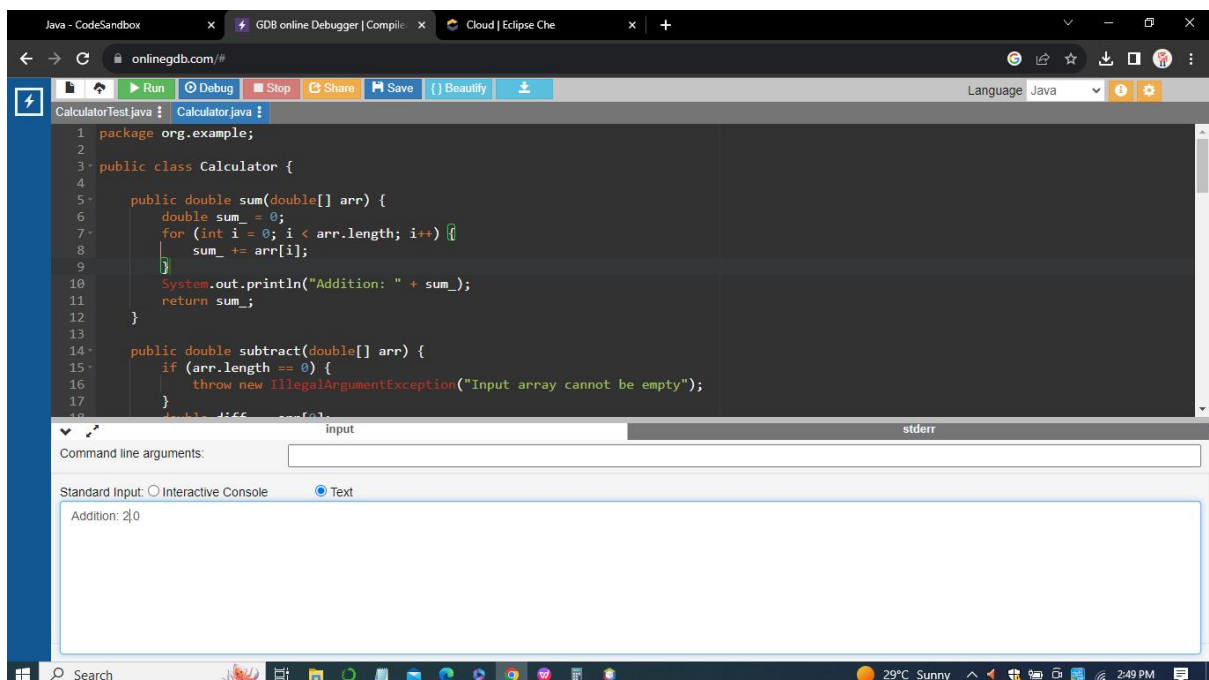
Command line arguments:

Standard Input: ☐ Interactive Console ☒ Text

Addition: 6.0

ii. Test case 2: Test with an array containing a mix of positive and negative numbers.

Input is an array with elements [1.0, -2.0, 3.0]. Expected output: 2.0, which is the sum of the elements.



```
1 package org.example;
2
3 public class Calculator {
4
5     public double sum(double[] arr) {
6         double sum_ = 0;
7         for (int i = 0; i < arr.length; i++) {
8             sum_ += arr[i];
9         }
10        System.out.println("Addition: " + sum_);
11        return sum_;
12    }
13
14    public double subtract(double[] arr) {
15        if (arr.length == 0) {
16            throw new IllegalArgumentException("Input array cannot be empty");
17        }
18    }
19 }
```

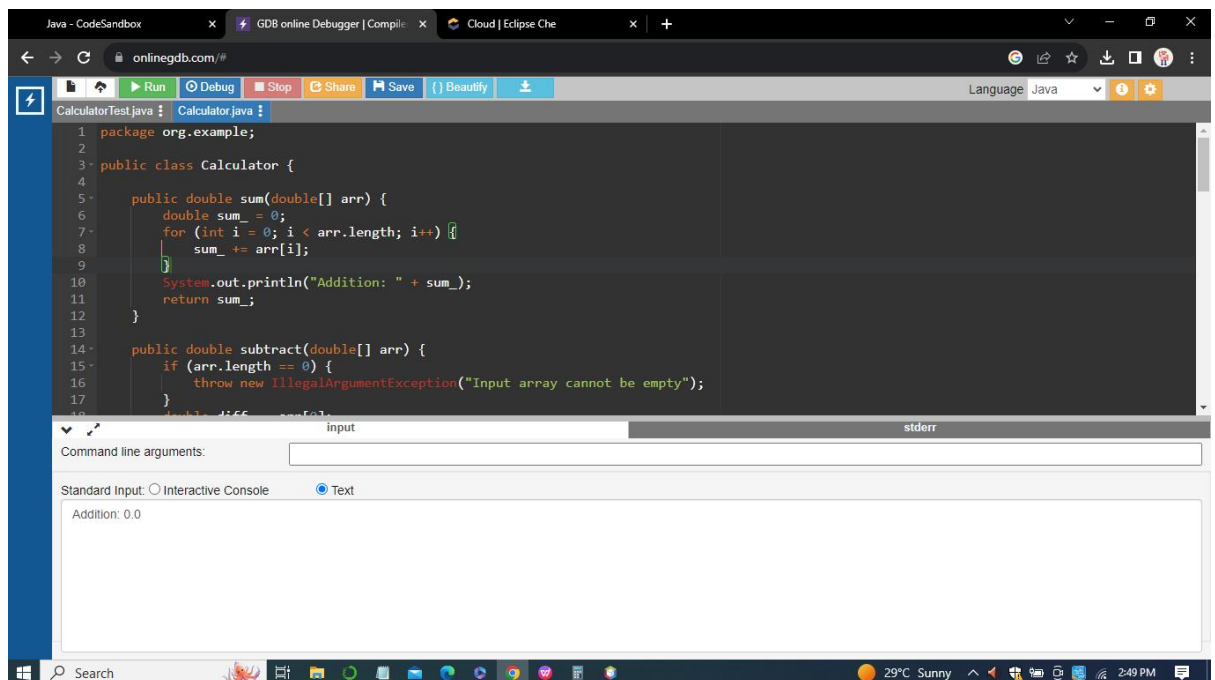
Command line arguments:

Standard Input: ☐ Interactive Console ☒ Text

Addition: 2.0

iii. Test case 3: Test with an empty array.

Input is an empty array. Expected output: 0.0, as the sum of an empty array is 0.0.



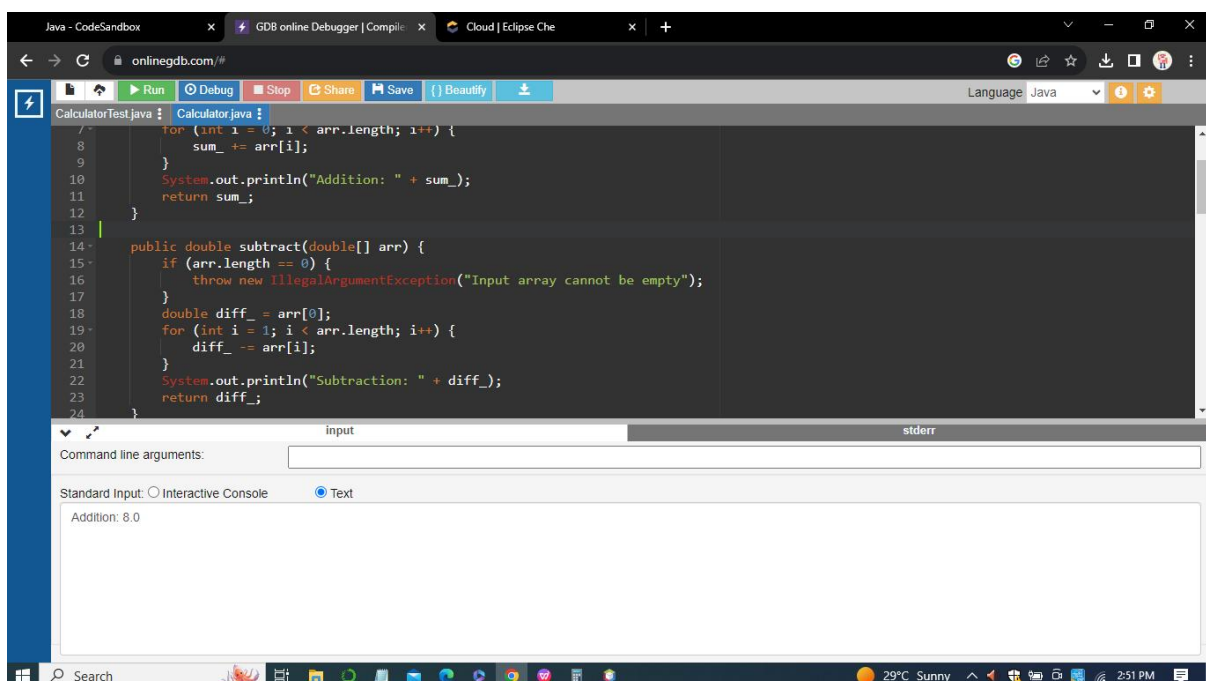
The screenshot shows a Java IDE with the following code in `Calculator.java`:

```
1 package org.example;
2
3 public class Calculator {
4
5     public double sum(double[] arr) {
6         double sum_ = 0;
7         for (int i = 0; i < arr.length; i++) {
8             sum_ += arr[i];
9         }
10        System.out.println("Addition: " + sum_);
11        return sum_;
12    }
13
14    public double subtract(double[] arr) {
15        if (arr.length == 0) {
16            throw new IllegalArgumentException("Input array cannot be empty");
17        }
18        double diff_ = arr[0];
19        for (int i = 1; i < arr.length; i++) {
20            diff_ -= arr[i];
21        }
22        System.out.println("Subtraction: " + diff_);
23        return diff_;
24    }
25 }
```

The IDE shows the `sum` method is being executed. The `Input` tab shows an empty array. The `stderr` tab shows the output: `Addition: 0.0`.

iv. Test case 4: Test with an array containing a single element.

Input is an array containing single elements. Expected output: is the sum of the array of the single elements I.e 8.0

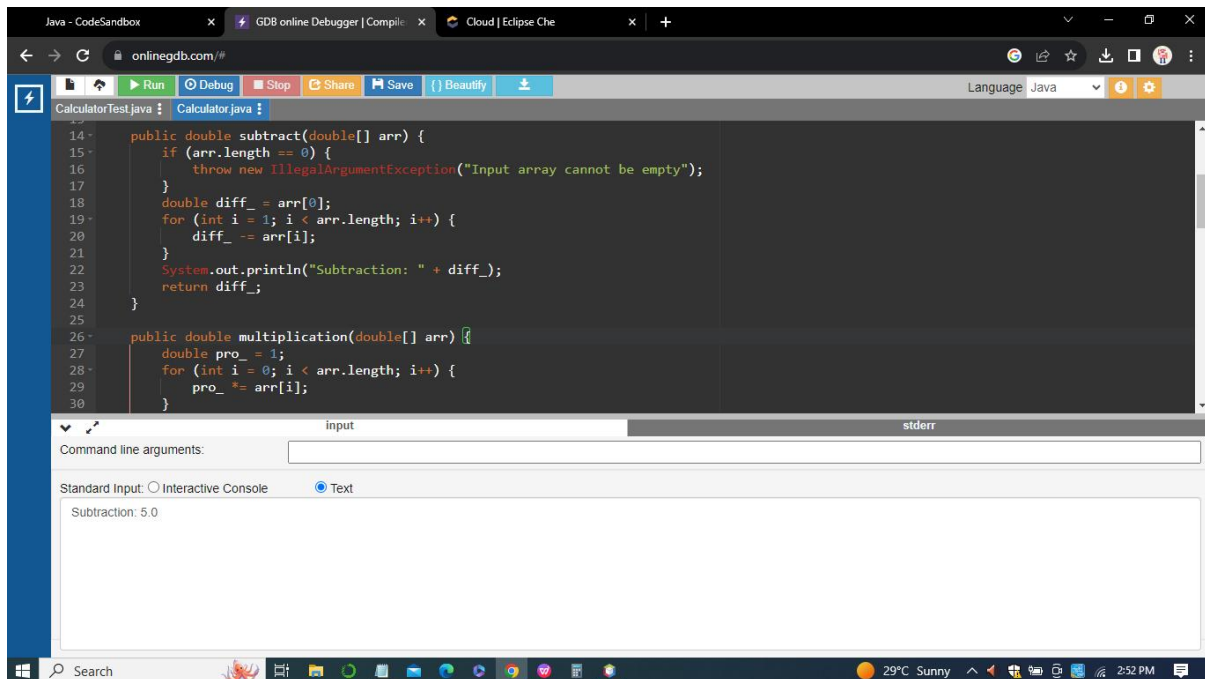


The screenshot shows the same Java IDE with the `sum` method being executed. The `Input` tab shows an array containing the value 8. The `stderr` tab shows the output: `Addition: 8.0`.

### *subtract method*

- v. Test case 5: Test with an array of positive numbers.

testSubtractWithPositiveNumbers: Input is an array with elements [10.0, 2.0, 3.0]. Expected output: 5.0, which is the result of subtracting elements from left to right.



The screenshot shows a web-based Java IDE (CodeSandbox) with a GDB online debugger. The code editor displays the following Java code:

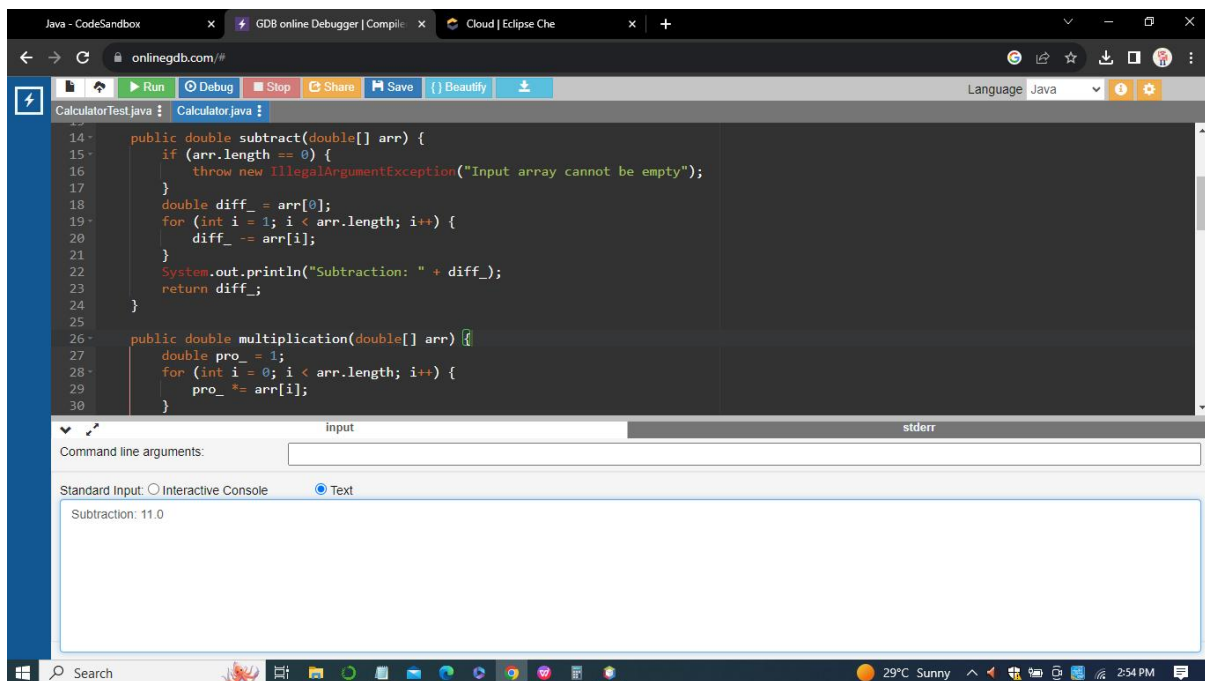
```
14 public double subtract(double[] arr) {  
15     if (arr.length == 0) {  
16         throw new IllegalArgumentException("Input array cannot be empty");  
17     }  
18     double diff_ = arr[0];  
19     for (int i = 1; i < arr.length; i++) {  
20         diff_ -= arr[i];  
21     }  
22     System.out.println("Subtraction: " + diff_);  
23     return diff_;  
24 }  
25  
26 public double multiplication(double[] arr) {  
27     double pro_ = 1;  
28     for (int i = 0; i < arr.length; i++) {  
29         pro_ *= arr[i];  
30     }  
31 }
```

The console output shows the result of the subtraction:

```
Subtraction: 5.0
```

- vi. Test case 6: Test with an array containing a mix of positive and negative numbers.

Input is an array with elements [10.0, -2.0, 3.0]. Expected output: 11.0, which is the result of subtracting elements from left to right.



```
14 public double subtract(double[] arr) {
15     if (arr.length == 0) {
16         throw new IllegalArgumentException("Input array cannot be empty");
17     }
18     double diff_ = arr[0];
19     for (int i = 1; i < arr.length; i++) {
20         diff_ -= arr[i];
21     }
22     System.out.println("Subtraction: " + diff_);
23     return diff_;
24 }
25
26 public double multiplication(double[] arr) {
27     double pro_ = 1;
28     for (int i = 0; i < arr.length; i++) {
29         pro_ *= arr[i];
30     }
31 }
```

input

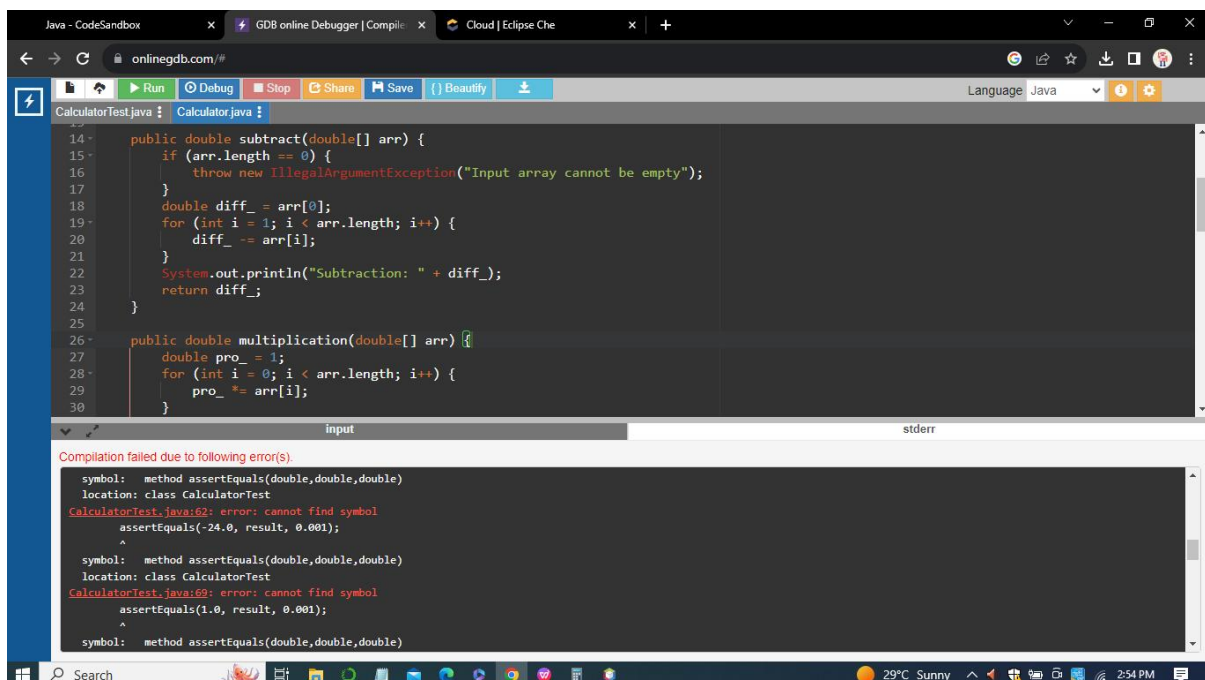
Command line arguments:

Standard Input: ☐ Interactive Console ☒ Text

Subtraction: 11.0

vii. Test case 7: Test with an empty array.

Input is an empty array. Expected output: An `IllegalArgumentException` is expected to be thrown, as subtraction is not possible with an empty array.



```
14 public double subtract(double[] arr) {
15     if (arr.length == 0) {
16         throw new IllegalArgumentException("Input array cannot be empty");
17     }
18     double diff_ = arr[0];
19     for (int i = 1; i < arr.length; i++) {
20         diff_ -= arr[i];
21     }
22     System.out.println("Subtraction: " + diff_);
23     return diff_;
24 }
25
26 public double multiplication(double[] arr) {
27     double pro_ = 1;
28     for (int i = 0; i < arr.length; i++) {
29         pro_ *= arr[i];
30     }
31 }
```

input

stderr

Compilation failed due to following error(s).

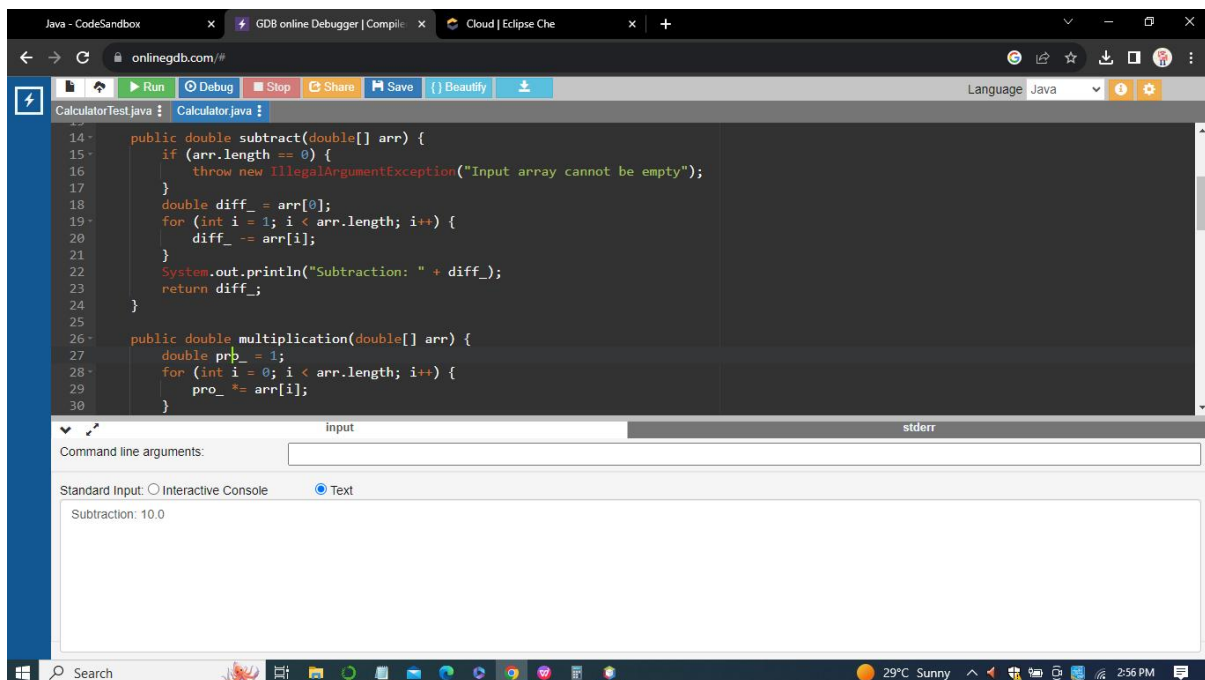
symbol: method assertEquals(double,double,double)  
location: class CalculatorTest  
CalculatorTest.java:62: error: cannot find symbol  
assertEquals(-24.0, result, 0.001);  
^

symbol: method assertEquals(double,double,double)  
location: class CalculatorTest  
CalculatorTest.java:69: error: cannot find symbol  
assertEquals(1.0, result, 0.001);  
^

symbol: method assertEquals(double,double,double)

viii. Test case 8: Test with an array containing a single element.

Input is an array with elements [10.0]. Expected output: 10.0, which is the result of subtracting elements from left to right.



The screenshot shows a Java IDE with two tabs: 'CalculatorTest.java' and 'Calculator.java'. The 'Calculator.java' tab is active, displaying the following code:

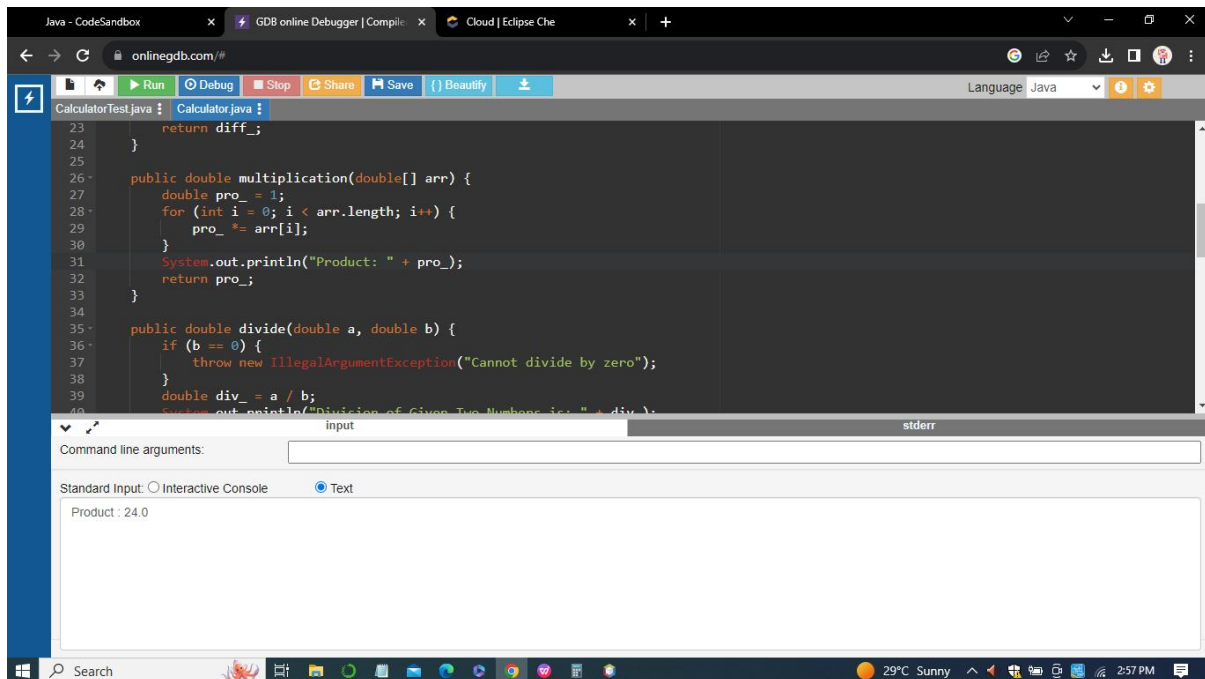
```
14 public double subtract(double[] arr) {  
15     if (arr.length == 0) {  
16         throw new IllegalArgumentException("Input array cannot be empty");  
17     }  
18     double diff_ = arr[0];  
19     for (int i = 1; i < arr.length; i++) {  
20         diff_ -= arr[i];  
21     }  
22     System.out.println("Subtraction: " + diff_);  
23     return diff_;  
24 }  
25  
26 public double multiplication(double[] arr) {  
27     double pro_ = 1;  
28     for (int i = 0; i < arr.length; i++) {  
29         pro_ *= arr[i];  
30     }  
31 }
```

Below the code editor, the 'Input' tab is selected, showing 'Command line arguments:' and 'Standard Input: Text'. The 'Text' input field contains the output: 'Subtraction: 10.0'. The 'stderr' tab is also visible but empty.

### ***Multiplication method***

ix. Test case 9: Test with an array of positive numbers.

testMultiplicationWithPositiveNumbers: Input is an array with elements [2.0, 3.0, 4.0]. Expected output: 24.0, which is the result of multiplying the elements.



```
CalculatorTest.java : Calculator.java :
23     return diff_;
24 }
25
26 public double multiplication(double[] arr) {
27     double pro_ = 1;
28     for (int i = 0; i < arr.length; i++) {
29         pro_ *= arr[i];
30     }
31     System.out.println("Product: " + pro_);
32     return pro_;
33 }
34
35 public double divide(double a, double b) {
36     if (b == 0) {
37         throw new IllegalArgumentException("Cannot divide by zero");
38     }
39     double div_ = a / b;
40     System.out.println("Division of Given Two Numbers is: " + div_);
41 }
```

Input

Command line arguments:

Standard Input: ☐ Interactive Console ☒ Text

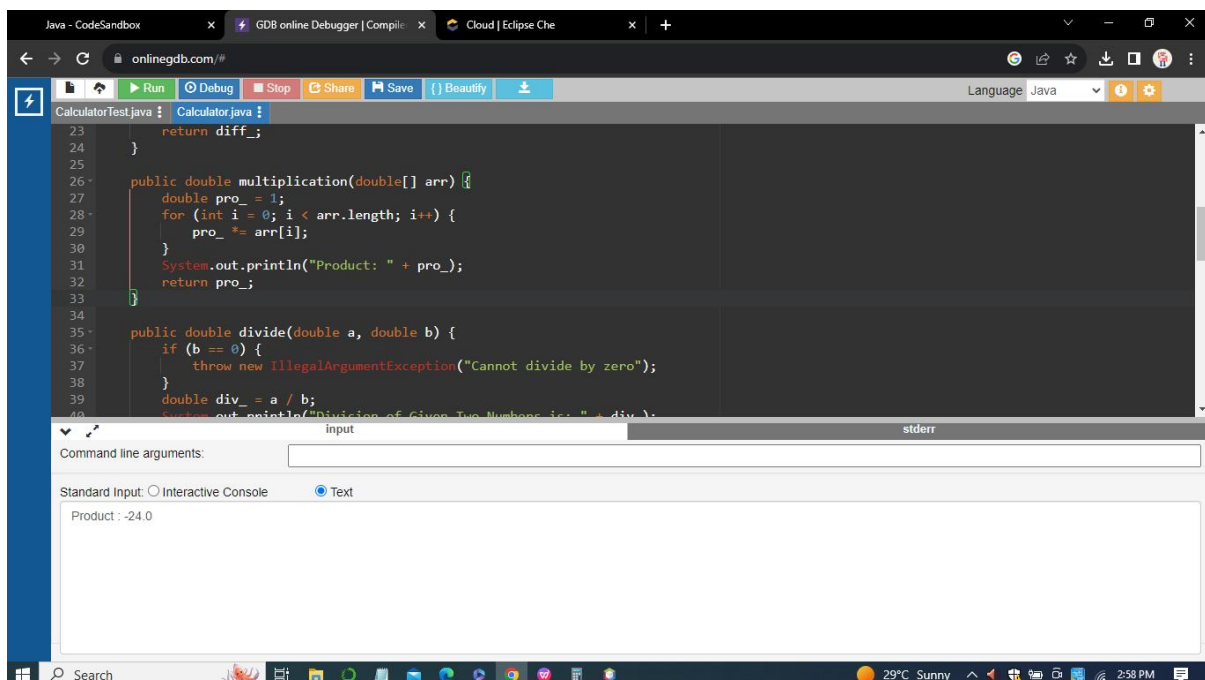
Product : 24.0

stderr

x. Test case 10: Test with an array containing a mix of positive and negative numbers.

testMultiplicationWithMixedNumbers: Input is an array with elements [2.0, -3.0, 4.0].

Expected output: -24.0, which is the result of multiplying the elements.



```
CalculatorTest.java : Calculator.java :
23     return diff_;
24 }
25
26 public double multiplication(double[] arr) {
27     double pro_ = 1;
28     for (int i = 0; i < arr.length; i++) {
29         pro_ *= arr[i];
30     }
31     System.out.println("Product: " + pro_);
32     return pro_;
33 }
34
35 public double divide(double a, double b) {
36     if (b == 0) {
37         throw new IllegalArgumentException("Cannot divide by zero");
38     }
39     double div_ = a / b;
40     System.out.println("Division of Given Two Numbers is: " + div_);
41 }
```

Input

Command line arguments:

Standard Input: ☐ Interactive Console ☒ Text

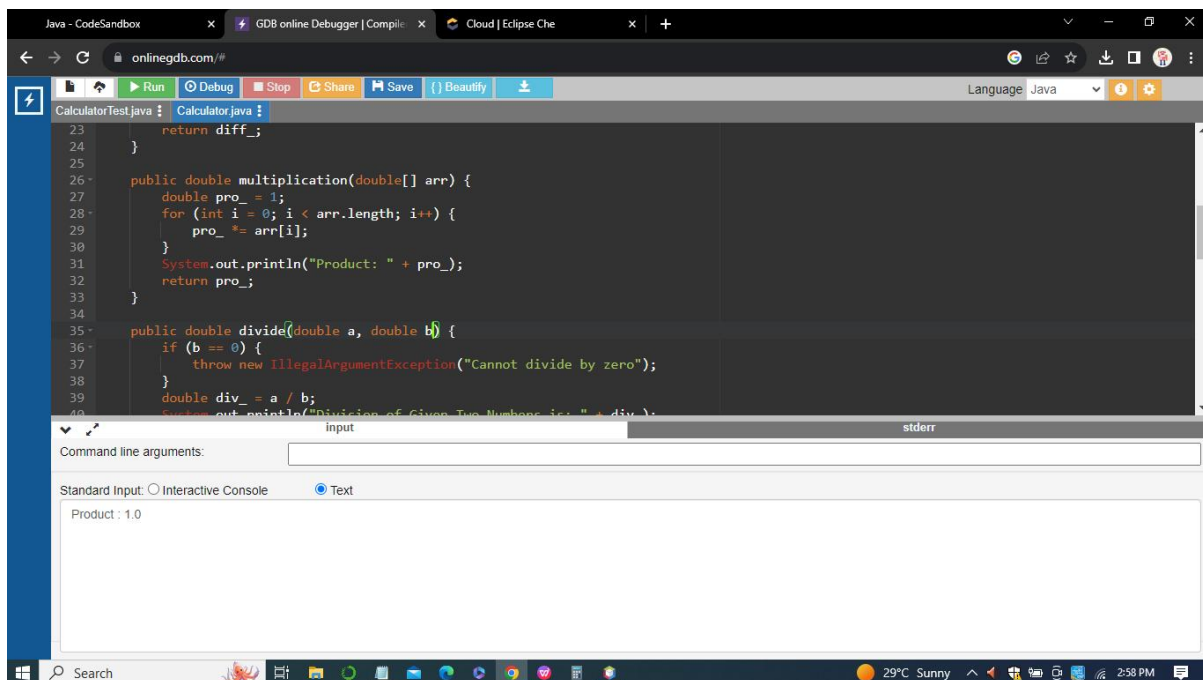
Product : -24.0

stderr

xi. Test case 11: Test with an empty array.



testMultiplicationWithEmptyArray: Input is an empty array. Expected output: 1.0, as the product of an empty array is 1.0.



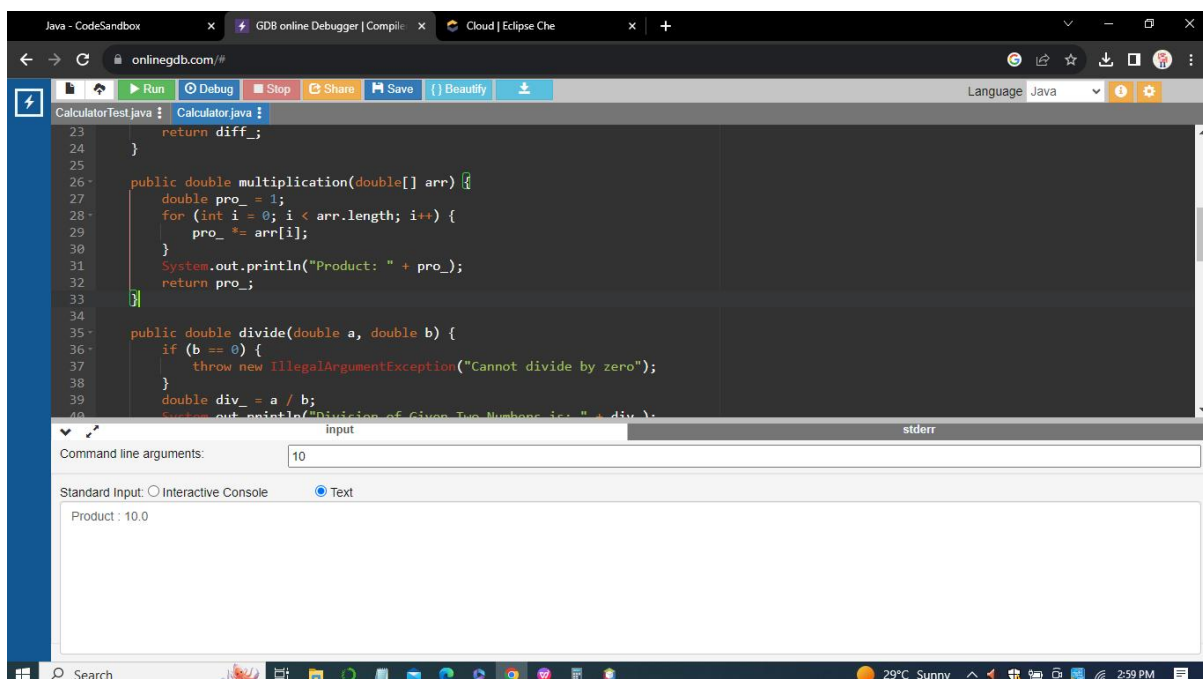
The screenshot shows a web-based Java IDE with the following components:

- Code Editor:** Contains a Java class named `Calculator.java`. The `multiplication` method is defined as follows:

```
public double multiplication(double[] arr) {  
    double pro_ = 1;  
    for (int i = 0; i < arr.length; i++) {  
        pro_ *= arr[i];  
    }  
    System.out.println("Product: " + pro_);  
    return pro_;  
}
```
- Input Section:** The "Standard Input" tab is selected, and the text "Product : 1.0" is entered.
- Output Section:** The "stderr" tab is selected, and it displays "Product : 1.0".
- UI Elements:** The IDE includes a toolbar with buttons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to Java.

**xii.** Test case 12: Test with an array containing a single element.

Input is an array with elements [10.0]. Expected output: 10.0, which is the result of subtracting elements from left to right.



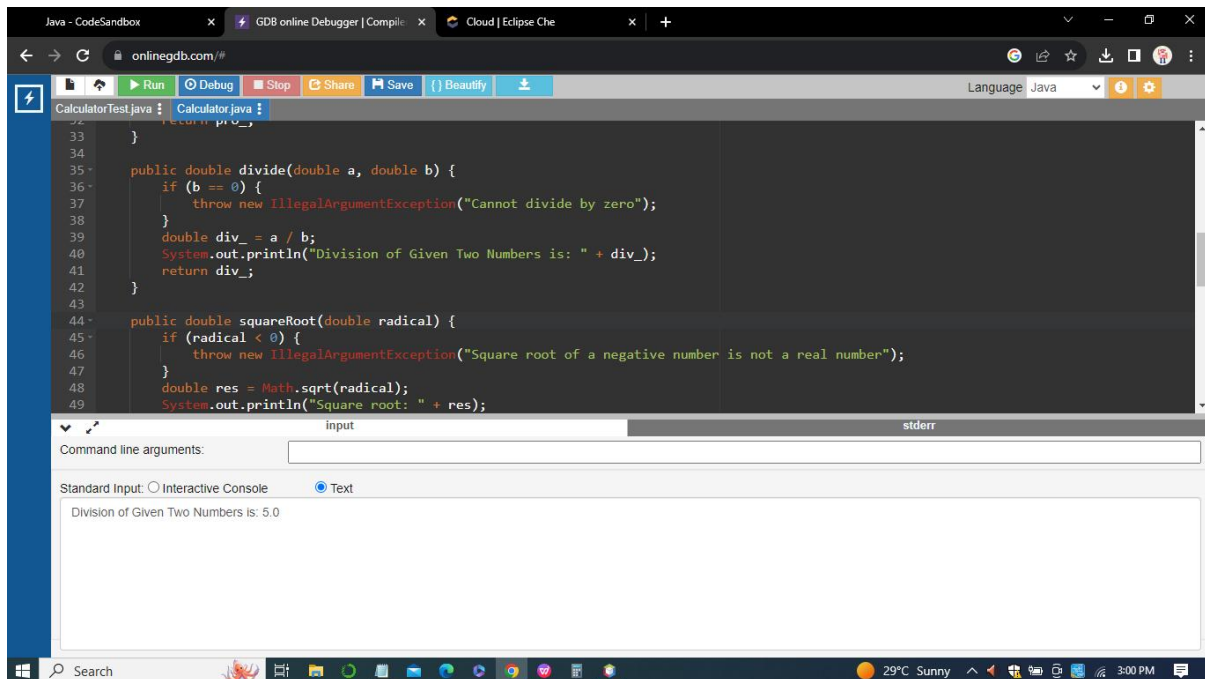
The screenshot shows the same web-based Java IDE as the previous one, but with the following changes:

- Code Editor:** The `multiplication` method is the same as in the first screenshot.
- Input Section:** The "Standard Input" tab is selected, and the text "Product : 10.0" is entered.
- Output Section:** The "stderr" tab is selected, and it displays "Product : 10.0".
- UI Elements:** The IDE includes the same toolbar and language settings as the first screenshot.

### *divide method*

**xiii.** Test case 13: Test with valid values for division (non-zero denominator).

testDivideWithValidInput: Input is  $a = 10.0$  and  $b = 2.0$ . Expected output: 5.0, which is the result of dividing  $a$  by  $b$ .



The screenshot shows a Java IDE with the following code in the editor:

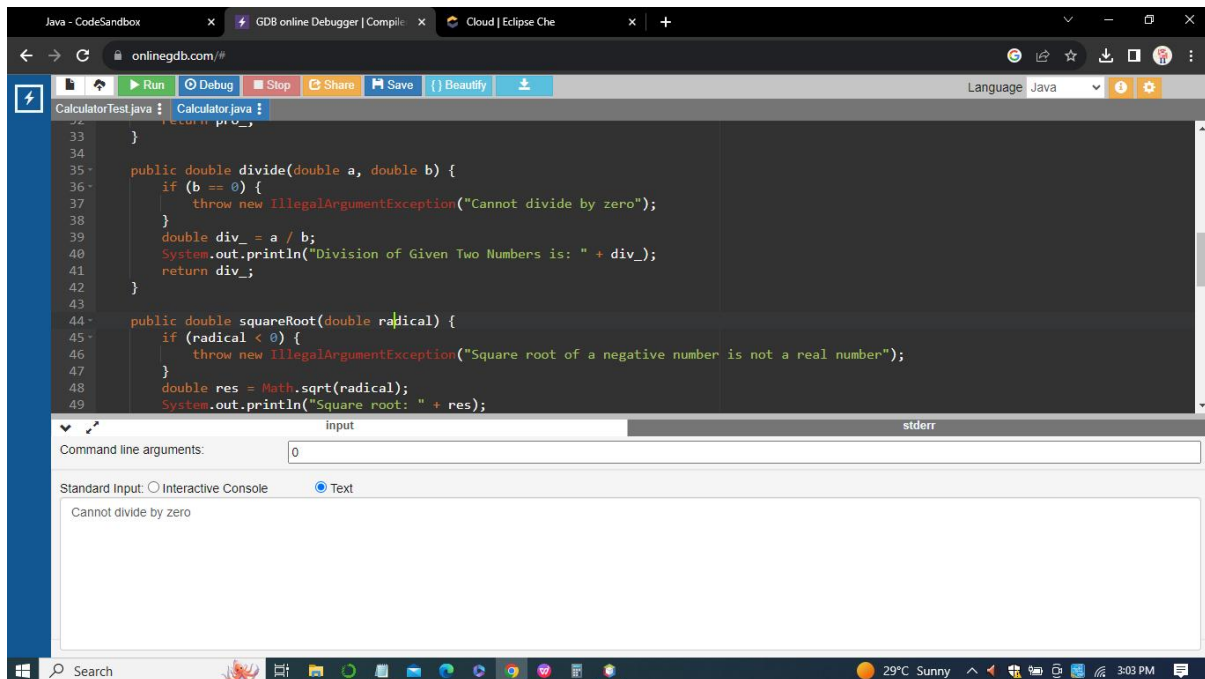
```
33 }
34
35 public double divide(double a, double b) {
36     if (b == 0) {
37         throw new IllegalArgumentException("Cannot divide by zero");
38     }
39     double div_ = a / b;
40     System.out.println("Division of Given Two Numbers is: " + div_);
41     return div_;
42 }
43
44 public double squareRoot(double radical) {
45     if (radical < 0) {
46         throw new IllegalArgumentException("Square root of a negative number is not a real number");
47     }
48     double res = Math.sqrt(radical);
49     System.out.println("Square root: " + res);
50 }
```

The IDE also shows the output of the program in the console:

```
Division of Given Two Numbers is: 5.0
```

**xiv.** Test case 14: Test with division by zero.

testDivideByZero: Input is  $a = 10.0$  and  $b = 0.0$ . Expected output: An `IllegalArgumentException` is expected to be thrown, indicating division by zero.



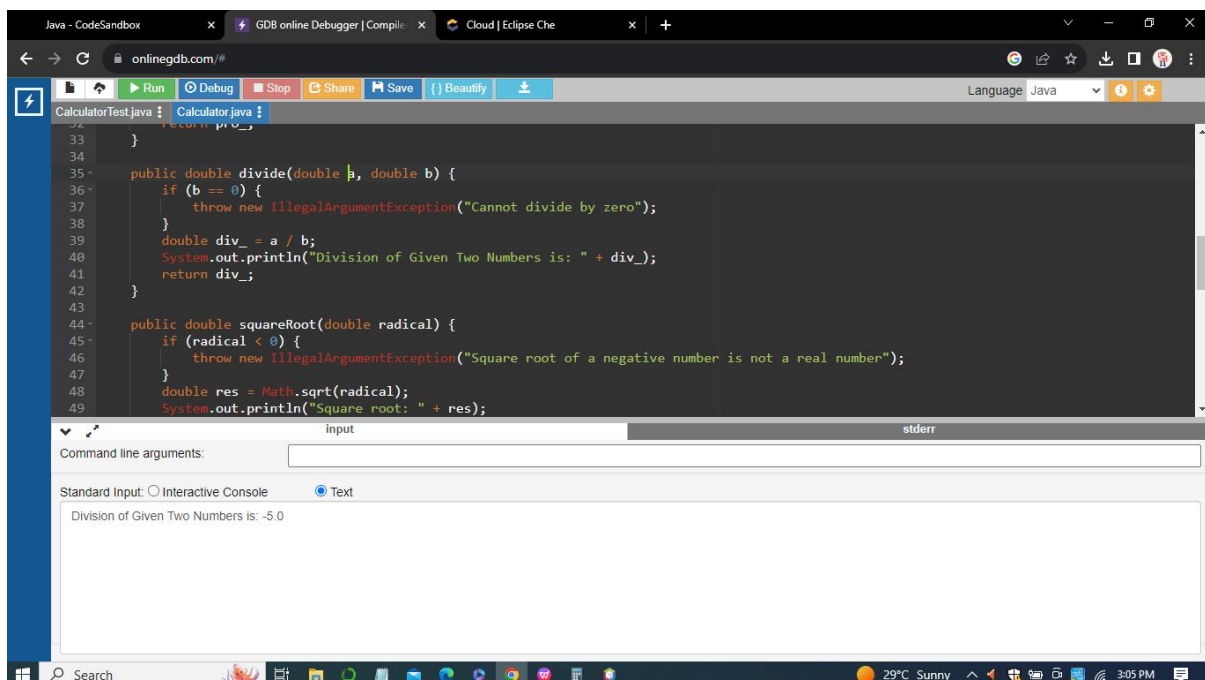
The screenshot shows a web-based Java IDE with the following components:

- Code Editor:** Contains two methods:

```
public double divide(double a, double b) {  
    if (b == 0) {  
        throw new IllegalArgumentException("Cannot divide by zero");  
    }  
    double div_ = a / b;  
    System.out.println("Division of Given Two Numbers is: " + div_);  
    return div_;  
}  
  
public double squareRoot(double radical) {  
    if (radical < 0) {  
        throw new IllegalArgumentException("Square root of a negative number is not a real number");  
    }  
    double res = Math.sqrt(radical);  
    System.out.println("Square root: " + res);  
}
```
- Input Field:** Labeled "input", containing the value "0".
- Standard Input:** Labeled "Standard Input", with radio buttons for "Interactive Console" and "Text". The "Text" option is selected, and the output area displays "Cannot divide by zero".
- UI Elements:** Includes a toolbar with "Run", "Debug", "Stop", "Share", "Save", and "Beautify" buttons. The bottom status bar shows "29°C Sunny" and "3:03 PM".

xv. Test case 15: Test with a negative denominator.

Input is  $a = 10.0$  and  $b = -2.0$ . Expected output:  $-5.0$ , which is the result of dividing  $a$  by  $b$ .



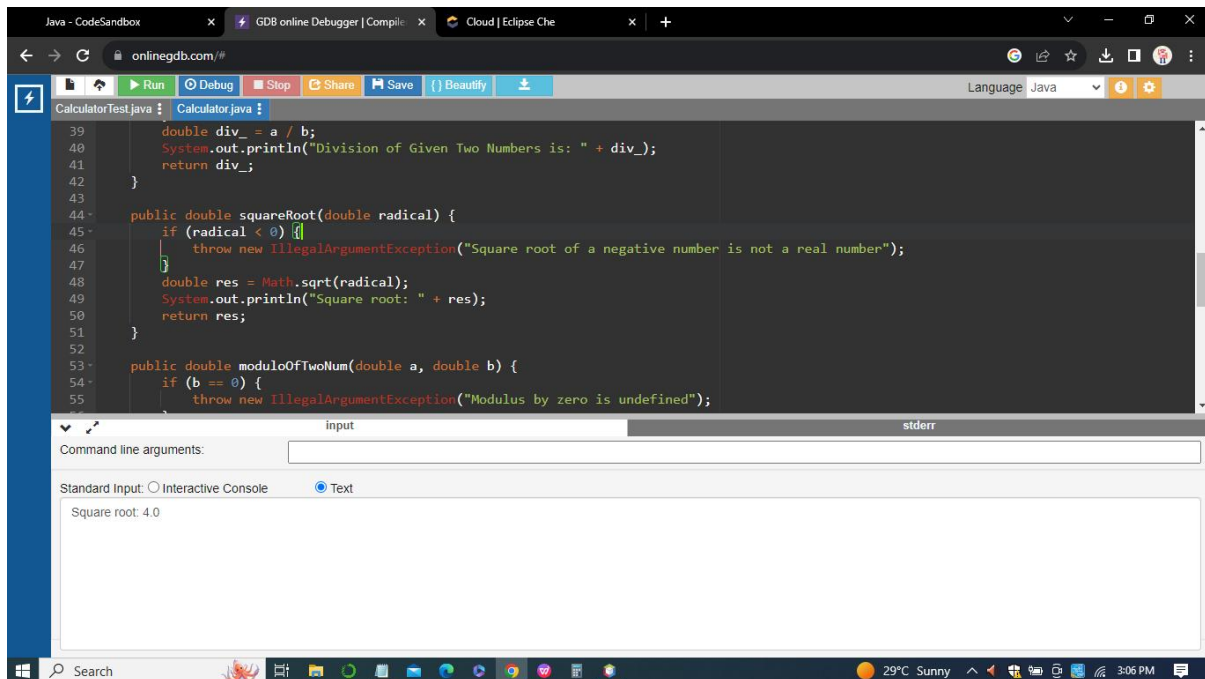
This screenshot shows the same Java IDE after a successful execution:

- Code Editor:** The same code as in the previous screenshot is visible.
- Input Field:** The "input" field is now empty.
- Standard Input:** The "Text" option is still selected, and the output area displays "Division of Given Two Numbers is: -5.0".
- UI Elements:** The interface remains the same, with the status bar showing "29°C Sunny" and "3:05 PM".

### *squareRoot method*

**xvi.** Test case 16: Test with a positive number.

testSquareRootWithPositiveNumber: Input is radical = 16.0. Expected output: 4.0, which is the square root of 16.0.



The screenshot shows a Java IDE with the following code in `Calculator.java`:

```
39 double div_ = a / b;
40 System.out.println("Division of Given Two Numbers is: " + div_);
41 return div_;
42 }
43
44 public double squareRoot(double radical) {
45     if (radical < 0) {
46         throw new IllegalArgumentException("Square root of a negative number is not a real number");
47     }
48     double res = Math.sqrt(radical);
49     System.out.println("Square root: " + res);
50     return res;
51 }
52
53 public double moduloOfTwoNum(double a, double b) {
54     if (b == 0) {
55         throw new IllegalArgumentException("Modulus by zero is undefined");
56     }
57 }
```

The IDE also shows the output in the console:

```
input
Command line arguments:
Standard Input: Interactive Console
Text
Square root: 4.0
```

**xvii.** Test case 17: Test with a negative number.

testSquareRootOfNegativeNumber: Input is radical = -16.0. Expected output: An `IllegalArgumentException` is expected to be thrown, as the square root of a negative number is not a real number.

```

CalculatorTest.java : Calculator.java :
39     double div_ = a / b;
40     System.out.println("Division of Given Two Numbers is: " + div_);
41     return div_;
42 }
43
44 public double squareRoot(double radical) {
45     if (radical < 0) {
46         throw new IllegalArgumentException("Square root of a negative number is not a real number");
47     }
48     double res = Math.sqrt(radical);
49     System.out.println("Square root: " + res);
50     return res;
51 }
52
53 public double moduloOfTwoNum(double a, double b) {
54     if (b == 0) {
55         throw new IllegalArgumentException("Modulus by zero is undefined");
56     }
57     double mod = a % b;
58     System.out.println("Modulo of 2 numbers: " + mod);
59     return mod;
60 }

```

input stderr

Command line arguments:

Standard Input: ☐ Interactive Console ☒ Text

Square root of a negative number is not a real number

*xviii.* Test case 18: Test with zero.

testSquareRootOfZero: Input is radical = 0.0. Expected output: 0.0, as the square root of 0.0 is 0.0.

```

42 }
43
44 public double squareRoot(double radical) {
45     if (radical < 0) {
46         throw new IllegalArgumentException("Square root of a negative number is not a real number");
47     }
48     double res = Math.sqrt(radical);
49     System.out.println("Square root: " + res);
50     return res;
51 }
52
53 public double moduloOfTwoNum(double a, double b) {
54     if (b == 0) {
55         throw new IllegalArgumentException("Modulus by zero is undefined");
56     }
57     double mod = a % b;
58     System.out.println("Modulo of 2 numbers: " + mod);
59     return mod;
60 }

```

input stderr

Command line arguments:

Standard Input: ☐ Interactive Console ☒ Text

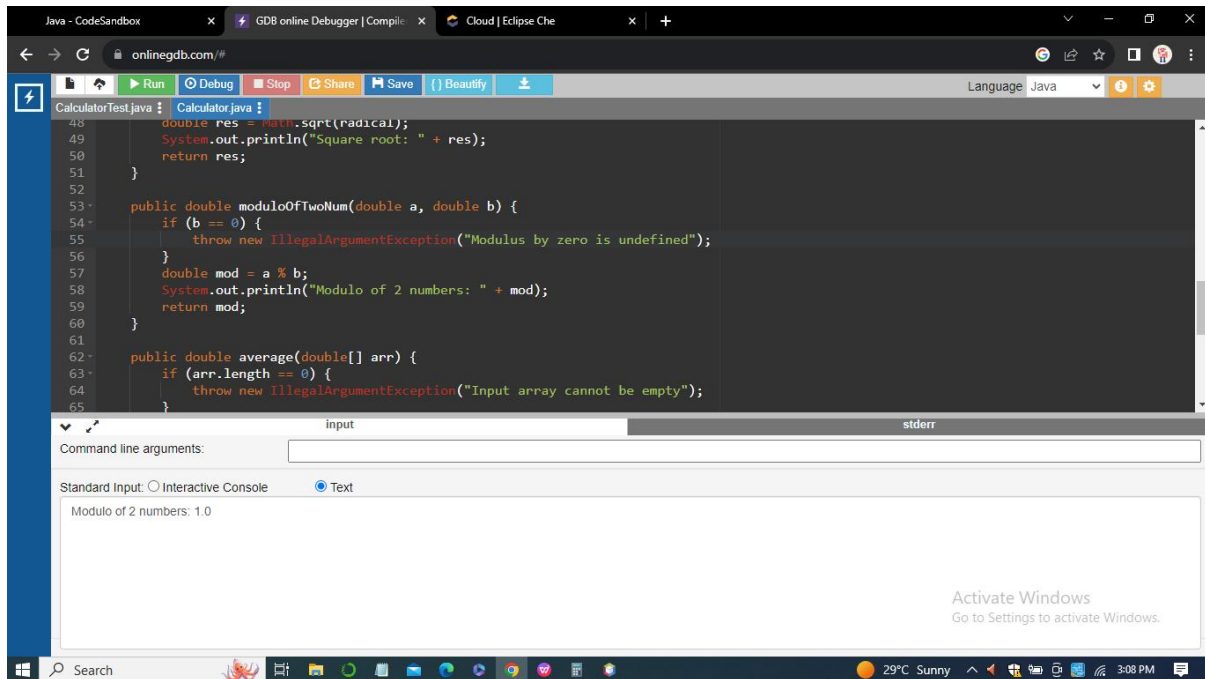
Square root: 0.0

Activate Windows  
Go to Settings to activate Windows.

*moduloOfTwoNum method*

**xix.** Test case 19: Test with positive values for a and b.

testModuloOfTwoNumWithPositiveValues: Input is a = 10.0 and b = 3.0. Expected output: 1.0, which is the result of a % b.



The screenshot shows a Java IDE with the following code in the editor:

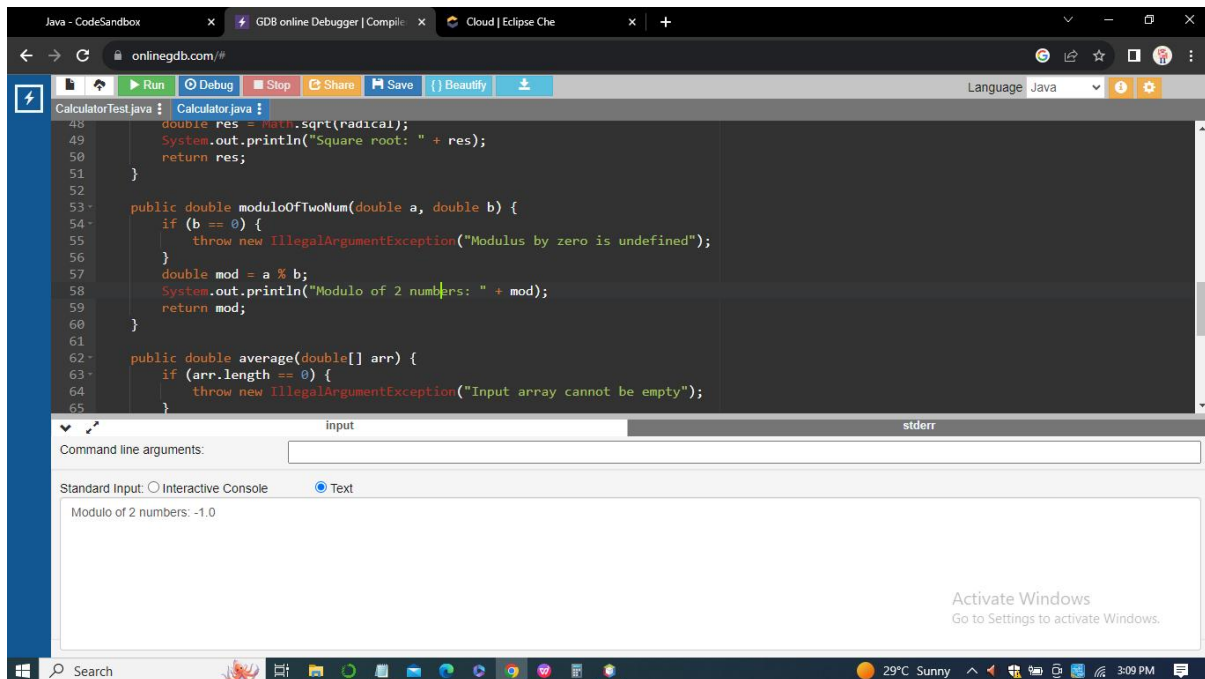
```
48 double res = Math.sqrt(radical);
49 System.out.println("Square root: " + res);
50 return res;
51 }
52
53 public double moduloOfTwoNum(double a, double b) {
54     if (b == 0) {
55         throw new IllegalArgumentException("Modulus by zero is undefined");
56     }
57     double mod = a % b;
58     System.out.println("Modulo of 2 numbers: " + mod);
59     return mod;
60 }
61
62 public double average(double[] arr) {
63     if (arr.length == 0) {
64         throw new IllegalArgumentException("Input array cannot be empty");
65     }
66 }
```

The IDE also shows the output of the program in the console:

```
Modulo of 2 numbers: 1.0
```

**xx.** Test case 20: Test with a negative value for a and a positive value for b.

testModuloOfTwoNumWithNegativeValueA: Input is a = -10.0 and b = 3.0. Expected output: -1.0, which is the result of a % b with a negative a.



```
CalculatorTest.java : Calculator.java :
48 double res = Math.sqrt(radical);
49 System.out.println("Square root: " + res);
50 return res;
51 }
52
53 public double moduloOfTwoNum(double a, double b) {
54     if (b == 0) {
55         throw new IllegalArgumentException("Modulus by zero is undefined");
56     }
57     double mod = a % b;
58     System.out.println("Modulo of 2 numbers: " + mod);
59     return mod;
60 }
61
62 public double average(double[] arr) {
63     if (arr.length == 0) {
64         throw new IllegalArgumentException("Input array cannot be empty");
65     }
66 }
```

input stderr

Command line arguments:

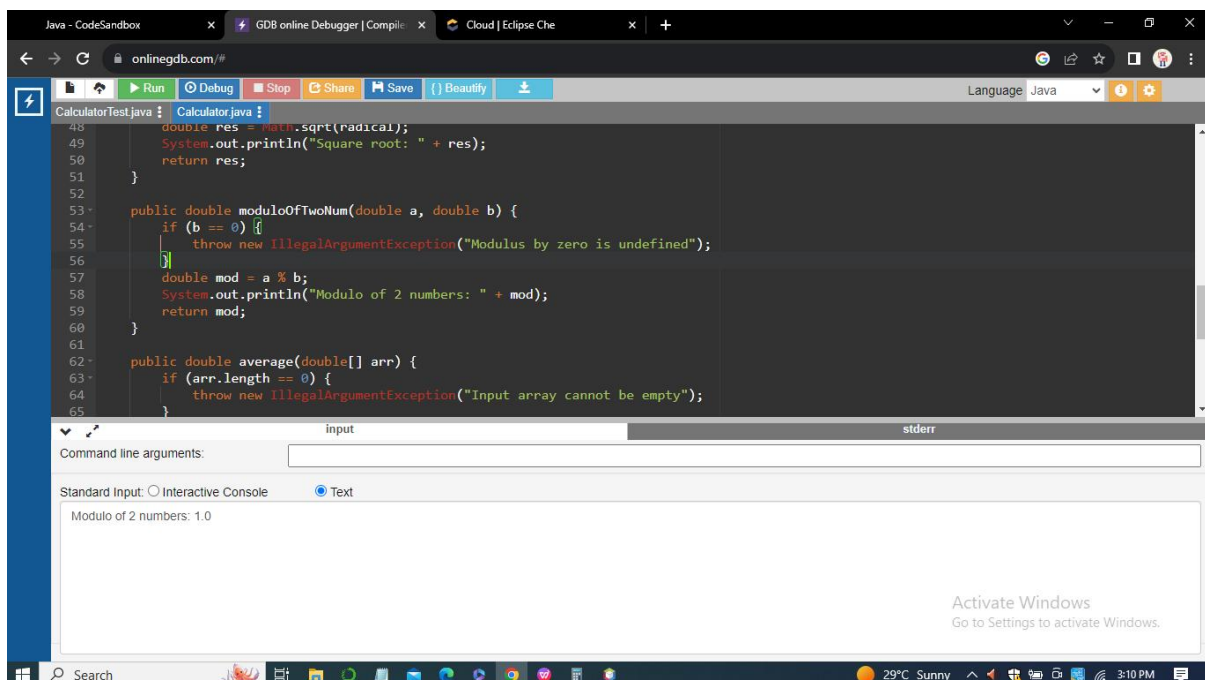
Standard Input: ☐ Interactive Console ☒ Text

Modulo of 2 numbers: -1.0

Activate Windows  
Go to Settings to activate Windows.

**xxi.** Test case 21: Test with a positive value for a and a negative value for b.

testModuloOfTwoNumWithNegativeValueB: Input is a = 10.0 and b = -3.0. Expected output: 1.0, which is the result of a % b with a negative b.



```
CalculatorTest.java : Calculator.java :
48 double res = Math.sqrt(radical);
49 System.out.println("Square root: " + res);
50 return res;
51 }
52
53 public double moduloOfTwoNum(double a, double b) {
54     if (b == 0) {
55         throw new IllegalArgumentException("Modulus by zero is undefined");
56     }
57     double mod = a % b;
58     System.out.println("Modulo of 2 numbers: " + mod);
59     return mod;
60 }
61
62 public double average(double[] arr) {
63     if (arr.length == 0) {
64         throw new IllegalArgumentException("Input array cannot be empty");
65     }
66 }
```

input stderr

Command line arguments:

Standard Input: ☐ Interactive Console ☒ Text

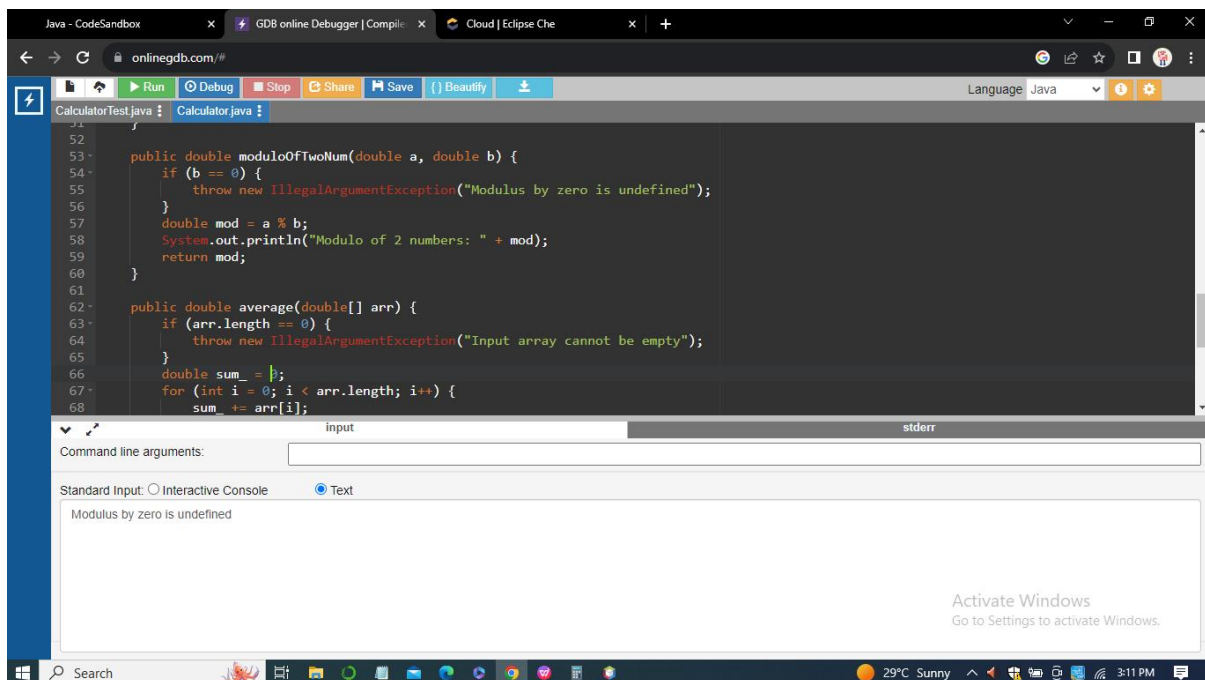
Modulo of 2 numbers: 1.0

Activate Windows  
Go to Settings to activate Windows.

**xxii.** Test case 22: Test with zero as b.



testModuloOfTwoNumWithZeroB: Input is a = 10.0 and b = 0.0. Expected output: An `IllegalArgumentException` is expected to be thrown, as modulus by zero is undefined.



The screenshot shows a web-based Java IDE with the following components:

- Code Editor:** Contains two methods:

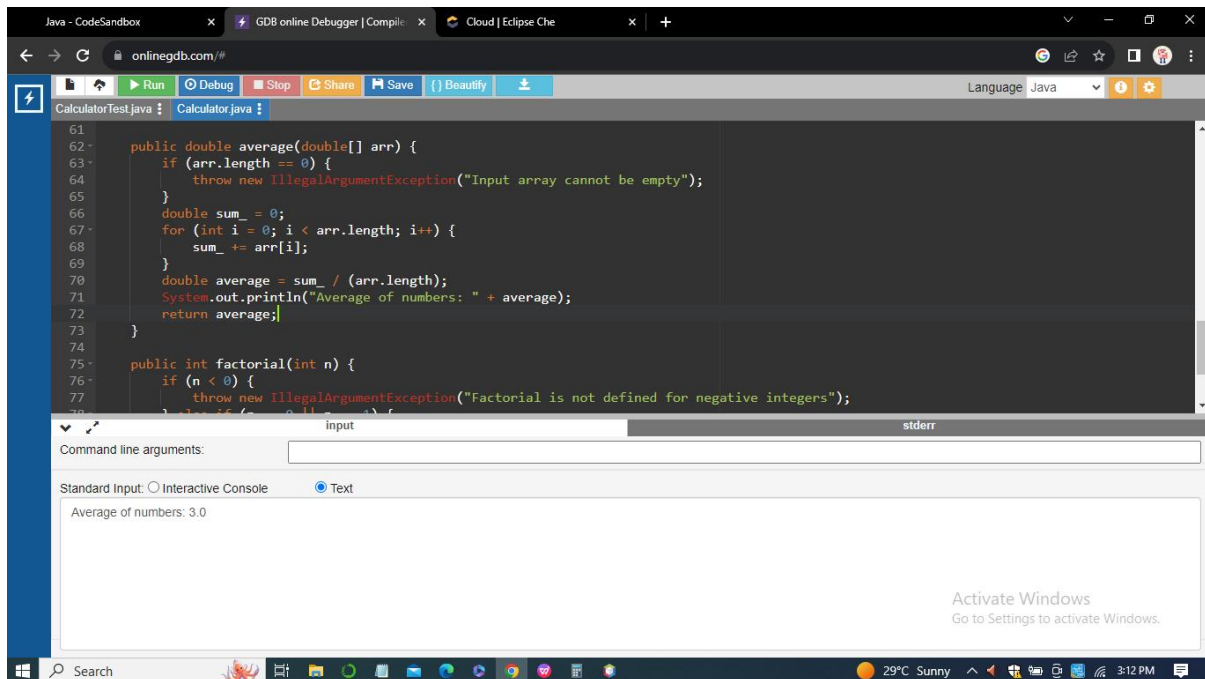
```
52  
53 public double moduloOfTwoNum(double a, double b) {  
54     if (b == 0) {  
55         throw new IllegalArgumentException("Modulus by zero is undefined");  
56     }  
57     double mod = a % b;  
58     System.out.println("Modulo of 2 numbers: " + mod);  
59     return mod;  
60 }  
61  
62 public double average(double[] arr) {  
63     if (arr.length == 0) {  
64         throw new IllegalArgumentException("Input array cannot be empty");  
65     }  
66     double sum = 0;  
67     for (int i = 0; i < arr.length; i++) {  
68         sum += arr[i];  
69     }  
70     return sum / arr.length;  
71 }
```
- Input/Output Panel:** Shows "Standard Input: Text" with the message "Modulus by zero is undefined".
- Taskbar:** At the bottom, it shows a Windows taskbar with a search bar, application icons, and system tray information (29°C Sunny, 3:11 PM).

### *Average method*

**xxiii.** Test case 23: Test with an array of positive numbers.

testAverageWithPositiveNumbers: Input is an array with elements [1.0, 2.0, 3.0, 4.0, 5.0]. Expected output: 3.0, which is the average of the elements.





```
61  
62 public double average(double[] arr) {  
63     if (arr.length == 0) {  
64         throw new IllegalArgumentException("Input array cannot be empty");  
65     }  
66     double sum_ = 0;  
67     for (int i = 0; i < arr.length; i++) {  
68         sum_ += arr[i];  
69     }  
70     double average = sum_ / (arr.length);  
71     System.out.println("Average of numbers: " + average);  
72     return average;  
73 }  
74  
75 public int factorial(int n) {  
76     if (n < 0) {  
77         throw new IllegalArgumentException("Factorial is not defined for negative integers");  
78     }  
79     if (n == 0) {  
80         return 1;  
81     }  
82     return n * factorial(n - 1);  
83 }
```

Input

Command line arguments:

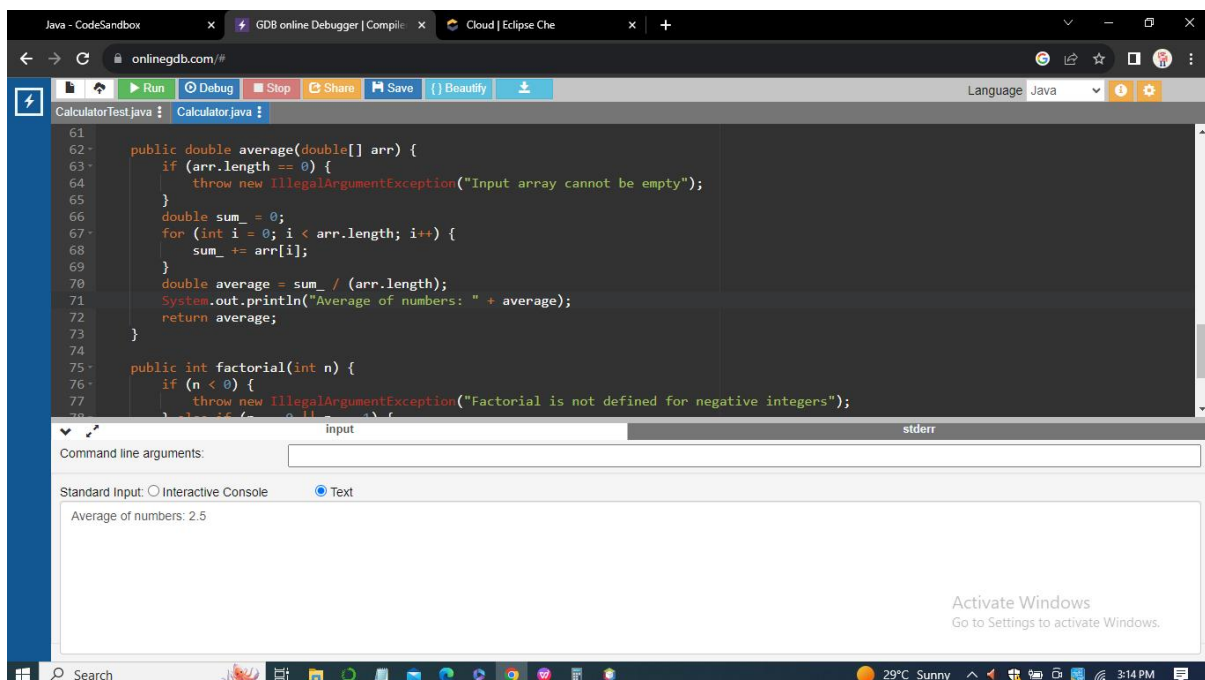
Standard Input: ☐ Interactive Console ☒ Text

Average of numbers: 3.0

Activate Windows  
Go to Settings to activate Windows.

**xxiv.** Test case 24: Test with an array containing a mix of positive and negative numbers.

testAverageMixPositiveNumbers: Input is an array with elements [1.0, -2.0, -3.0, 4.0, 5.0]. Expected output: 2.5, which is the average of the elements.



```
61  
62 public double average(double[] arr) {  
63     if (arr.length == 0) {  
64         throw new IllegalArgumentException("Input array cannot be empty");  
65     }  
66     double sum_ = 0;  
67     for (int i = 0; i < arr.length; i++) {  
68         sum_ += arr[i];  
69     }  
70     double average = sum_ / (arr.length);  
71     System.out.println("Average of numbers: " + average);  
72     return average;  
73 }  
74  
75 public int factorial(int n) {  
76     if (n < 0) {  
77         throw new IllegalArgumentException("Factorial is not defined for negative integers");  
78     }  
79     if (n == 0) {  
80         return 1;  
81     }  
82     return n * factorial(n - 1);  
83 }
```

Input

Command line arguments:

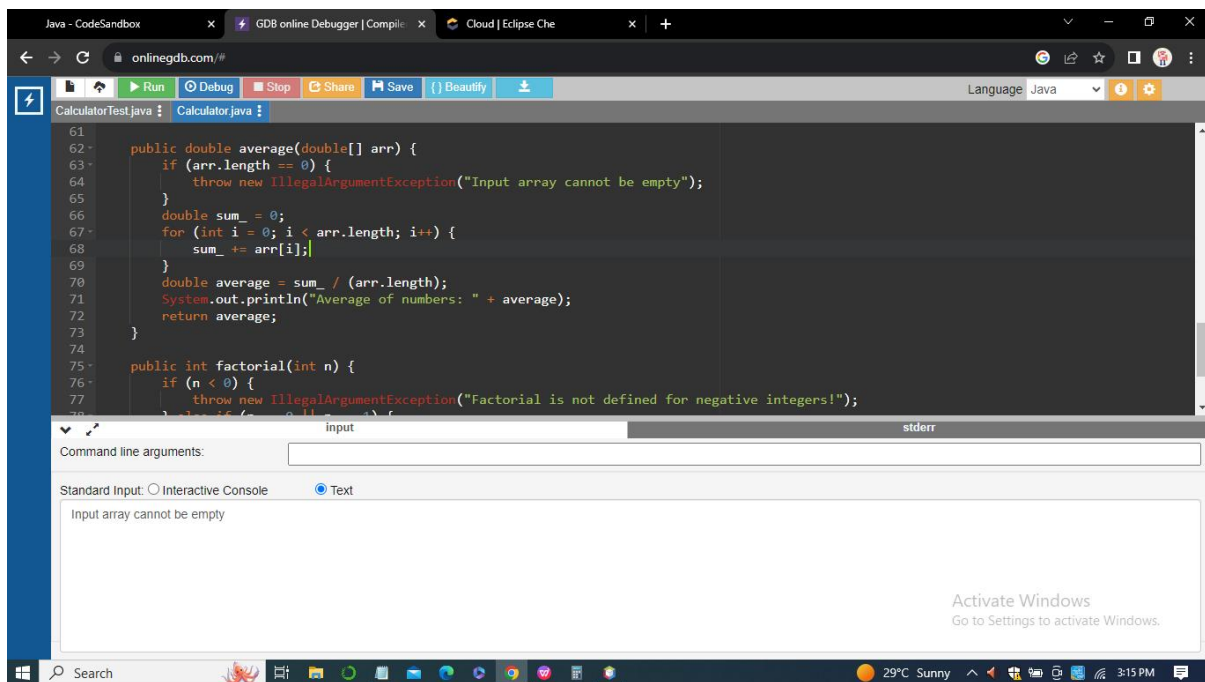
Standard Input: ☐ Interactive Console ☒ Text

Average of numbers: 2.5

Activate Windows  
Go to Settings to activate Windows.

**xxv.** Test case 25: Test with an empty array.

testAverageWithEmptyArray: Input is an empty array. Expected output: An `IllegalArgumentException` is expected to be thrown, as computing the average of an empty array is not possible.



The screenshot shows a web-based Java IDE interface. The top bar includes tabs for 'Java - CodeSandbox', 'GDB online Debugger | Compile', and 'Cloud | Eclipse Che'. The main editor displays a Java file named 'Calculator.java' with the following code:

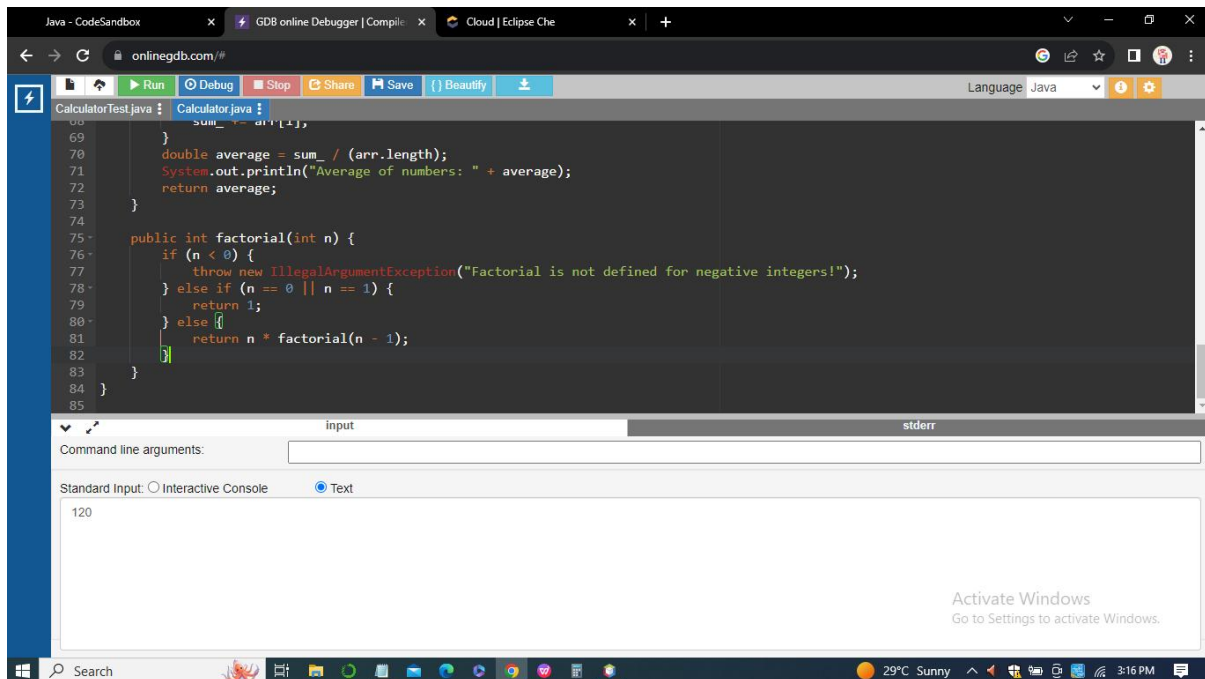
```
61
62 public double average(double[] arr) {
63     if (arr.length == 0) {
64         throw new IllegalArgumentException("Input array cannot be empty");
65     }
66     double sum_ = 0;
67     for (int i = 0; i < arr.length; i++) {
68         sum_ += arr[i];
69     }
70     double average = sum_ / (arr.length);
71     System.out.println("Average of numbers: " + average);
72     return average;
73 }
74
75 public int factorial(int n) {
76     if (n < 0) {
77         throw new IllegalArgumentException("Factorial is not defined for negative integers!");
78     }
79     if (n == 0) {
80         return 1;
81     }
82     return n * factorial(n - 1);
83 }
```

Below the code editor, the 'Input' tab is selected, showing 'Command line arguments:' as an empty field. The 'Standard Input' is set to 'Text', and the input field contains the text 'Input array cannot be empty'. The 'stderr' tab is also visible but empty. At the bottom of the IDE, there is a Windows taskbar with a search bar, system icons, and a clock showing 3:15 PM.

### *factorial method*

**xxvi.** Test case 26: Test with a positive integer.

testFactorialWithPositiveInteger: Input is  $n = 5$ . Expected output: 120, which is the factorial of 5.



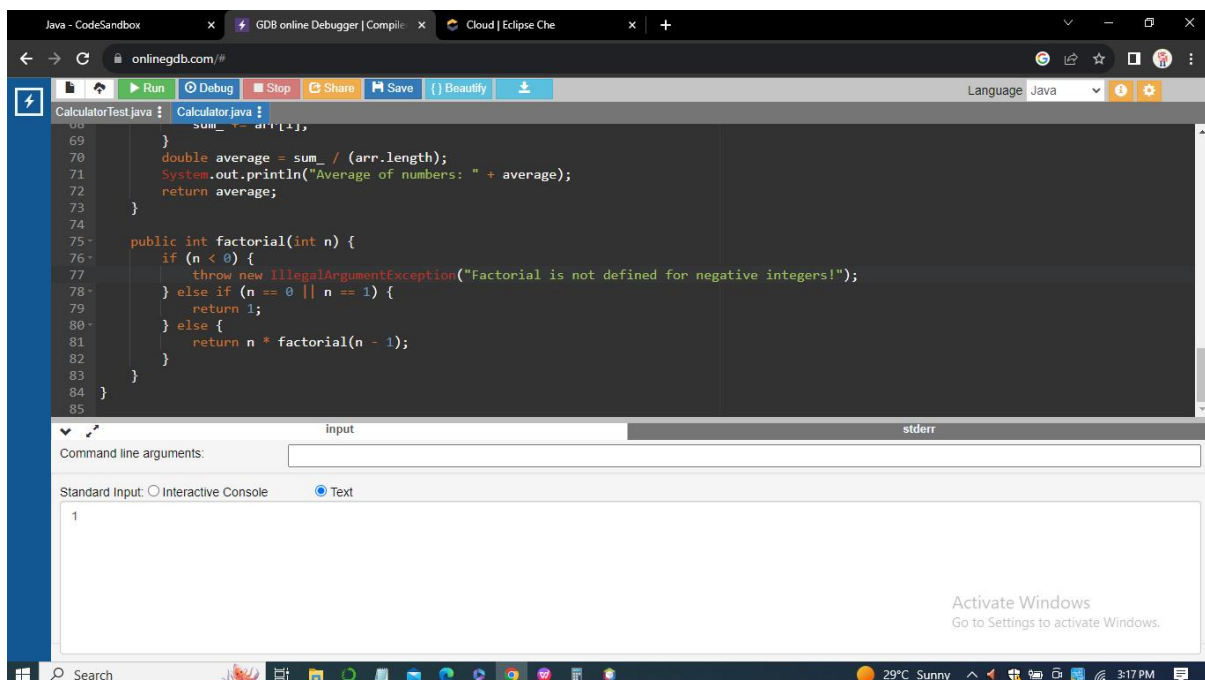
The screenshot shows a web-based Java IDE interface. The top bar includes tabs for 'Java - CodeSandbox', 'GDB online Debugger | Compile', and 'Cloud | Eclipse Che'. The main editor displays a Java file named 'Calculator.java' with the following code:

```
68     sum_ += arr[i];
69     }
70     double average = sum_ / (arr.length);
71     System.out.println("Average of numbers: " + average);
72     return average;
73 }
74
75 public int factorial(int n) {
76     if (n < 0) {
77         throw new IllegalArgumentException("Factorial is not defined for negative integers!");
78     } else if (n == 0 || n == 1) {
79         return 1;
80     } else {
81         return n * factorial(n - 1);
82     }
83 }
84 }
85 }
```

Below the code editor, there is an 'input' field and a 'stderr' field. The 'input' field contains the text '120'. The 'stderr' field is empty. The interface also includes buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', and 'Beautify'. A Windows taskbar is visible at the bottom of the browser window.

**xxvii.** Test case 27: Test with zero.

testFactorialWithZero: Input is  $n = 0$ . Expected output: 1, as the factorial of 0 is defined as 1.



This screenshot is identical to the previous one, showing the same Java code in the editor. However, the 'input' field now contains the text '1' instead of '120'. The 'stderr' field remains empty. The rest of the interface, including the top bar, buttons, and Windows taskbar, is the same.

**xxviii.** Test case 28: Test with a negative integer.

testFactorialWithNegativeInteger: Input is  $n = -5$ . Expected output: An `IllegalArgumentException` is expected to be thrown, as the factorial is not defined for negative integers.

The screenshot shows a web-based Java IDE interface. The main editor displays a Java file named `Calculator.java` with the following code:

```

67-     for (int i = 0; i < arr.length; i++) {
68-         sum_ += arr[i];
69-     }
70-     double average = sum_ / (arr.length);
71-     System.out.println("Average of numbers: " + average);
72-     return average;
73- }
74-
75- public int factorial(int n) {
76-     if (n < 0) {
77-         throw new IllegalArgumentException("Factorial is not defined for negative integers!");
78-     } else if (n == 0 || n == 1) {
79-         return 1;
80-     } else {
81-         return n * factorial(n - 1);
82-     }
83- }
84- }

```

Below the code editor, the 'Input' and 'stderr' tabs are visible. The 'stderr' tab shows the output: `Factorial is not defined for negative integers!`. The 'Input' tab is empty. The bottom of the window shows a Windows taskbar with the time 3:15 PM and temperature 29°C.

## Defects in the provided 2 source files

### *Defect 1: subtract method in Calculator.java*

#### *Defect*

The `subtract` method in the `Calculator` class initializes `diff_` with the first element of the input array, assuming that the array is not empty. This can lead to an `ArrayIndexOutOfBoundsException` if the input array is empty.

#### *Fix*

Add a check at the beginning of the `subtract` method to handle the case when the input array is empty. You can return a meaningful default value (e.g., 0) or throw an exception to indicate that subtraction is not possible.

```

public double subtract(double[] arr) {

    if (arr.length == 0) {

        throw new IllegalArgumentException("Input array cannot be empty");

    }

    double diff_ = arr[0];

    for (int i = 1; i < arr.length; i++) {

        diff_ -= arr[i];

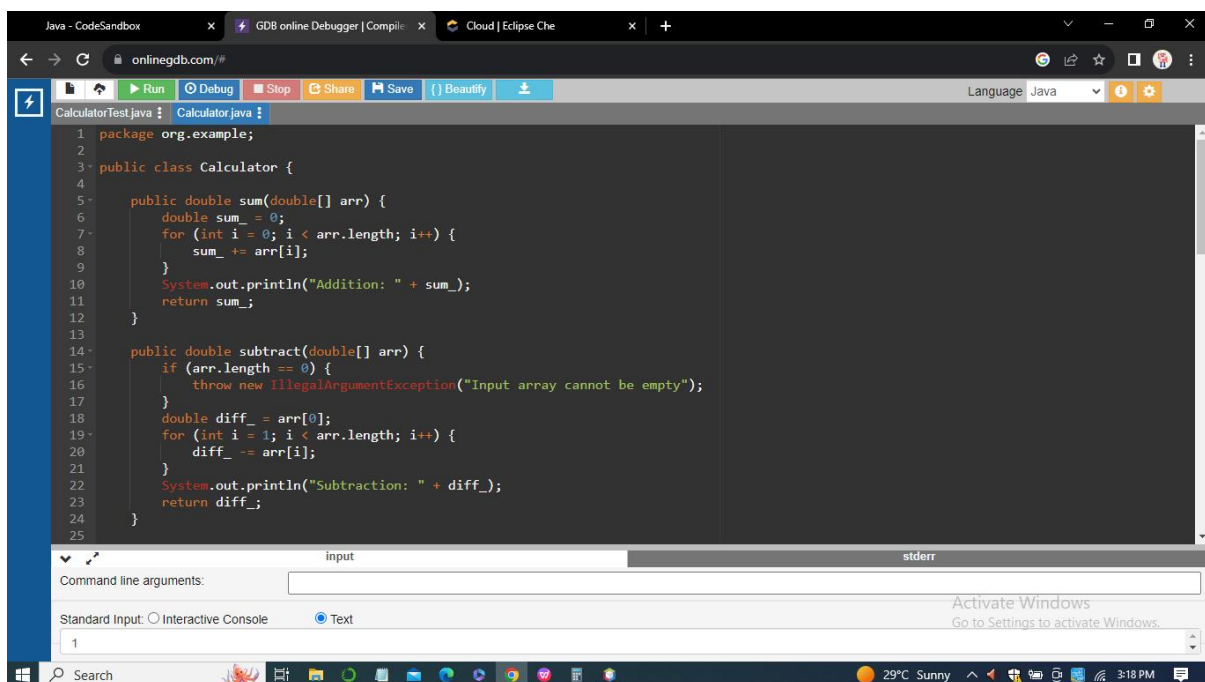
    }

    System.out.println("Subtraction: " + diff_);

    return diff_;

}

```



## Defect 2: divide method in Calculator.java

### Defect

The divide method does not properly handle the case when the denominator is zero. It returns Double.MIN\_VALUE, which is not a suitable indicator of division by zero.

*Fix*

Add a check at the beginning of the divide method to handle the case when the denominator is zero. You can throw an exception or return a specific value (e.g., Double.NaN) to indicate division by zero.

```
public double divide(double a, double b) {

    if (b == 0) {

        throw new IllegalArgumentException("Cannot divide by zero");

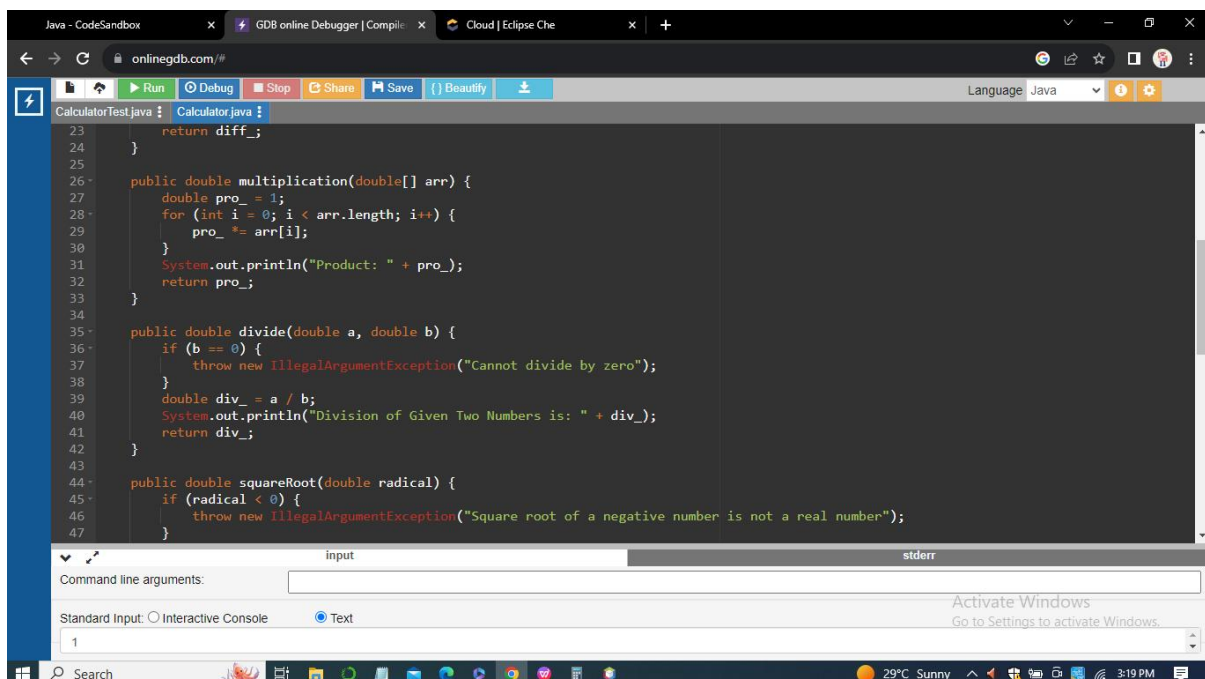
    }

    double div_ = a / b;

    System.out.println("Division of Given Two Numbers is: " + div_);

    return div_;

}
```

***Defect 3: Lack of Unit Tests in CalculatorTest.java****Defect*

The provided CalculatorTest.java file only contains one test method (divide), and it does not cover all the methods in the Calculator class.

### Fix

More test methods were added to cover all the methods in the Calculator class e.g testSumWithPositiveNumbers(),testSumWithMixedNumbers(), testSumWithEmptyArray() etc . Test cases were also written, that check various scenarios, including edge cases and error handling.

```
1 public class CalculatorTest {  
2  
3     Calculator calculator;  
4  
5     // @Before  
6     public void setUp() {  
7         calculator = new Calculator();  
8     }  
9  
10    // @Test  
11    public void testSumWithPositiveNumbers() {  
12        double[] numbers = {1.0, 2.0, 3.0};  
13        double result = calculator.sum(numbers);  
14        assertEquals(6.0, result, 0.001);  
15    }  
16  
17    // @Test  
18    public void testSumWithMixedNumbers() {  
19        double[] numbers = {1.0, -2.0, 3.0};  
20        double result = calculator.sum(numbers);  
21        assertEquals(2.0, result, 0.001);  
22    }  
23  
24    // @Test  
25    public void testSumWithEmptyArray() {  
26        double[] numbers = {};  
27        double result = calculator.sum(numbers);  
28        assertEquals(0.0, result, 0.001);  
29    }  
30  
31 }  
32  
33  
34  
35  
36
```

### Conclusion

In conclusion, this assignment has successfully addressed defects in the Calculator project source code and significantly improved its reliability. By introducing proper error handling and comprehensive unit tests, we have enhanced the overall quality of the codebase. The unit tests provide confidence in the correctness and robustness of the Calculator application, ensuring it functions as expected and handles various scenarios gracefully. This exercise reinforces the importance of testing and defect resolution in software development,

contributing to more reliable and maintainable code. Moving forward, it is essential to continue practicing thorough testing and code review processes to maintain and improve software quality.