# *Programming I (420-B10-HR)*

# *Lab 13 – Graphical User Interfaces*

Date assigned: Tuesday, November 24, 2015

Date due: **Tuesday, November 24, 2015**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will be able to:

1. write a simple Java **JFrame**.
2. use the inheritance principle in Java by extending a class.
3. add GUI objects to an frame.
4. implement an interface.
5. override the **actionPerformed()** method of the **ActionListener** interface.
6. use a wrapper class to convert a **String** to a number.
7. use **String.isEmpty()** to check for data entry in a text field.
8. use the **String.format()** method to format an output report

**To Be Handed In:**

1. The files ***username*\_B10\_L13\_GUI** folder should be uploaded to **Moodle**.

**To Start:**

1. Download and unzip the **B10\_L13\_GUI** folder to your **H:\420-B10\Labs** folder. Rename it to ***username*\_B10\_L13\_GUI**.
2. Start **Eclipse**. Use your **H:\420-B10\Labs** folder as the workspace.
3. Create a **New Java Project** called ***username*\_B10\_L13\_GUI**.

# The LeapYearFrame

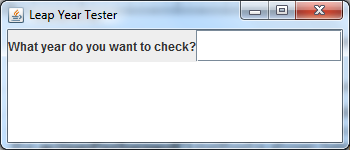
***Purpose:*** Code a Java JFrame.

***To Do:***

## Open **LeapYearFrame.java**. Add the following components:

* a **JLabel** called **lblYear**. It shouldbe initialized to "What year do you want to check?"
* a **JTextField** called **fldYear**. It should be initialized to 10 columns.
* a **JTextArea** called **areaDisplay**. It should be initialized to 5 rows and 10 columns.

## Add **lblYear** to the *West* of the component pane, **fldYear** to the *Center* and **areaDisplay** to the *South*. Your frame should look similar to:



## We want to space out the label and text better. To do this, we will group them in a panel and then add the panel to the center of the content pane:

### Add a private instance **JPanel** variable called **inputPanel**. Instantiate it after the setTitle() statement in the constructor.

### Replace the two statements that add **lblYear** and **fldYear** to the content pane with statements that add **lblYear** and **fldYear** to **inputPanel**. You do not have to specify where they are to be added for a panel.

### Add **inputPanel** to the center of the content pane. Run the frame again. It should look nicer now.

## Try typing in the **areaDisplay** text area. Can you? \_\_Yes.\_\_\_\_\_\_\_\_

## Use **setEditable()** to ensure that the user cannot write in **areaDisplay**.

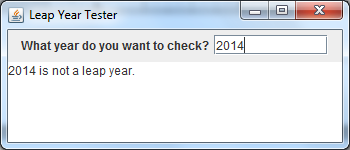
## Add an **actionListener()** to the **JTextField** you created for the year.

## Code the **actionPerformed()** method to display a message indicating whether or not a year that is entered in the **JTextField** is a leap year. The IPO diagram for the **actionPerformed()** method is shown below.

**IPO diagram for actionPerformed()**

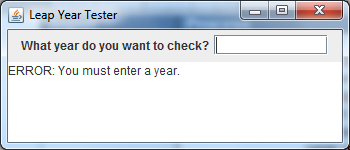
| **Input** | **Process** | **Output** |
| --- | --- | --- |
| **actionEvent e** | Declare numeric year  1. call getText() for fldYear, convert it to an integer[[1]](#footnote-1) and store it in year  2. call Year[[2]](#footnote-2).isLeapYear(year)  3. if the return value from Year.isLeapYear () is true  display a message in the areaDisplay[[3]](#footnote-3) stating that year is a leap year  else  display a message in the areaDisplay stating that year is not a leap year |  |

## Test your program. The completed frame will look similar to:



## Run the program again. Hit enter in the empty text field. (Do not enter a year first.) What happens?

## We want to make sure that there is data in the field before converting it. To do that use **getText()** to get the **String** contents of the text field and then test it with the **String** **isEmpty()** method. If the field is empty print an error message:



# Changing a User Interface

***Purpose:*** Convert a command line interface to a GUI interface.

***To Do:***

## Open **ArithmeticTables.java**. Run it.

## You can convert the **ArithmeticTables** command line interface to a GUI interface. To convert the interface:

### Create a new class called **ArithmeticTablesFrame**. Include a constructor, a main method and comments.

### Add the following description to the comments at the beginning of the program – "Display different arithmetic tables".

### Your class should extend **JFrame**.

### Add a statement to the constructor to set the title of the frame to "Arithmetic Tables".

### Add the following GUI components to your frame:

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Name** | **Initial value** | **Size** |
| JButton | btnSquaresAndCubes | "Squares and Cubes" |  |
| JButton | btnAddition | "Addition Table" |  |
| JTextArea | areaDisplay |  | 15,45 |
| JPanel | buttonPanel |  |  |

### Add the two JButtons to the **buttonPanel**:

### Add the **buttonPanel** to the north of the content pane.

### In the **main()** method, instantiate an **ArithmeticTablesFrame** object and add the statements necessary to set the size to (450,350), location to (100,100), exit on closing and make it visible.

### Run **ArithmeticTablesFrame**.

### Add the **areaDisplay JTextArea** in the center of the content pane. Run the frame again.

### Use **setEditable()** to ensure that the user cannot write in **areaDisplay**.

## Add the code to display the appropriate table when a button is clicked:

### Add the statements necessary to implement an action listener for the two buttons.

### Copy the two methods that display the tables (**displaySquaresAndCubes()**, **displayAdditionTable()**) from the **ArithmeticTables** class. Place them after the end of the **actionPerformed()** method.

### Delete the word **static** from each of the method headers of the methods you just copied and make the methods private.

### In the **actionPerformed()** method:

#### Add an if-else statement to determine which of the two buttons was clicked and execute the appropriate method. To determine which button was clicked, compare the button name with **e.getSource()**. (e.g. **if (e.getSource() == btnSquaresAndCube)** )

### Run the program. Where is the output display when a button is clicked? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Change the first **System.out.printf()** statement in each of the three display method to **areaDisplay.setText()**. You will get an error because the **setText()** method only takes one parameter. The **String format()** method can be used to format the string before displaying it. The **format()** method has the same parameter list as the **printf()** statement so all you have to do is add **String.format(** before your format string and add an extra **)** before the semi-colon. For example the first **printf()** for the **displaySquaresAndCubes()** method would be:

**areaDisplay.setText(String.format("%25s\n",**

**"Squares and Cubes"));**

### Change the rest of the **printf()**'s in each method to **areaDisplay.append()**. Use the **String format()** method to keep the same formatting. (e.g. The **printf()** in the loop of **displayAdditionTable()** will be:

**areaDisplay.append(String.format("%5d", row + col));**

### Change any **System.out.print()** or **System.out.println()** statements to equivalent **display.append()** statements.

## Run **ArithmeticTablesFrame** to test your changes.

## Notice that the text area doesn't display the entire table. We can drag the edges to expand the frame to be big enough, but we want to be able to scroll up and down and left and right as well. To do this, change the statement that adds the text area to the content pane to:

**getContentPane().add(new JScrollPane( areaDisplay), BorderLayout.CENTER);**

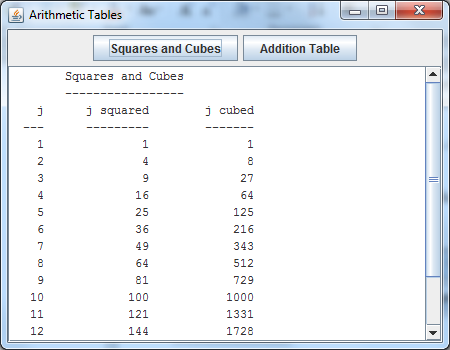
## It's still not perfect. The columns are not correctly lined up. This is because the output font for a frame is variable-spaced – that is each letter takes up only the amount of space required. For the **String format()** method to work properly, the output must be monospaced. Import **java.awt.Font** and add the following in the constructor after **areaDisplay** has been instantiated.

**areaDisplay.setFont(new Font("Monospaced",Font.PLAIN,12));**

## Run the program again.

## Add an append statement to display lines under the column headings for the Squares and Cubes display.

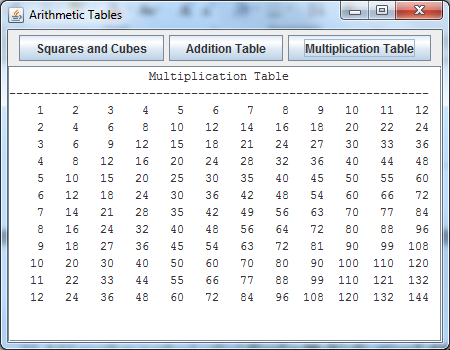
## Run the program again. Your output should look like:



## Add another method called **displayMultiplicationTable()**. It should display the multiplication table for 1 to 12.

## Add a button for "Multiplication Table". Add it to the panel. Add an action listener for it and modify the **actionPerformed()** method to call **displayMultiplicationTable()** when the button is clicked.

## Test your changes. If you click the Multiplication Table button, your output should look like:



1. You will have to use the **Integer.parseInt()** method to convert the **JTextField** data from **String** to integer. [↑](#footnote-ref-1)
2. You do not need to create an object of the **Year** class to call the **isLeapYear()** method because it is a static method. To call it code **Year.isLeapYear(num)**. [↑](#footnote-ref-2)
3. To display text in a JTextArea, use the **setText()** method. [↑](#footnote-ref-3)