



# Java Lesson 5

Concept Review




# Swing

- Applications are more enjoyable for people to use when they have a GUI (Graphical User Interface)!
- If you think about it, how many applications do you find being sold at the local store that run solely at the command prompt? Not very many of them, which is one reason why it's important to know how to create GUI-based applications. Your users will greatly appreciate it!
- To give developers the ability to create GUI-based programs, Oracle provides a powerful feature in Java known as [Swing](#).



# The JFrame

- The first step to creating a graphical application is to use a JFrame.
- The JFrame is a fancy term for basically a “window” in which you can later add various graphical components to it, such as buttons, pictures, etc.
- One of the most common ways to create a JFrame is to **extend** it, as shown in Figure 14-15 on page 657.



```
import javax.swing.*;
public class JMyFrame extends JFrame
{
    final int WIDTH = 200;
    final int HEIGHT = 120;

    public JMyFrame()
    {
        super("My frame");
        setSize(WIDTH, HEIGHT);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

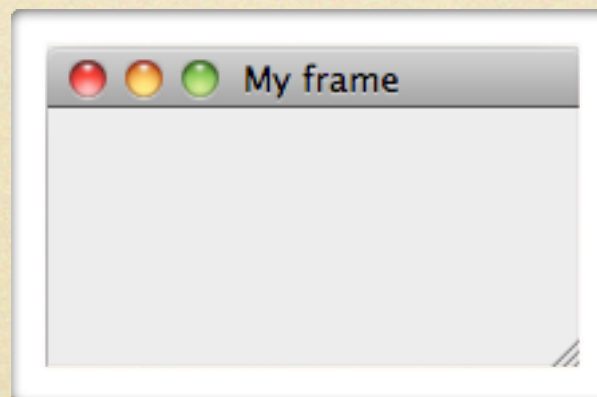


# The JFrame

- To display a JFrame that has been extended, simply instantiate the class as an object, as shown:

```
public static void main(String[] args)
{
    JMyFrame myFrame = new JMyFrame();
}
```

- When the program is run, the JFrame displays as it should.





# Swing Components

- Some of the most common components that you'll use within the GUI apps that you create will be:
  - **JLabels** - These allow you to display a line of text.
  - **JTextFields** - Allow the user to input a line of text.
  - **JButtons, JComboBoxes and JCheckBoxes**
    - Most of these components are self-explanatory and are covered within the textbook's assigned reading and online lesson.



# Event Listeners


- Some of the components that you'll utilize in this lesson will have the ability to respond to a user's actions, such as the clicking a user's mouse or the pressing of a key on their keyboard. These are known as [events](#).
- To enable a Swing component, such as a JButton or a JTextField to detect and respond to a user's events, you need to assign an [ActionListener](#) to them.
- Check out the following code:



# ActionListener

- Notice that this program implements an `ActionListener` in the class declaration. This gives your program the ability to process a user's events, such as the clicking of a mouse button.
- To add an `ActionListener` to a component, use the `addActionListener()` method.

Adds the  
`ActionListener` to  
the `clickMe` button  
component.



```
public class JMyFrame extends JFrame implements ActionListener
{
    JButton clickMe = new JButton("Click Me");

    public JMyFrame()
    {
        super("My frame");
        setSize(250, 100);
        setLayout(new FlowLayout());
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        add(clickMe);
        clickMe.addActionListener(this);

        setVisible(true);
    }
}
```

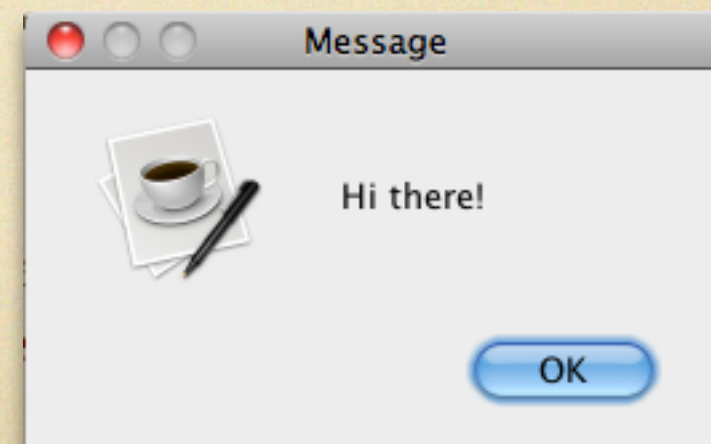
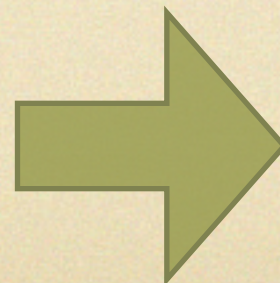
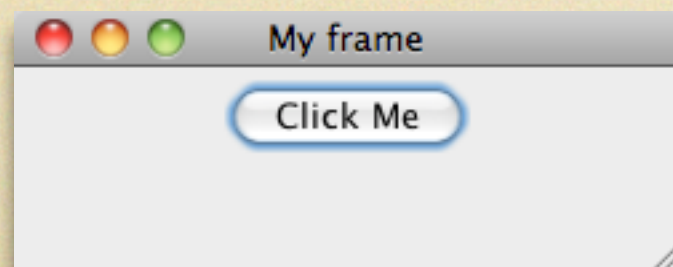


# ActionPerformed

- To determine which component in the JFrame was triggered by the user, you must include a method called `actionPerformed()`. Here's an example:

```
public void actionPerformed(ActionEvent e)
{
    JOptionPane.showMessageDialog(null, "Hi there!");
}
```

- This method **automatically runs** the moment that the user triggers an event, such as clicking on the `clickMe` button.





# ActionPerformed

- Sometimes, you might have more than one component in a JFrame that can trigger a user event. If this is the case, you need a way to determine which component the user utilized, that triggered the event.
- The easiest way to do this is to use the `getSource()` method, as demonstrated in Figure 14-27 on page 668 in the textbook.
- Notice how an `if()` statement is used to determine which component triggered the event.