National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS 212: Object Oriented Programming

Class: BSCS-6ABC

Lab 09: Exception Handling and Introduction to GUI

Date: April 24, 2017

Instructor:

Dr. Mian M. Hamayun / Dr. Anis ur Rahman

Learning Objectives

The learning objective of this lab is to understand and practice the concept of Exception Handling and Graphical User Interfaces in Java.

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?

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	
+Exception()	Constructs an exception with no message.
+Exception(message: String)	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 `TestCircleWithException` class 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?

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?

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.

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!

Task # 1

Compile and Test the following program:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class CelsiusConverter implements ActionListener {
    JFrame converterFrame;
    JPanel converterPanel;
    JTextField tempCelsius;
    JLabel celsiusLabel, fahrenheitLabel;
    JButton convertTemp;

    public CelsiusConverter() {
```

```
        //Create and set up the window.
        converterFrame = new JFrame("Convert Celsius to Fahrenheit");

converterFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        converterFrame.setSize(new Dimension(400, 150));

        //Create and set up the panel.
        converterPanel = new JPanel(new GridLayout(2, 2));

        //Add the widgets.
        addWidgets();

        //Set the default button.
        converterFrame.getRootPane().setDefaultButton(convertTemp);

        //Add the panel to the window.
        converterFrame.getContentPane().add(converterPanel,
BorderLayout.CENTER);

        //Display the window.
        //converterFrame.pack();
        converterFrame.setVisible(true);
    }

    /**
     * Create and add the widgets.
     */
    private void addWidgets() {
        //Create widgets.
        tempCelsius = new JTextField(2);
        celsiusLabel = new JLabel("Celsius", SwingConstants.LEFT);
        convertTemp = new JButton("Convert");
        fahrenheitLabel = new JLabel("Fahrenheit",
SwingConstants.LEFT);

        //Listen to events from the Convert button.
        convertTemp.addActionListener(this);

        //Add the widgets to the container.
        converterPanel.add(tempCelsius);
        converterPanel.add(celsiusLabel);
        converterPanel.add(convertTemp);
        converterPanel.add(fahrenheitLabel);


celsiusLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));

fahrenheitLabel.setBorder(BorderFactory.createEmptyBorder(5,5,5,5));
    }
```

```
    public void actionPerformed(ActionEvent event) {
        //Parse degrees Celsius as a double and convert to
Fahrenheit.
        int tempFahr =
(int)((Double.parseDouble(tempCelsius.getText())) * 1.8 + 32);
        fahrenheitLabel.setText(tempFahr + " Fahrenheit");
    }

    public static void main(String[] args) {
        //Make sure we have nice window decorations.
        JFrame.setDefaultLookAndFeelDecorated(true);

        CelsiusConverter converter = new CelsiusConverter();
    }
}
```

Now create a copy of the above program as `UnitsConverter.java` and add a drop-down list (Combo Box) offering multiple conversion options and write code to perform these conversions. Some of the notable conversions are:

- Celsius to Fahrenheit (already done above)
- Meters to Feet
- Kilograms (KGs) to Pounds (LBs)
- Radians to Degrees (Angles)

For more units of conversion, you can consult:

https://en.wikipedia.org/wiki/Conversion_of_units

You may like to consult the Java Tutorials on Swing:

http://docs.oracle.com/javase/tutorial/uiswing/index.html

http://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html

Hand in

Hand in the source code from this lab at the appropriate location on the LMS system. You should hand in a single compressed/archived file named Lab_9_<Your CMS_ID. Your_NAME >.zip (without angle brackets) that contains ONLY the following files.

1) All completed java source files representing the work accomplished for this lab: `CircleWithException.java`, `Addition.java`, `HelloWorldSwing.java`, `CelsiusConverter.java` and `UnitsConverter.java`. The files should contain author in the comments at the top.
2) A plain text file named **README.TXT** that includes a) author information at the beginning, b) a brief explanation of the lab, and c) any comments, or suggestions.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab assignment.
2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.
3. The lab time is not intended as free time for working on your programming/other assignments. Only if you have completely solved the lab assignment, including all challenges, and have had your work checked off for completeness by your TA/Lab Engineer should you begin the programming/other assignments.