**Week 5 (To study about Interfaces and Exception Handling in Java)**

**Program 1:** WAP to compute the area of circle and rectangle using Interfaces.

**Program 2:** WAP that uses interface for the implementation of fixed-size and dynamic-size stacks (for dynamic stack size should be redefine as per the number of elements).

**Program 3:** Write a program for exception handling with StringIndexOutOfBound exception

* Create an object of the class having StringIndexOutOfBound exception whenever an index is invoked of a string, which is not in the range.
* Each character of a string object is stored in a particular index starting from 0.
* To get a character present in a particular index of a string you can use a method charAt(int) of java.lang.String where int argument is the index.

import java.util.Scanner;

public class StringIndexOutOfBoundsExceptionExample {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String str = scanner.nextLine();

try {

System.out.print("Enter an index to get the character at that position: ");

int index = scanner.nextInt();

char ch = str.charAt(index);

System.out.println("The character at index " + index + " is " + ch);

} catch (StringIndexOutOfBoundsException e) {

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

} finally {

scanner.close();

}

}

}

**Program 4:** An array is declared with 5 elements. Then the code tries to access the 6th element of the array which throws an exception. Write the program for this.

public class ArrayIndexOutOfBoundsExceptionExample {

public static void main(String[] args) {

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

try {

int sixthElement = arr[5];

System.out.println("The sixth element is: " + sixthElement);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Array index is out of bounds!");

}

}

}

**Program 5:** Write a suitable program for the following conditions:

1. A try block followed by multiple catch blocks

public class TryCatchExample {

public static void main(String[] args) {

try {

int[] arr = new int[5];

arr[5] = 10;

} catch (ArithmeticException e) {

System.out.println("ArithmeticException caught!");

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBoundsException caught!");

} catch (Exception e) {

System.out.println("Exception caught!");

}

}

}

2. Catching multiple type of exceptions

public class MultipleExceptionExample {

public static void main(String[] args) {

try {

int result = 10 / 0;

int[] arr = new int[5];

arr[5] = 10;

} catch (ArithmeticException | ArrayIndexOutOfBoundsException e) {

System.out.println("Exception caught: " + e);

}

}

}

3. Using throws/throw keywords

public class ThrowsExample {

public static void main(String[] args) throws InterruptedException {

Thread.sleep(1000);

}

}

4. Using finally block

public class FinallyExample {

public static void main(String[] args) {

try {

int result = 10 / 0;

System.out.println(result);

} catch (ArithmeticException e) {

System.out.println("ArithmeticException caught!");

} finally {

System.out.println("Finally block executed!");

}

}

}

5. Using try-with-resources

public class TryWithResourcesExample {

public static void main(String[] args) {

try (FileReader reader = new FileReader("example.txt")) {

int data = reader.read();

while (data != -1) {

System.out.print((char) data);

data = reader.read();

}

} catch (IOException e) {

System.out.println("IOException caught: " + e);

}

}

}

6. User-defined exceptions

class NegativeNumberException extends Exception {

public NegativeNumberException() {

super("Negative number not allowed!");

}

}

public class UserDefinedExceptionExample {

public static void main(String[] args) {

try {

int num = -10;

if (num < 0) {

throw new NegativeNumberException();

}

System.out.println("The number is: " + num);

} catch (NegativeNumberException e) {

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

}

}

}