# Department of Computing

# CS 212: Object Oriented Programming

# Class: BSCS-8AB

# Lab 12: Exception Handling

# Date: May 6th, 2019

# Instructor: Hirra Anwar

**Learning Objectives**

The learning objective of this lab is to understand and practice the concept of Exception Handling in Java. This will include how to throw an exception, how to declare that exception can be raised from a piece of code, how to catch the exception using try catch block. In addition to this, you will also learn how to create your own exception handling classes and use it for exception handling; this class should be inheriting one of the Exception parent classes.

**Activity # 1**

Execute the code examples given in the lecture slides of Exception Handling to get understanding of the concepts.

**Activity # 2** The following example demonstrates declaring, throwing, and catching exceptions.

public class CircleWithException {

/\*\* The radius of the circle \*/

private double radius;

/\*\* The number of the objects created \*/

private static int numberOfObjects = 0;

/\*\* Construct a circle with radius 1 \*/

public CircleWithException() {

this(1.0);

numberOfObjects++;

}

/\*\* Construct a circle with a specified radius \*/

public CircleWithException(double newRadius) {

setRadius(newRadius);

numberOfObjects++;

}

/\*\* Return radius \*/

public double getRadius() {

return radius;

}

/\*\* Set a new radius \*/

public void setRadius(double newRadius)

throws IllegalArgumentException {

if (newRadius >= 0)

radius = newRadius;

else

throw new IllegalArgumentException("Radius cannot be negative");

}

/\*\* Return numberOfObjects \*/

public static int getNumberOfObjects() {

return numberOfObjects;

}

/\*\* Return the area of this circle \*/

public double findArea() {

return radius \* radius \* 3.14159;

}

}

A test program which uses the CircleWithException class is given below:

|  |
| --- |
| public class TestCircleWithException {  public static void main(String[] args) {  try {  CircleWithException c1 = new CircleWithException(5);  CircleWithException c2 = new CircleWithException(-5);  CircleWithException c3 = new CircleWithException(0);  CircleWithException c4 = new CircleWithException();  }  catch (IllegalArgumentException ex) {  System.out.println(ex);  }  System.out.println("Number of objects created: " +  CircleWithException.getNumberOfObjects());  }  } |

What will be the output?

|  |
| --- |
|  |

What happens if we remove the clause throws IllegalArgumentException from the setRadius method declaration, and re-compile the CircleWithException class? Would it compile? If so, why?

|  |
| --- |
|  |

What happens if we do not handle the IllegalArgumentException in the TestCircleWithException class by not using the try statement?

|  |
| --- |
|  |

Java provides quite a few exception classes. Use them whenever possible instead of defining your own exception classes. However, if you run into a problem that cannot be adequately described by the predefined exception classes, you can create your own exception class, derived from Exception or from a subclass of Exception, such as IOException.

In CircleWithException, the setRadius method throws an exception if the radius is negative. Suppose you wish to pass the radius to the handler. In that case, you can define a custom exception class, as shown below:

|  |
| --- |
| public class InvalidRadiusException extends Exception {  private double radius;  /\*\* Construct an exception \*/  public InvalidRadiusException(double radius) {  super("Invalid radius " + radius);  this.radius = radius;  }  /\*\* Return the radius \*/  public double getRadius() {  return radius;  }  } |

This custom exception class extends java.lang.Exception. The Exception class extends java.lang.Throwable. All the methods (e.g., getMessage(), toString(), and printStackTrace()) in Exception are inherited from Throwable. The Exception class contains four constructors. Among them, the two constructors are often used:

**Java.lang.Exception**

Constructs an exception with no message.

Constructs an exception with the specified message.

**+Exception()**

**+Exception(message: String)**

The constructor in InvalidRadiusException invokes the superclass’s constructor with a message. This message will be set in the exception object and can be obtained by invoking getMessage() on the object.

Now, update the CircleWithException class such that it should use the custom InvalidRadiusException rather than the Java’s IllegalArgumentException. After the update, running the TestCircleWithException class should output the following information:

|  |
| --- |
| InvalidRadiusException: Invalid radius -5.0  Number of objects created: 1 |

**Task # 1**

Suppose that statement2 causes an exception in the following:

|  |
| --- |
| try {  statement1;  statement2;  statement3;  }  catch (Exception1 ex1) {  }  catch (Exception2 ex2) {  throw ex2;  }  finally {  statement4;  }  statement5; |

If no exception occurs, will statement4 be executed, and will statement5 be executed?

|  |
| --- |
|  |

If the exception is of type Exception1, will statement4 and statement5 be executed?

|  |
| --- |
|  |

If the exception is of type Exception2, will statement4 and statement5 be executed?

|  |
| --- |
|  |

If the exception is not Exception1 nor Exception2, will statement4 and statement5 be executed?

|  |
| --- |
|  |

**Task # 2**

Compile and Test the following program:

|  |
| --- |
| import javax.swing.JOptionPane; // program uses JOptionPane  public class Addition  {  public static void main( String[] args )  {  // obtain user input from JOptionPane input dialogs  String firstNumber =  JOptionPane.showInputDialog( "Enter first integer" );  String secondNumber =  JOptionPane.showInputDialog( "Enter second integer" );  // convert String inputs to int values for use in a  // calculation  int number1 = Integer.parseInt( firstNumber );  int number2 = Integer.parseInt( secondNumber );  int sum = number1 + number2; // add numbers  // display result in a JOptionPane message dialog  JOptionPane.showMessageDialog( null, "The sum is " + sum,  "Sum of Two Integers",  JOptionPane.PLAIN\_MESSAGE );  } // end method main  } // end class Addition |

Now modify the above program to take three numbers as input and compute their sum.

**Task # 3**

Write a temperature-conversion application that converts from Fahrenheit to Celsius. The Fahrenheit temperature should be entered by the user and then display the converted temperature. Use the following formula for the conversion:

Celsius = 5/9 × ( Fahrenheit – 32 )

Now also add the Kelvin temperature scale which is calculated by the given formula. The application should also allow the user to make conversions between any two scales. Use the following formula for the conversion between Kelvin and Celsius.

Kelvin = Celsius + 273.15

Also make sure that exceptions that can arise from your code are properly handled as well. GUI can be implemented for this tasks however it is not mandatory.

**Deliverables**

Compile a single word file with code and screenshots of output and upload on LMS.