Southern University Bangladesh

Department of Computer Science and Engineering

Faculty of Science and Engineering

Object Oriented Programming Lab

Lab Report: 08

Course code: CSE 0613-108

Submitted by:

Name : Sharmistha Chowdhury

ID: 666-61-60

Submitted to :

Mohammed Arif Hasan Chowdhury, Assistant Professor

Dept. of Computer Science and Engineering

Date: 21/05/2024

Problem 01: Write a Java program that throws an exception and catch it using a try-catch block.

Objective:

The objective of the program is to demonstrate the use of try-catch blocks in Java by intentionally causing an ArrayIndexOutOfBoundsException and handling it..

Algorithm:

- Start the main method.
- Inside a try block, declare an array of size 5.
- Attempt to access an element at index 10 of the array, which is out of bounds.
- Catch the ArrayIndexOutOfBoundsException in a catch block.
- In the catch block, print the exception message to the console.
- End the main method.

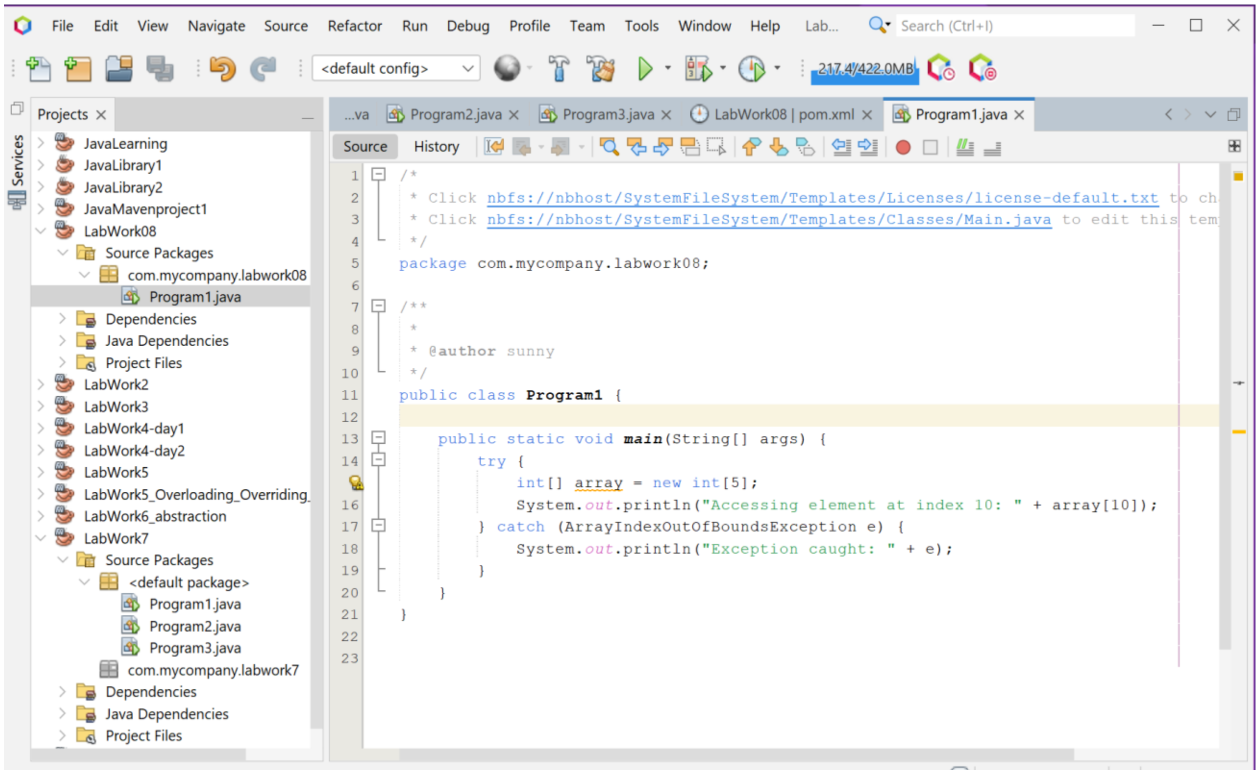
Code:

```
public class Program1 {
  public static void main(String[] args) {
    try {
      int[] array = new int[5];
      System.out.println("Accessing element at index 10: " + array[10]);
    } catch (ArrayIndexOutOfBoundsException e) {
      System.out.println("Exception caught: " + e);
    }
  }
}
```

Input/ Expected Output:

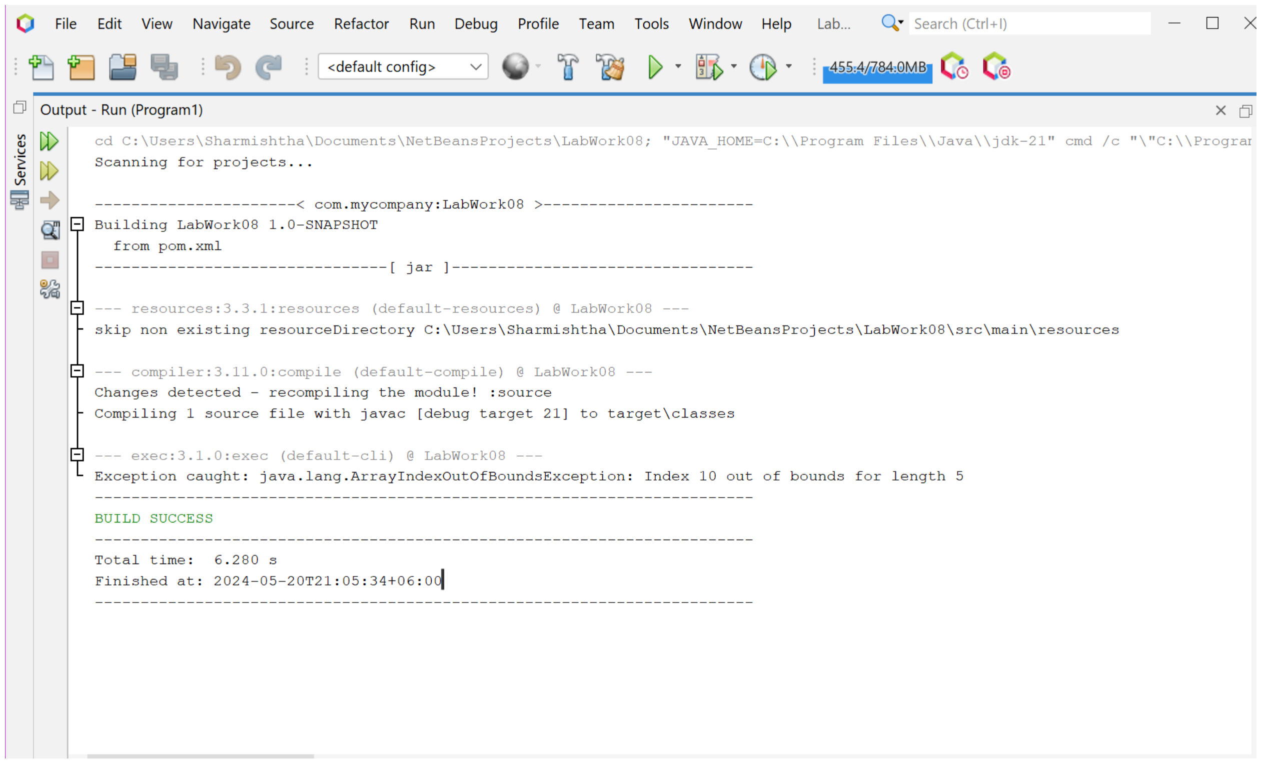
```
Exception caught: java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds
for length 5
```

Screenshot of code edition window:



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to ch
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this tem
 */
package com.mycompany.labwork08;

/**
 *
 * @author sunny
 */
public class Program1 {

    public static void main(String[] args) {
        try {
            int[] array = new int[5];
            System.out.println("Accessing element at index 10: " + array[10]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Exception caught: " + e);
        }
    }
}
```

Screenshot of Output screen/Run screen window:


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Lab...
Search (Ctrl+I)
<default config>
455.4/784.0MB
Output - Run (Program1)
Services

```
cd C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork08; "JAVA_HOME=C:\\Program Files\\Java\\jdk-21" cmd /c "\"C:\\Prograr
Scanning for projects...

----------------------< com.mycompany:LabWork08 >-----------------------
Building LabWork08 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ LabWork08 ---
skip non existing resourceDirectory C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork08\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ LabWork08 ---
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 21] to target\classes

--- exec:3.1.0:exec (default-cli) @ LabWork08 ---
Exception caught: java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 5
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  6.280 s
Finished at: 2024-05-20T21:05:34+06:00
------------------------------------------------------------------------
```

Explanation:

The program begins by entering the main method. Inside a try block, it declares an array of size 5 and then attempts to access an element at index 10. Since the array only has 5 elements, accessing index 10 is out of bounds and causes an ArrayIndexOutOfBoundsException to be thrown. The catch block catches this exception and handles it by printing the exception message to the console. This demonstrates how a try-catch block can be used to handle exceptions in Java.

Discussion

This program is a simple demonstration of exception handling in Java. By placing code that might throw an exception inside a try block and handling the exception in a catch block, the program can prevent the exception from causing the program to terminate prematurely. Instead, the program can handle the exception in a controlled manner and continue executing. This is a fundamental aspect of robust software design. In a real-world application, the catch block might do more than just print a message. For example, it might log the exception, notify the user, or take some corrective action. It's also worth noting that a try block can be followed by multiple catch blocks to handle different types of exceptions.

Problem 02: Write a Java program to create a method that takes an integer as a parameter and throws an exception if the number is odd.

Objective:

The objective of the Program2 class is to demonstrate the use of exceptions in Java by throwing an IllegalArgumentException when an odd number is encountered.

Algorithm:

1. Start the main method.
2. Call the tryNumber method with an even number (18).
3. Inside the tryNumber method, call the checkEvenNumber method inside a try block.
4. In the checkEvenNumber method, check if the number is even. If it's odd, throw an IllegalArgumentException.
5. If no exception is thrown, print a message stating that the number is even.
6. If an exception is thrown, catch it in a catch block and print the exception message.
7. Repeat steps 2-6 with an odd number (7).
8. End the main method.

Code:

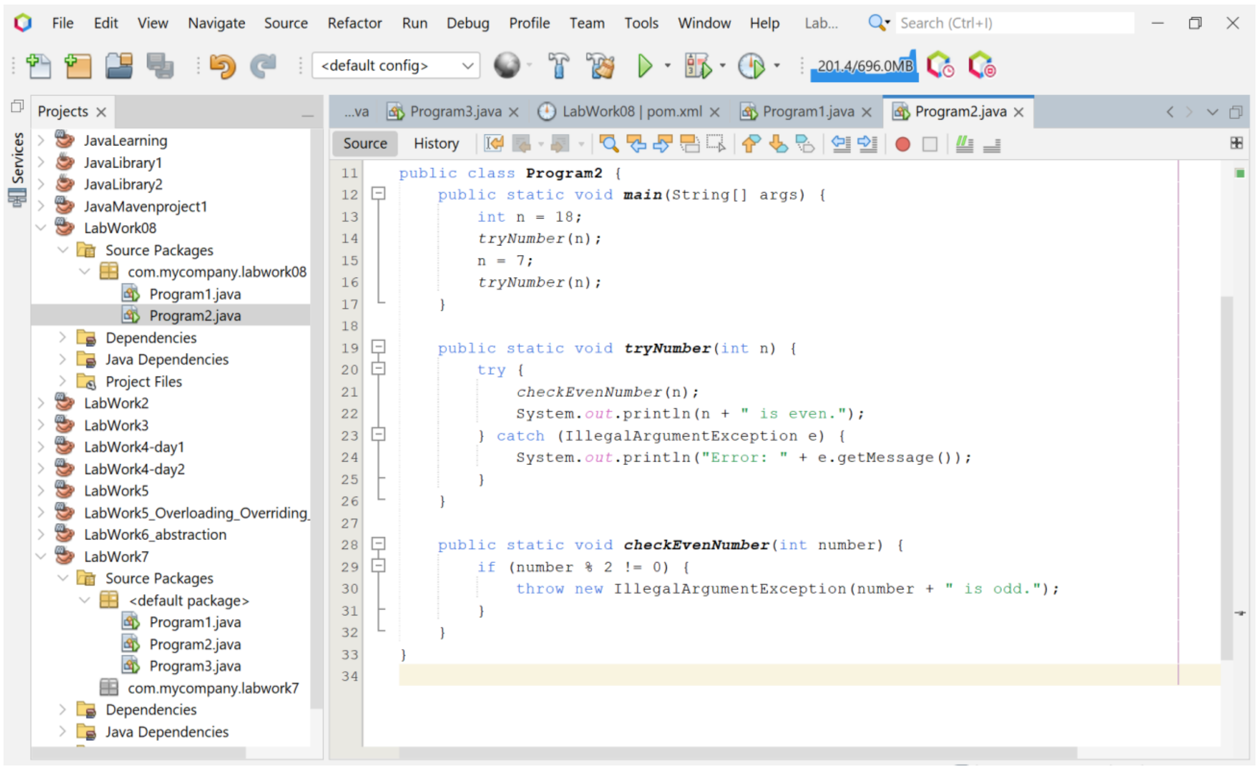
```
public class Program2 {
  public static void main(String[] args) {
    int n = 18;
    tryNumber(n);
    n = 7;
    tryNumber(n);
  }

  public static void tryNumber(int n) {
    try {
      checkEvenNumber(n);
      System.out.println(n + " is even.");
    } catch (IllegalArgumentException e) {
      System.out.println("Error: " + e.getMessage());
```

```
        }
    }

    public static void checkEvenNumber(int number) {
        if (number % 2 != 0) {
            throw new IllegalArgumentException(number + " is odd.");
        }
    }
}
```

Input/ Expected Output:

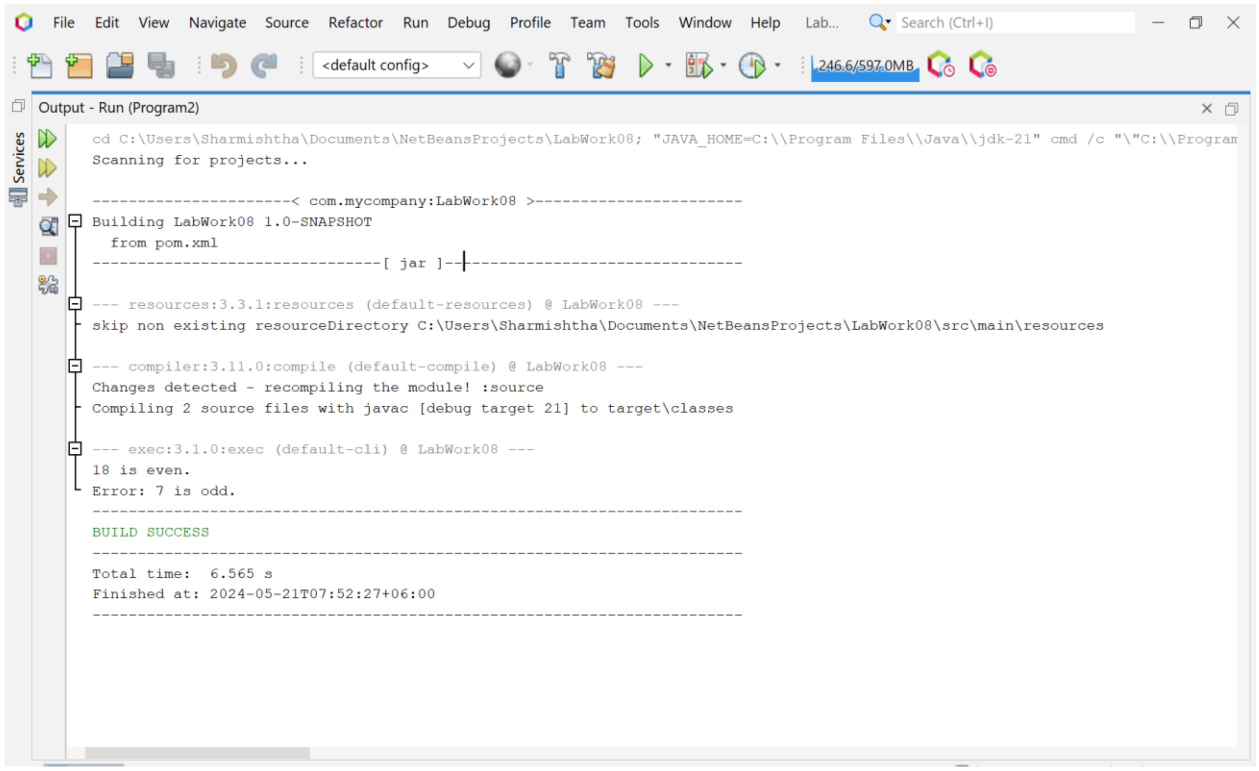
```
18 is even.

Error: 7 is odd
```

Screenshot of code edition window:



```java
public class Program2 {
    public static void main(String[] args) {
        int n = 18;
        tryNumber(n);
        n = 7;
        tryNumber(n);
    }

    public static void tryNumber(int n) {
        try {
            checkEvenNumber(n);
            System.out.println(n + " is even.");
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void checkEvenNumber(int number) {
        if (number % 2 != 0) {
            throw new IllegalArgumentException(number + " is odd.");
        }
    }
}
```

Screenshot of Output screen/Run screen window:



```
cd C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork08; "JAVA_HOME=C:\\Program Files\\Java\\jdk-21" cmd /c "\"C:\\Program
Scanning for projects...

-----------------------< com.mycompany:LabWork08 >-----------------------
Building LabWork08 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ LabWork08 ---
skip non existing resourceDirectory C:\Users\Sharmishtha\Documents\NetBeansProjects\LabWork08\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ LabWork08 ---
Changes detected - recompiling the module! :source
Compiling 2 source files with javac [debug target 21] to target\classes

--- exec:3.1.0:exec (default-cli) @ LabWork08 ---
18 is even.
Error: 7 is odd.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  6.565 s
Finished at: 2024-05-21T07:52:27+06:00
------------------------------------------------------------------------
```

Explanation :

The Program2 class demonstrates how to use exceptions in Java. It defines a tryNumber method that calls a checkEvenNumber method inside a try block. The checkEvenNumber method checks if a number is even and throws an IllegalArgumentException if the number is odd. If an exception is thrown, it is caught in the catch block of the tryNumber method, and the exception message is printed to the console.

Discussion:

This program is a simple demonstration of how to use exceptions in Java. Exceptions are used to indicate that an abnormal condition has occurred that a method can't handle itself. In this case, the abnormal condition is encountering an odd number when only even numbers are expected. By throwing an exception, the checkEvenNumber method indicates that it can't handle this condition and leaves it up to the caller of the method to decide what to do. The caller (tryNumber method) chooses to handle the exception by catching it and printing an error message. This is a common pattern in Java and many other programming languages: a method that encounters an error throws an exception, and the caller of the method catches the exception and handles the error.