Program 7:

Title: "Java Program: Resizable Interface for Object Resizing with Rectangle Implementation".

Problem Description: Develop a JAVA program to create an interface Resizable with methods resizeWidth(int width) and resizeHeight(int height) that allow an object to be resized. Create a class Rectangle that implements the Resizable interface and implements the resize methods.

Method: Ensure that the program follows proper object-oriented principles, such as encapsulation and abstraction, and provides clear and concise output demonstrating the resizing functionality.

Theory Reference: Module 3 Page no:157

Code:

```
import java.util.Scanner;
// Interface Resizable
interface Resizable {
    void resizeWidth(int width);
    void resizeHeight(int height);
}

// Class Rectangle implementing Resizable
class Rectangle implements Resizable {
    private int width;
    private int height;

    // Constructor
    public Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }

    // Method to resize the width
    @Override
    public void resizeWidth(int newWidth) {
        this.width = newWidth;
        System.out.println("New width: " + width);
```

```
    }
    // Method to resize the height
    @Override
    public void resizeHeight(int newHeight) {
        this.height = newHeight;
        System.out.println("New height: " + height);
    }
    // Method to display the current dimensions of the rectangle
    public void displayDimensions() {
        System.out.println("Current dimensions: " + width + " x " + height);
    }
}
// Main class to test the program
public class p7{
    public static void main(String[] args) {
        // Create a rectangle with width 10 and height 5
        Scanner sc=new Scanner(System.in);

        Rectangle rectangle = new Rectangle(10,5) ;
        rectangle.displayDimensions();

        // Resize the rectangle's width and height
        System.out.println("Enter the new width and height");
        int w=sc.nextInt();
        int h=sc.nextInt();
        rectangle.resizeWidth(w);
        rectangle.resizeHeight(h);
        rectangle.displayDimensions();
    }
}
```

Program 8:

Title: "Java Program: Custom Exception Handling for DivisionByZero and Arithmetic Exceptions".

Problem Description: Develop a Java program that demonstrates the handling of custom exceptions, specifically for DivisionByZero and Arithmetic exceptions. You are required to utilize try-catch blocks along with throw statements to handle these exceptions gracefully.

Method: Ensure that the program demonstrates the proper usage of try-catch blocks, throw statements, and custom exception handling for DivisionByZero and Arithmetic exceptions.

Theory Reference: Module 4 Page no:205

Code:

```
import java.util.Scanner;
//Custom exception class
class DivisionByZeroException extends Exception {
public DivisionByZeroException(String message) {
super(message);
}
}

public class pgm8 {
// Method to perform division and throw custom exception if denominator is zero
static double divide(int numerator, int denominator) throws DivisionByZeroException {
if (denominator == 0) {
throw new DivisionByZeroException("Cannot divide by zero!");
}
return (double) numerator / denominator;
}

public static void main(String[] args) {
// TODO Auto-generated method stub
Scanner input = new Scanner(System.in);
System.out.println("Enter numerator and denominator ");
int numerator = input.nextInt();
int denominator = input.nextInt();
```

```
try {
double result = divide(numerator, denominator);
System.out.println("Result of division: " + result);
} catch (DivisionByZeroException e) {
System.out.println("Exception caught: " + e.getMessage());
}
finally {
System.out.println("Finally block executed");
}
}
}
```

Program 9:

Title: "Write a Java program to generate random numbers using multiple threads.

Problem Description: Develop a Java program that implements a multi-threaded application with three threads. Each thread has a specific task as described as follows: First Thread (Random Number Generator): This thread generates a random integer every 1 second. Second Thread (Square Computation): This thread receives the random integer generated by the first thread and computes its square. After computing the square, it prints the result. Third Thread (Cube Computation): This thread receives the random integer generated by the first thread and computes its cube. After computing the cube, it prints the result.

Method: Program should demonstrate the multi-threading capabilities of Java and showcase the asynchronous computation of squares and cubes. Additionally, it should handle synchronization and data sharing effectively to prevent race conditions and ensure thread safety.

Theory Reference: Module 4 Page no:205

Code:

```
import java.util.*;
class Square implements Runnable {
public int x;
public Square(int num) {
x = num;
}
public void run() {
System.out.println("Thread 2: Square of " + x + " is: " + x * x);
}
}
class Cube implements Runnable {
public int x;
public Cube(int num) {
x = num;
}
public void run() {
System.out.println("Thread 3: Cube of " + x + " is: " + x * x * x);
}
}
class GenerateNos implements Runnable {
```

```
public void run() {
int num = 0;
Random r = new Random();
try {
for (int i = 0; i < 5; i++) {
num = r.nextInt(100);
System.out.println("Thread 1: Generated random number is " + num);
Thread t1 = new Thread(new Square(num));
t1.start();
Thread t2 = new Thread(new Cube(num));
t2.start();
Thread.sleep(1000);
System.out.println("-------------------------------------");
}
} catch (Exception ex) {
System.out.println(ex.getMessage());
}
}
} // End of GenerateNos
public class pgm9_2 {

public static void main(String[] args) {
Thread t = new Thread(new GenerateNos());
t.start();
}
}
```