ASSIGNMENT

B. LIKHITHA

Assignment 1

Title: Handle Arithmetic Exception

Problem Statement:

Write a Java program that accepts two integers and performs division. Handle the scenario where the denominator is zero.

Requirements / Constraints:

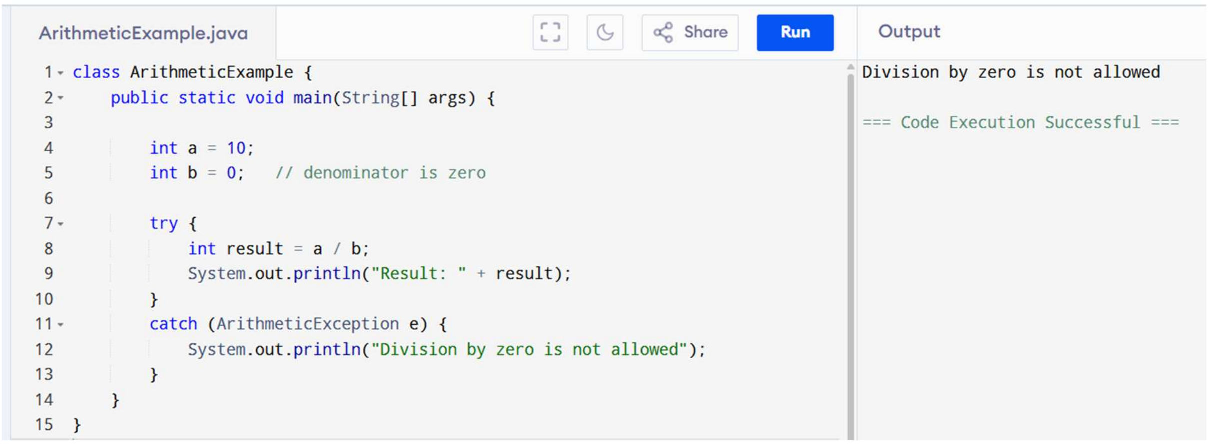
• Use try-catch

• Catch ArithmeticException

• Display a user-friendly message

Expected Outcome:

Program should not crash and should display "Division by zero is not allowed".



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

        int a = 10;
        int b = 0;    // denominator is zero

        try {
            int result = a / b;
            System.out.println("Result: " + result);
        }
        catch (ArithmeticException e) {
            System.out.println("Division by zero is not allowed");
        }
    }
}
```

```
Division by zero is not allowed

=== Code Execution Successful ===
```

Assignment 2

Title: Handle Array Index Exception

Problem Statement:

Create an array of size 5 and try to access an invalid index.

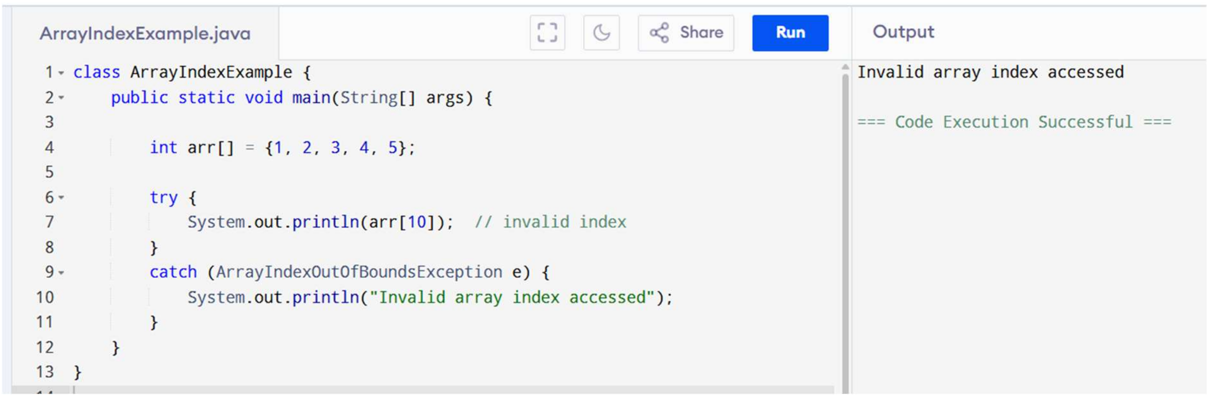
Requirements / Constraints:

• Use try-catch

• Catch ArrayIndexOutOfBoundsException

Expected Outcome:

Program should print "Invalid array index accessed".



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

        int arr[] = {1, 2, 3, 4, 5};

        try {
            System.out.println(arr[10]);  // invalid index
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Invalid array index accessed");
        }
    }
}
```

```
Invalid array index accessed

=== Code Execution Successful ===
```

Assignment 3

Title: Handle Null Pointer Exception

Problem Statement:

Create a string variable with null value and attempt to call a method on it.

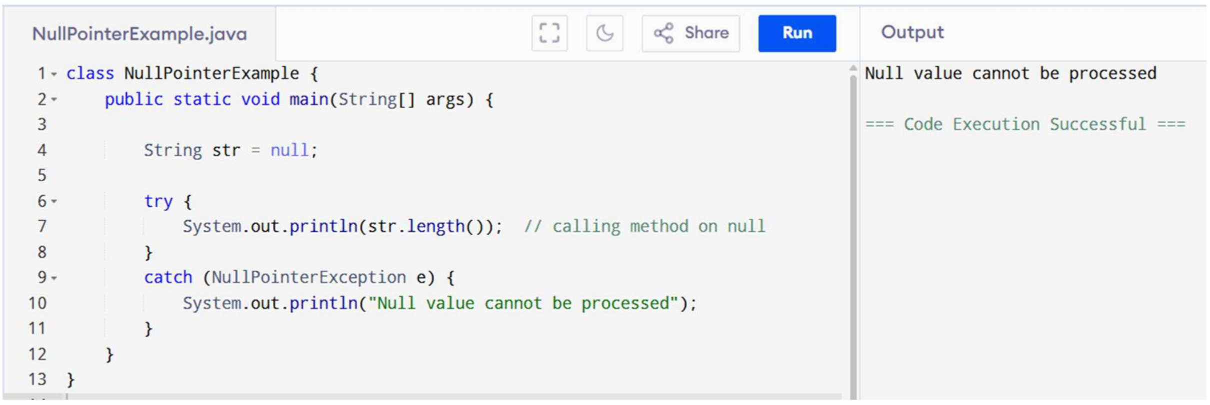
Requirements / Constraints:

• Use try-catch

• Catch NullPointerException

Expected Outcome:

Program should handle the exception gracefully.



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

        String str = null;

        try {
            System.out.println(str.length());  // calling method on null
        }
        catch (NullPointerException e) {
            System.out.println("Null value cannot be processed");
        }
    }
}
```

```
Null value cannot be processed

=== Code Execution Successful ===
```

Assignment 4

Title: Multiple Catch Blocks

Problem Statement:

Write a program that can throw both ArithmeticException and

ArrayIndexOutOfBoundsException.

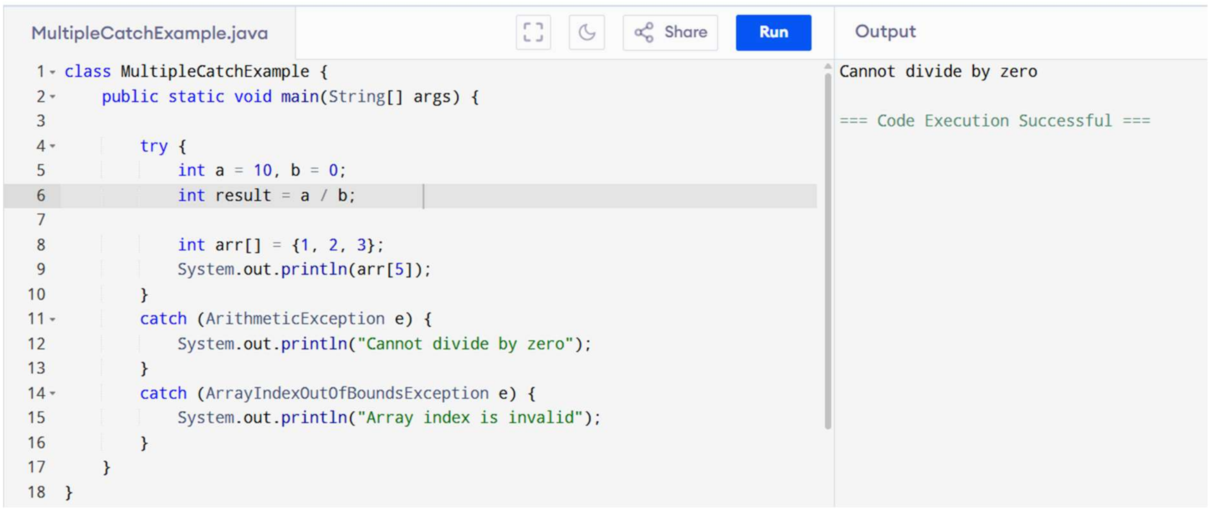
Requirements / Constraints:

• Use multiple catch blocks

• Each exception should have a specific message

Expected Outcome:

Correct exception should be caught and message displayed.



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

        try {
            int a = 10, b = 0;
            int result = a / b;

            int arr[] = {1, 2, 3};
            System.out.println(arr[5]);
        }
        catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero");
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index is invalid");
        }
    }
}
```

```
Cannot divide by zero

=== Code Execution Successful ===
```

Assignment 5

Title: Exception Handling with finally Block

Problem Statement:

Write a program that opens a file and ensures the file resource is closed using finally.

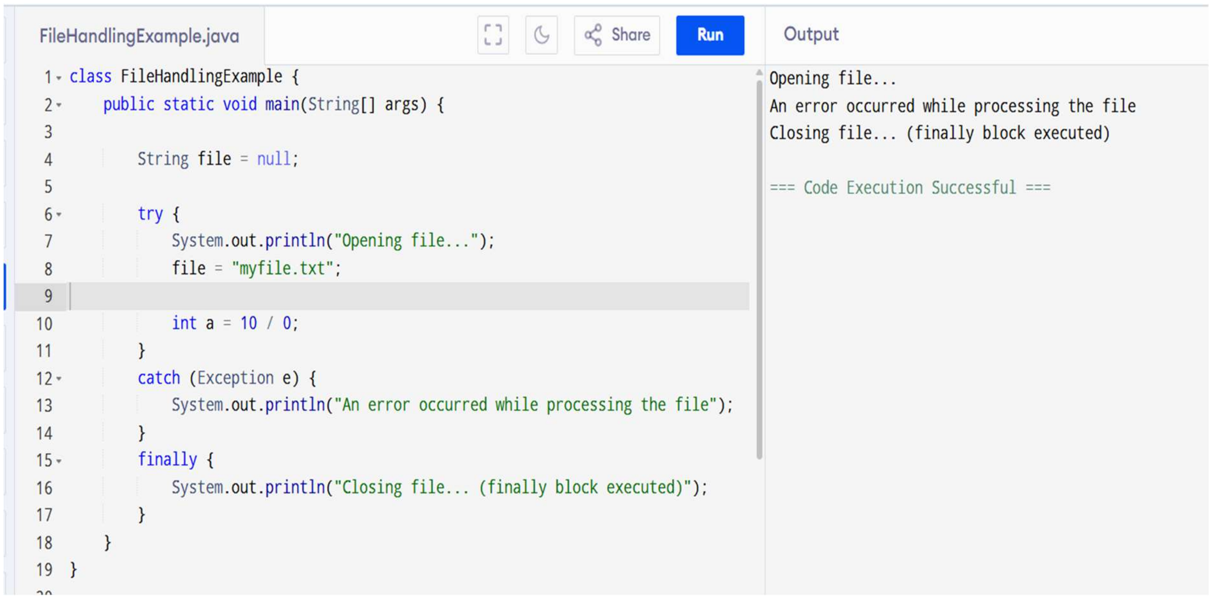
Requirements / Constraints:

• Use try-catch-finally

• Simulate file handling logic

Expected Outcome:

finally block should always execute.



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

        String file = null;

        try {
            System.out.println("Opening file...");
            file = "myfile.txt";

            int a = 10 / 0;
        }
        catch (Exception e) {
            System.out.println("An error occurred while processing the file");
        }
        finally {
            System.out.println("Closing file... (finally block executed)");
        }
    }
}
```

```
Opening file...
An error occurred while processing the file
Closing file... (finally block executed)

=== Code Execution Successful ===
```