# CSC 205 Lab 8 : GUI Programming with the AWT

## Goals

After completing this lab, you should be able to:

* Write GUI applications using AWT components.
* Know how to implement action listeners to handle events.

## Lab Startup

Change into your Labs directory, and let's create and change into a Lab8 directory.

Now, let's copy over some files by typing : cp /pub/digh/CSC205/Lab8/\* .

**My First Interactive GUI**

1. Compile and execute the GUI application (MyGUI.java). Play with it and make sure that the button and text field will not respond to mouse clicks. Check the source file, and determine which Java statements are being used to generate the GUI layout. Which layout manager class is being used? How does it set up the components within the window? What lines of code are needed to set it up?

2. Remove the comment signs to bring the code segments for the ButtonListener and WindowCloser classes to life. Make additional changes as follows:

1. Create a ButtonListener object and attach it to the button b. You can use one line of code as follows.

b.addActionListener(new ButtonListener());

1. Create a WindowCloser object and attach it to the MyGUIFrame. You can use the addWindowListener() method of the Window class, which is the direct superclass of the Frame class.

addWindowListener(new WindowCloser());

c. What types of components are going to be added by the MyGUIFrame constructor?

d. What steps are being performed by the ButtonListener object?

3. Compile and execute the GUI application again. Test the enter button and the close window button (X) on the frame. Verify your answers from above.

4. Replace the f.setSize() method in the driver class (MyGUI) with the f.pack() method. Compile and execute the GUI application again. Do you notice any differences from the previous frame? Write down your explanation below.

## A Simple Phone Dial Pad

|  |  |
| --- | --- |
|  | We’ve learned how to add Component objects to a Container such as a Frame. In addition, we’ve learned how to add an ActionListener to catch and respond to events. Now, let’s play with a Frame with more buttons to simulate a phone dial pad.  This frame contains a text field, twelve buttons (corresponding to the twelve buttons on a Touchtone phone), and a "Dial" button. When the user presses a numbered button, the corresponding digit will be added to the text field, so that the number being dialed is gradually spelled out. (Pressing the "\*" and "#" buttons has no effect.) When the user presses the "Dial" button, the text field will display the message "Dialing..." for two seconds. The text field will then go blank, and the user can enter another phone number. |

1. Take a look at the source code for Dial.java. Answer the following three questions.

* 1. What LayoutManager is used in the buttonPanel to make sure the numbered keys are displayed in a 4 by 3 mesh?

* 1. A for loop was used to generate and add keys 1 through 9 into the pad. Why were the numbered keys displayed row-wise, instead of column-wise?
  2. What LayoutManager is used to put the phoneNumber, centerPanel, and bottomPanel into desired places?

2. To make the pad work, complete the following eight exercises. Each is labeled within your source code (e.g., \*\* 1\*\*).

1. Create a DigitListener object for all numbered keys (i.e., right after the line with \*\*1\*\*.) That is, DigitListener dl = new DigitListener();

Why is it possible that only one listener is enough for 10 different digits?

1. Add the listener reference created in part a. to each button. buttonPanel.add(b);
2. Add the listener reference for the button ‘0’. b.addActionListener(dl);
3. Create and add a DialListener object to the dial button.

b.addActionListener(new DialListener());

1. Implement the DigitListener class to modify the phone number when a new digit is pressed. Include the previous numerical string with the current button label. phoneNumber.setText(phoneNumber.getText()+buttonLabel);

3. Compile your program. If there are no errors, execute your class file for viewing.

4.Now, let’s edit the Dial2 program. Here, we are using an array to hold up to ten recently used numbers. We are simulating a type of “Caller ID.”

Here’s what you need to do :

a. Replace the “\*” and “#” keys by “>” and “<” to retrieve the next or previous numbers, respectively at the portion of the program labeled \*\*1\*\* . The “>” button should be added prior to the “0” button, and the “<” button should be after it. You will need to add an arrow listener object to both.

Notice that we have added a new listener class, ArrowListener, and a method named addPastNumber. In private, we have an array that is holding the numbers that have been recently dialed. Numbers currently in the array will not be added again. When the array is exhausted, the next new number will be added to the first element in the array. Also, the most recently dialed phone number will be used to locate the currently position in the array.

b. Now, move to the portion of the program labeled \*\*2\*\* within the ArrowListener. If the current index within our pastNumbers array is 9, the number stored at index 0 will be displayed if the “>” key is pressed. If the current index is 0, the number stored at index 9 will be displayed if the “<” key is pressed.

To set this up, you’ll need to retrieve the label from your button. If it is the “<” key, set your arrow index to :

(arrowIndex + pastNumberCount - 1) % pastNumberCount

and add the past number at this index in your array of past numbers to your phoneNumber text field.

Otherwise, set your array index to : (arrowIndex + 1) % pastNumberCount,

and add the past number at this index to your text field.

Compile and execute your finished product.