**Department of Computing**

**CS 212: Object Oriented Programming**

**Class: BESE-7AB**

**Lab 09: Exception Handling and Introduction to GUI**

**Date: April 28, 2017 Instructor:**

**Dr. Muhammad Muneeb Ullah**

**Submitted by: Nimra**

**Registration No. 199281**

**Activity # 1**

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?

|  |
| --- |
| java.lang.IllegalArgumentException: Radius cannot be negative  Number of objects created: 1 |

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?

|  |
| --- |
| Nothing will happen. The code will still run. There will be no compilation error because it is unchecked exception. |

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**

**+**

**Exception**

**()**

**+**

**Exception(message: String**

**)**

|  |
| --- |
| Constructs an exception with no message.  Constructs an exception with the specified message. |

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 TestCircleWithExceptionclass should output the following information:

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

**Activity # 2**

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?

|  |
| --- |
| Statement 4 and Statement 5 both will be executed. |

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

|  |
| --- |
| Yes both statements will be executed because the exception is catch and handled. So program will run. |

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

|  |
| --- |
| Only statement4 will be executed in in finally block. For statement5 it will give an error as ex2 is again passed in the same catch block and is not handled. |

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

|  |
| --- |
| Only statement 4 will be executed in finally block as statement2 error is not resolved. |

# Activity # 3

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.

**The source code is provided.**

# Activity # 4

Compile and Test the following program:

|  |
| --- |
| import javax.swing.\*;    public class HelloWorldSwing {  public static void main(String[] args) {  //Make sure we have nice window decorations. JFrame.setDefaultLookAndFeelDecorated(true);    //Create and set up the window.  JFrame frame = new JFrame("HelloWorldSwing");  frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  //Add the ubiquitous "Hello World" label.  JLabel label = new JLabel("Hello World! How are you? Hopefully, doing Great!");  frame.getContentPane().add(label);    //Display the window. frame.pack();  frame.setVisible(true);  }  } |

Now that you are comfortable with basic GUI examples in Java, lets move towards an application that does something useful!

**Output is:**

