# **Experiment 6**

**1. Write a Java program that prompts the user to enter two integers. Perform division operation on these integers and handle the ArithmeticException that may occur if the second integer is zero. Display an appropriate error message if the second integer is zero, otherwise display the result of the division operation.**

# **Program:**

import java.util.Scanner;

 public class Exp6\_1{

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter First Number");

        int a=sc.nextInt();

        System.out.println("Enter Second Number");

        int b=sc.nextInt();

        try{

            int result=a/b;

            System.out.println("Result: "+result);

        }catch(ArithmeticException e){

            System.out.println("Error: Can not divide by zero");

        }finally{

            sc.close();

        }

    }

 }

# **Output:**

Enter First Number

1

Enter Second Number

0

Error: Can not divide by zero

**2. Write a Java program to perform basic arithmetic operations such as addition, subtraction, multiplication, and division. Implement exception handling to catch arithmetic exceptions such as division by zero. Upon encountering such exceptions, the program should gracefully handle the error and provide a meaningful message to the user. Your program should include the following functionalities: Accept two integer inputs from the user.**

**Perform addition, subtraction, multiplication, and division operations on the input numbers. Implement exception handling to catch arithmetic exceptions, specifically division by zero. Display the result of each operation or an appropriate error message if an arithmetic**

**exception occurs.**

# **Program:**

import java.util.Scanner;

public class Exp6\_2 {

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter first integer");

        int a=sc.nextInt();

        System.out.println("Enter Second integer");

        int b=sc.nextInt();

        int Result;

        Result=a+b;

        System.out.println("Addition:"+Result);

        Result=a-b;

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

        Result=a\*b;

        System.out.println("Multiplication:"+Result);

        try{

            Result=a/b;

            System.out.println("Division:"+Result);

        }catch(ArithmeticException e){

            System.out.println("Error:Can not divide by zero");

        }

    }

}

# **Output:**

Enter first integer

4

Enter Second integer

0

Addition:4

Subtraction:4

Multiplication:0

Error:Can not divide by zero

**3. Write a Java program that accepts an array of integers from the user and prints the element at a specified index. Implement exception handling to catch the ArrayIndexOutOfBoundsException in case the user enters an index that is out of bounds for the array.**

**Requirements:**

**The program should prompt the user to enter the size of the array and then the elements of the array. After accepting the array, prompt the user to enter the index of the element they want to retrieve. Implement exception handling to catch ArrayIndexOutOfBoundsException if the user enters an invalid index.**

**Print the element at the specified index if it is within the array bounds; otherwise, display an error message indicating the invalid index.**

**Input:**

**Enter the size of the array: 5**

**Enter the elements of the array:**

**10**

**20**

**30**

**40**

**50**

**Output:**

**Enter the index of the element you want to retrieve: 3**

**Element at index 3: 40**

**Enter the index of the element you want to retrieve: 7**

**Index is out of bounds.**

# **Program:**

import java.util.Scanner;

public class Exp6\_3 {

    public static void main(String args[]){

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the size of the array");

        int index=sc.nextInt();

        int[] Array =new int[index];

        for(int i=0; i<index; i++){

            System.out.print("Enter "+(i+1)+"th Number:");

            Array[i]=sc.nextInt();

        }

        System.out.println("Enter Which index's number you want to acess from Array");

        int a=sc.nextInt();

        try{

            System.out.println("Element at index"+a+"is"+Array[a]);

        }catch(ArrayIndexOutOfBoundsException e){

            System.out.println("Error: Array index out of bound");

        } } }

# **Output:**

Enter the size of the array

5

Enter 1th Number:1

Enter 2th Number:2

Enter 3th Number:3

Enter 4th Number:4

Enter 5th Number:5

Enter Which index's number you want to acess from Array

6

Error: Array index out of bound

**4. Create a Java program to handle the"ArrayIndexOutOfBoundsException" exception class. Prompt the user to input the size of an array and then attempt to access an element at an index beyond the array's bounds. Implement exception handling to gracefully handle this situation and display a user friendly error message indicating that the index is out of bounds.**

**Your program should:**

**Ask the user to input the size of the array.**

**Create an array of the specified size.**

**Try to access an element at an index beyond the array's bounds.**

**Handle the "ArrayIndexOutOfBoundsException" exception and display an appropriate**

**error message.**

# **Program:**

import java.util.Scanner;

public class Exp6\_4 {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");

        int size = scanner.nextInt();

        int[] array = new int[size];

        try {

            int indexOutOfRange = array[size];

        } catch (ArrayIndexOutOfBoundsException e) {

            // Handle the "ArrayIndexOutOfBoundsException" exception and display an appropriate error message

            System.out.println("Error: Index is out of bounds!");

            System.out.println("Array size is " + size + ", so the maximum index is " + (size - 1));

        }

        scanner.close();

    }

}

# **Output:**

Enter the size of the array: 4

Error: Index is out of bounds!

Array size is 4, so the maximum index is 3

**5. Write a Java program to prompt the user to enter an integer value representing their age. However, if the user enters a non-numeric value or a number that cannot be parsed as an integer, the program should catch the NumberFormatException and display an error message, asking the user to enter a valid numeric age.**

# **Program:**

 import java.util.Scanner;

 public class Exp6\_5 {

     public static void main(String args[]) {

         Scanner sc = new Scanner(System.in);

         try {

             System.out.println("Enter your age:");

             int age = sc.nextInt();

         } catch (java.util.InputMismatchException e) {

             System.out.println("Error: Enter valid numeric age");

         }

         sc.close();

     }

 }

# **Output:**

Enter your age:

r

Error: Enter valid numeric age

**6. Write a Java program that demonstrates exception handling using multiple catch blocks. The program should include code that potentially throws exceptions of different types, such as ArithmeticException, ArrayIndexOutOfBoundsException, and NullPointerException. Implement separate catch blocks for each type of exception to handle them appropriately, displaying meaningful error messages for each scenario**

# **Program:**

public class Exp6\_6 {

        public static void main(String[] args) {

            // Potential exceptions: ArithmeticException, ArrayIndexOutOfBoundsException, NullPointerException

            try {

                // Example 1: ArithmeticException

                int result = 10 / 0; // Attempting to divide by zero

            } catch (ArithmeticException e) {

                System.out.println("Arithmetic Exception occurred: " + e.getMessage());

            }

            try {

                // Example 2: ArrayIndexOutOfBoundsException

                int[] arr = new int[5];

                int value = arr[10]; // Accessing an element outside the bounds of the array

            } catch (ArrayIndexOutOfBoundsException e) {

                System.out.println("ArrayIndexOutOfBoundsException occurred: " + e.getMessage());

            }

            try {

                // Example 3: NullPointerException

                String str = null;

                int length = str.length(); // Attempting to call a method on a null reference

            } catch (NullPointerException e) {

                System.out.println("NullPointerException occurred: " + e.getMessage());

            }

        }

    }

# **Output:**

Arithmetic Exception occurred: / by zero

ArrayIndexOutOfBoundsException occurred: Index 10 out of bounds for length 5

NullPointerException occurred: Cannot invoke "String.length()" because "<local1>" is null

**7. Write a Java program to handle exceptions using a finally block. The program should prompt the user to input two integers, divide the first integer by the second integer, and handle any possible exceptions that may occur during the division operation. Ensure that the program always closes any open resources, such as scanner objects, regardless of whether an exception occurs or not.**

# **Program:**

 import java.util.InputMismatchException;

 import java.util.Scanner;

 public class Exp5\_7 {

     public static void main(String args[]) {

         Scanner sc = new Scanner(System.in);

         try {

             System.out.println("Enter first integer");

             int a = sc.nextInt();

             System.out.println("Enter Second integer");

             int b = sc.nextInt();

             int result = a / b;

             System.out.println("Division is " + result);

         } catch (InputMismatchException e) {

             System.out.println("Error: Please enter valid integers.");

         } catch (ArithmeticException e) {

             System.out.println("Error: Can not divide by zero");

         } finally {

             sc.close();

         }

     }

 }

# **Output:**

Enter first integer

5

Enter Second integer

u

Error: Please enter valid integers.

**8. Write a Java program that demonstrates exception handling using the finally block. The program should attempt to perform a task that might throw an exception, such as dividing by zero or accessing an out-of-bounds index in an array. Regardless of whether an exception occurs or not, the program should include a finally block to execute cleanup code, such as closing resources or releasing locks.**

# **Program:**

public class Exp6\_8 {

    public static void main(String[] args) {

        try {

            // Attempting division by zero

            int result = divideByZero();

            System.out.println("Result: " + result);

        } catch (ArithmeticException e) {

            System.out.println("Error: Division by zero");

        } finally {

            // Cleanup code

            System.out.println("Cleanup code executed.");

        }

    }

    // Method to demonstrate division by zero

    public static int divideByZero() {

        int numerator = 10;

        int denominator = 0;

        return numerator / denominator;

    }

}

# **Output:**

Error: Division by zero

Cleanup code executed.