GUIs: Graphical User Interfaces

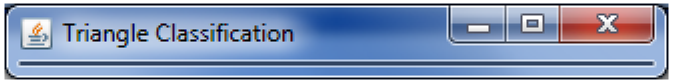
- Objectives - when we have completed this topic, you should be familiar with:
  - frames
  - panels
  - nested panels
  - text fields
  - buttons
  - event listeners

GUI Overview

- All of the programs that you have written so far have been designed to use standard input and standard output
- Although all programs have similar underlying mechanics such as classes, methods, and control structures (e.g., if-else, loops, etc.) most have a graphical user interface (GUI)
- Your programs run fine from the command prompt (or Run I/O tab in jGRASP), but chances are that when you use a program (e.g., web browser or IDE), you are working with a GUI

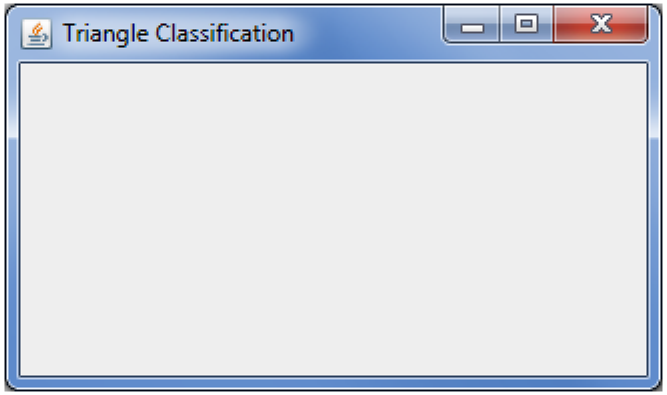
Frame: Review

- A frame is the part of the GUI that represents the window that can be displayed. It includes a title bar and can be resized.
- A frame is GUI component called a container
- Below is an empty frame. The frame will look different on different operating systems.



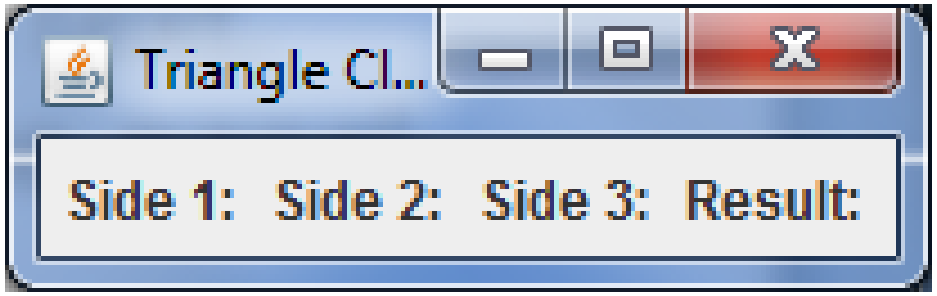
Panel: Review

- A panel is a GUI component that can be placed inside of another container (frame or panel) and can hold other components.
- If a frame is a window, then think of a panel as a piece of glass. You can stick things to it, but it is invisible. Below a panel of a certain size has been added to the frame.



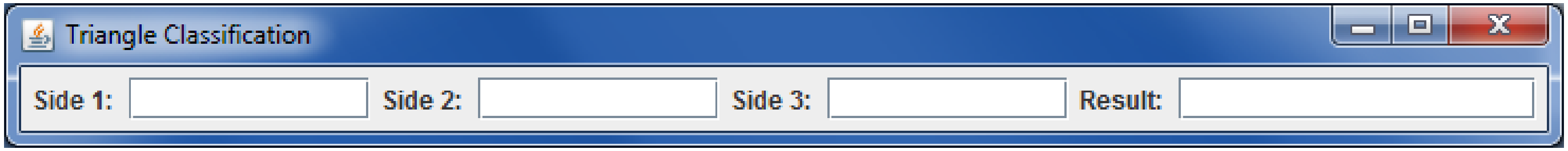
Label

- A *label* is a GUI *component* that holds text.
- Below, four labels have been added to a panel, then the panel has been added to the frame.



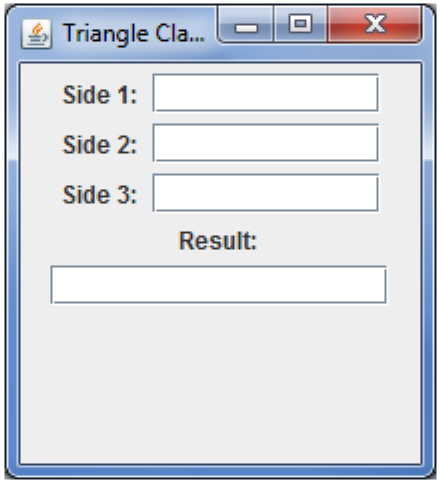
Text Box

- A *text box* is another GUI *component* that holds text. It allows users to modify the text within it. It can be used to:
  - Get user input
  - Display output



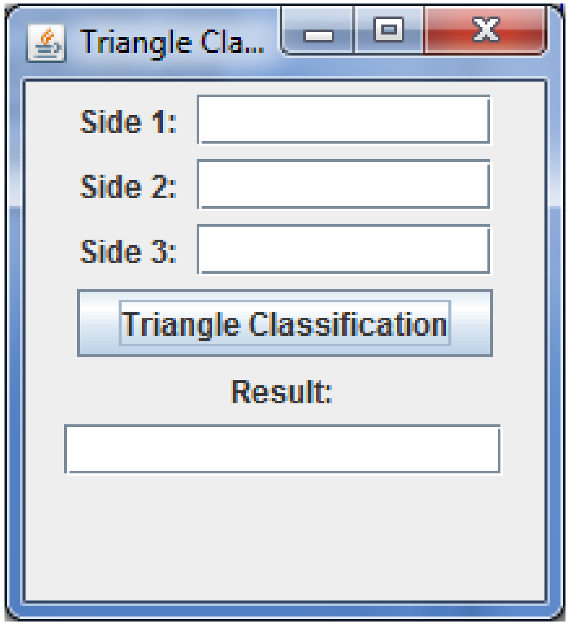
Nested Panels

- Notice that components are added left to right. In order to organize our components, we will first resize the panel to 200 x 200 pixels:



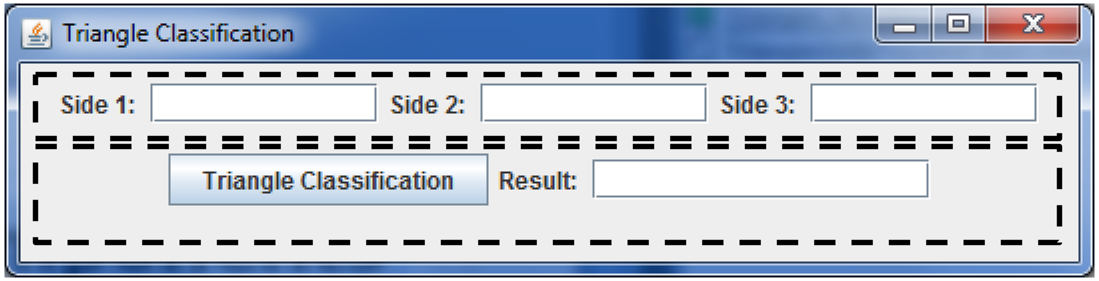
Buttons

- Buttons are another type of component that are added to containers. Below, a button has been added to the lower nested panel:



Nested Panels

- You can also add panels inside of other panels. For organizational purposes, we will add 2 different panels; one for the inputs and one for the results:
    - The panels inside of the main panel are invisible, but they are outlined below.



Overview

- Each part of a GUI is represented by a class in Java.
- The javax.swing package contains these classes:
  - **JFrame**: the frame of the GUI
  - **JPanel**: the panels inside of the frame and nested panels
  - **JLabel**: labels
  - **JTextField**: text field
  - **JButton**: buttons

GUI Creation

- First, you will create a class that represents the outer panel. It will extend JPanel, meaning that it is a panel itself (you’ll learn more about this later):
  - You’ll also need to add imports for all of the classes that you will use.

```
public class TriangleGUI extends JPanel {



}
```

GUI Creation

- Now you will create a Driver program that creates a JFrame and adds an instance of your panel to the frame.
  - Code for creating the pane and adding the panel:

```
JFrame frame

   = new JFrame("Triangle Classification");

TriangleGUI guiPanel = new TriangleGUI();

frame.getContentPane().add(guiPanel);
```

TriangleClassifier.java

Adding Nested Panels

- You will add all components to the outer panel in its constructor. For now, declare and instantiate the upper and lower panels:

```
public class TriangleGUI extends JPanel {

    public TriangleGUI() {

    JPanel upperPanel = new JPanel();
    JPanel lowerPanel = new JPanel();

    }

}
```

Instance Variables

- Add as instance variables all of your components. Recall that the GUI has 4 labels, 4 text fields, and a JButton.

```
private JLabel side1, side2, side3,
     resultLabel;

private JTextField side1Input,

     side2Input, side3Input, output;

private JButton classifyButton;
```

TriangleGUI.java

Instantiating Components

- Each component will now need to be instantiated in the constructor of the class:
  - JLabels are instantiated with the text that they will hold as the parameter:

    ```
    side1 = new JLabel("Side 1: ");
    ```

  - JTextFields are instantiated with the number of columns (their width in characters):

    ```
    side1Input = new JTextField(10);
    ```

  - JButtons are instantiated with the text that they display:

    ```
    classifyButton = new JButton("Triangle Classification");
    ```

Adding Components

- In order to show your components, they need to be added to a panel. Invoke the add method in JPanel to add the components to the panel:

```
upperPanel.add(side1);

upperPanel.add(side1Input); // etc
```

- You can refer to the outer panel (the TriangleGUI object itself), using the this reserved word to add the nested panels:

```
this.add(upperPanel);
this.add(lowerPanel);
```

Action Listeners

- Now you will be able to see your components, but nothing will happen when you press buttons or enter text.
- Every time you press a button, enter a letter into a text field, or click the mouse, an *event* occurs. Most events are ignored (imagine if something happened every time you moved the mouse)
- In order to do something when a useful event occurs, you must add an event *listener*.
- Classes needed:

```
import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;
```

Action Listeners

- In our case, you want to add a listener to the JButton. First, create a class inside of the TriangleGUI. The class can have any name that you want, but it must implement ActionListener and must contain an actionPerformed method:

```
private class ClassifyListener implements ActionListener {

   public void actionPerformed(ActionEvent event) {


   }
}
```

Action Listeners

- Inside of the action performed method, you must determine the source of the event using the getSource method of the incoming event:

```
if (event.getSource() == classifyButton) {

}
```

Variable name for the Jbutton.

- If you had multiple buttons, then you would need an if to handle each button. That way you can determine what button was pressed.

Action Listeners

- Once you determine the source of the event, you can determine what happens when the button is pressed.

- In the triangle classification program, you want to get the text from each field, convert it to a double, create a Triangle object, and place the classification in the result text box.
  - Use the getText and setText methods on your JTextFields. You can look up other methods in the Java API documentation for JTextField.
  - See line 55 of the TriangleGUI.java class.

Action Listeners

- Finally, you’ll need to “connect” the JButton and the action listener in your constructor. See lines 31-32 of TriangleGUI.
- Create an instance of the action listener:

```
ClassifyListener buttonListener

    = new ClassifyListener();
```

- Add the action listener to the JButton:

```
classifyButton.addActionListener(buttonListener);
```

GUIs: Wrap Up

- Run TriangleClassifier.java to get an idea of how it works. You'll be creating a class this week and you will be asked to complete GUI code that interacts with that class. Much of the code will be provided, but you will need to understand the basics of GUIs.

- The best way to learn about GUIs is by experimenting with the code. Try adding another button to the triangle GUI (perhaps one to clear the text fields). Also see the end of Chapter 4 and 5 in the book for more examples.