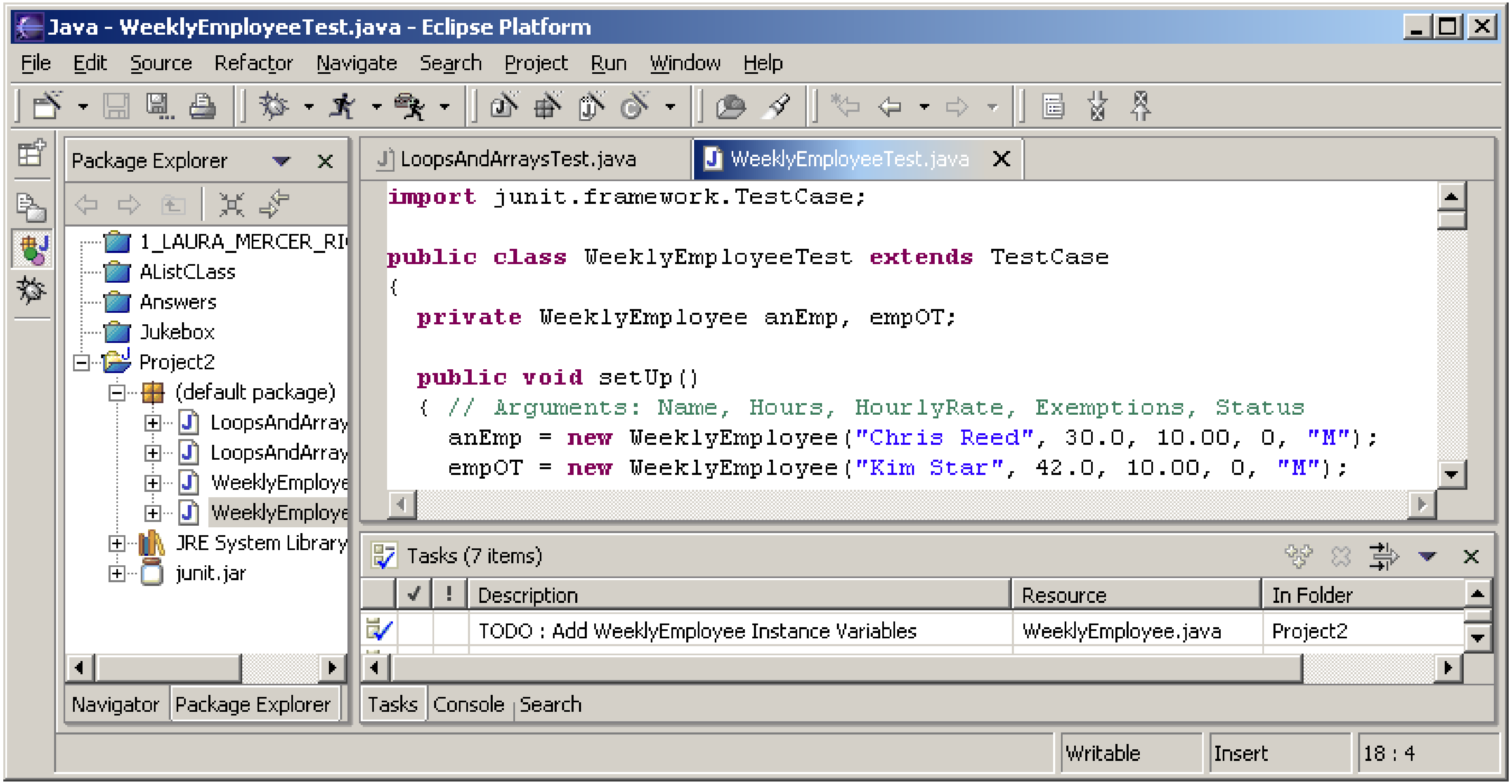
Event Driven Programming with Graphical User Interfaces (GUIs)

A Crash Course

Graphical User Interfaces (GUIs)

- Most apps provide services through GUIs



```
import junit.framework.TestCase;

public class WeeklyEmployeeTest extends TestCase
{
  private WeeklyEmployee anEmp, empOT;

  public void setUp()
  { // Arguments: Name, Hours, HourlyRate, Exemptions, Status
    anEmp = new WeeklyEmployee("Chris Reed", 30.0, 10.00, 0, "M");
    empOT = new WeeklyEmployee("Kim Star", 42.0, 10.00, 0, "M");
```

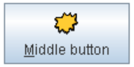
	✓	!	Description	Resource	In Folder
✓			TODO : Add WeeklyEmployee Instance Variables	WeeklyEmployee.java	Project2

A Few Graphical Components

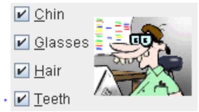
- To build an event-driven program with a graphical user interface (GUI)
  - Begin with a well-tested model such as
    - GameOfLife, GameTree, MineSweeper, Boggle
  - Layout graphical components to allow user interaction
  - Ensure the correct things happen for each ***events*** triggered by user interaction
    - user clicks a button, moves a mouse, presses the enter key, adjusts a slider bar, presses the escape key, clicks the mouse. Select a menu option, ...

A Few Graphical Components

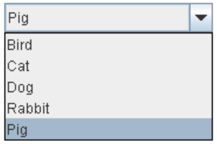
- Java's graphical components include
  - windows, buttons, text areas, menus, dialog boxes, tables with sortable columns, trees to expand folders, tabbed panes, images, ...http://web.mit.edu/6.005/www/sp14/psets/ps4/java-6-tutorial/components.html



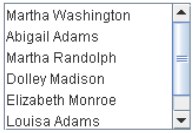
JButton



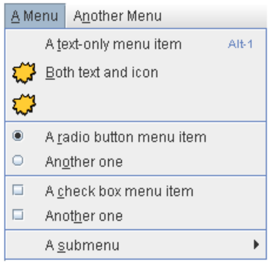
JCheckBox



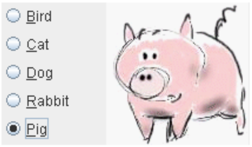
JComboBox



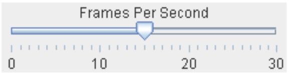
JList



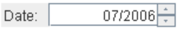
JMenu



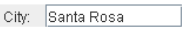
JRadioButton



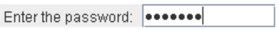
JSlider


Date: 07/2006

JSpinner


City: Santa Rosa

JTextField


Enter the password: •••••••

JPasswordField

A few types from the swing package

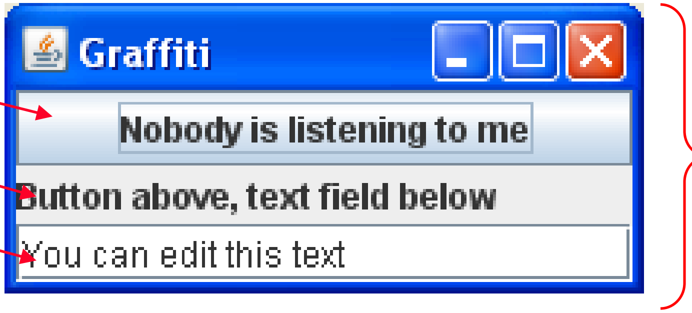
- The **`javax.swing`** package has many graphical components

`JFrame`: window with title, border, menu, buttons

`JButton`: A component that can "clicked"

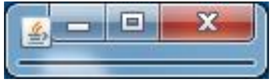
`JLabel`: A display area for a small amount of text

`JTextField`: Allows editing of a single line of text



Get a "window" to show itself

- Have a class extend javax.swing.JFrame
  - Your new class inherits all the methods and instance variables for putting a window on the computer
- Add a main method to call the constructor
- Layout Set up the GUI in the constructor
  - See program on the next slide



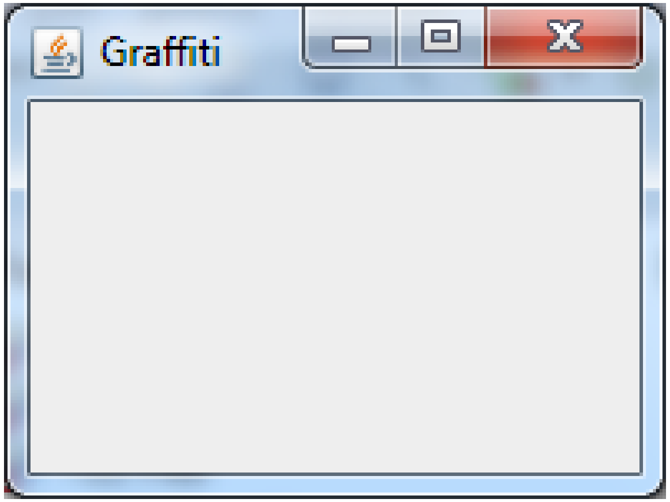
Can you see the window?

```
import javax.swing.*; // Lot of GUI components now visible
public class FirstGUI extends JFrame {

  public static void main(String[] args) {
    JFrame aWindow = new FirstGUI();
    // Tell the JFrame object to show itself
    aWindow.setVisible(true);
  }

  public FirstGUI() {
     this.setTitle("Graffiti");
     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
}
```

Some JFrame messages

- Set the size and location of the window with

```
this.setSize(210, 175); // 200 pixels wide, 150 pixels tall
this.setLocation(75, 40); // Upper left corner at 75 40
```



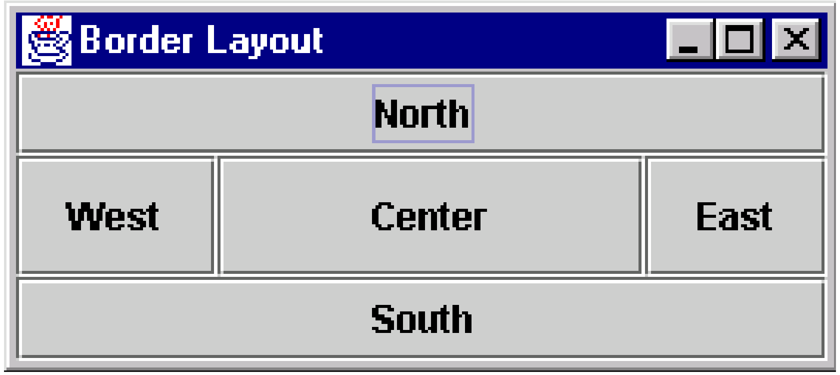
Building Graphical Components

- Add these three graphical components to the "content pane" of the JFrame

```
private JLabel aLabel =
       new JLabel("Change with setText(String)");

private JTextArea textEditor =
       new JTextArea("You can edit this text ");

private JButton clickMeButton =
       new JButton("Nobody is listening to me");
```

Using Null Layout

- By default, **`JFrame`** objects have only five places where you can add components *a 2nd add wipes out the 1st*



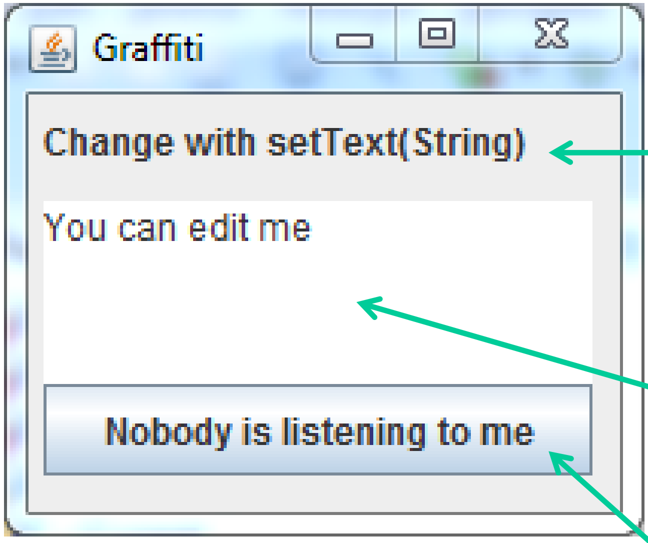
- There are many layout strategies, we'll use null layout

```
    setLayout(null);
```

  - State there is no object to decide where components go
  - We must set each component's size and location

Add components to a window

- Add the previously constructed components to the center part of the **`JFrame`**



You can edit me

```
this.setLayout(null);
aLabel.setSize(180, 30);
aLabel.setLocation(5, 0);
add(aLabel);

textEditor.setSize(180, 60);
textEditor.setLocation(5, 35);
add(textEditor);
clickMeButton.setSize(180, 30);
clickMeButton.setLocation(5, 95);
add(clickMeButton);
```

Other useful methods

- `String` **`getText()`** get the text stored in components **`JButton JTextArea TextField`**
- **`void setText(String str)`** changes the text stored in **`JButton JTextArea JTextField`**
- Two messages wrap at the word level in a **`JTextArea`** *default is NO wrapping*

```
textEditor.setLineWrap(true);
textEditor.setWrapStyleWord(true);
```

Other useful methods

```
textEditor.setBackground(Color.CYAN);

textEditor.setFont(
        new Font("Courier", Font.BOLD, 24));
```

So what happens next?

- You can layout a pretty GUI
- You can click on buttons, enter text into a text field, move the mouse, press a key
  - And NOTHING happens

- So let’s make something happen …

Java's Event Model

- Java and the operating system work together to detect user interaction
  - `JButton` objects are notified when clicked
  - `JTextField` and `JTextArea` objects with focus know when the user presses the enter key
  - A `MouseListener` object knows when the mouse is clicked, pressed, released, or enters/exits a component *MouseListener is an interface*
  - A `HyperLinkEventListener` object knows when the hyperlink has been clicked *HyperLinkEventListener is an interface*

Example: Action Events

- The buttons, text fields, and menu items do not perform the actions
  - Instead **`JButton`**, **`JTextField`**, **`JMenuItem`** objects send **`actionPerformed`** messages to other objects
- The code that responds to events is often located in **`actionPerformed`** methods
- We need to have a class that implements the **`ActionListener`** interface to guarantee the object has an **`actionPerformed`** message

How to Handle Events

- Add a private inner class that can listen to the event that the component generates
  - Your class must implement a listener interface to guarantee that it has the expected methods: First up: **`ActionListener`**
- Add the listener so the component can later send **`actionPerformed`** messages to the listener
  - Events occur anytime in the future--the listener is listening (waiting for user generated events such as clicking a button or entering text into a text field)

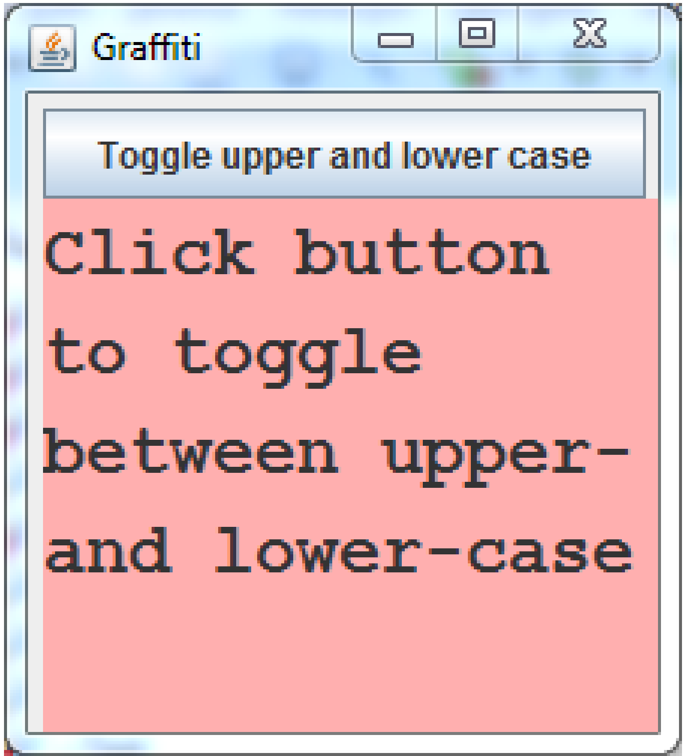
Have a class that **`implements ActionListener`**

```
// Inner class to listen to events
private class ButtonListener implements ActionListener {
  // clickMeButton will call this method once registered (below)
  public void actionPerformed(ActionEvent anActionEvent) {
     // Show contents of the JTextArea
    JOptionPane.showMessageDialog(null, textEditor.getText());
  }
}


  // In the constructor of the class that extends JFrame,
  // register the instance of the listener so the component
  // can later send messages to that object
  ButtonListener aListener = new ButtonListener();
  clickMeButton.addActionListener(aListener);
```

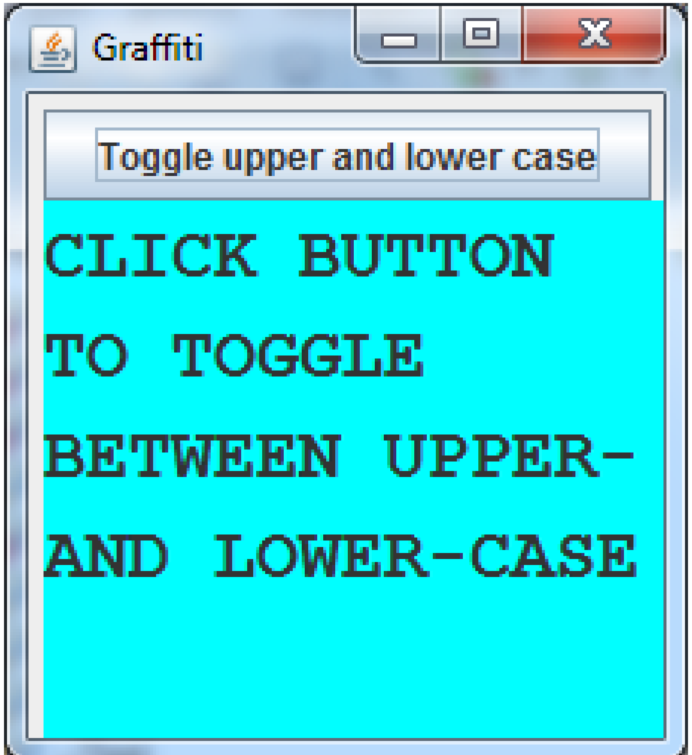
Caution: This is easy to forget.
It is a bug, not a compile time error

Active Collaborative Learning

- In teams of 2 or 3, on one computer, implement an event-driven program with a GUI to toggle text and colors in a JTextArea with button clicks. Match the GUI



Click button
to toggle
between upper-
and lower-case



CLICK BUTTON
TO TOGGLE
BETWEEN UPPER-
AND LOWER-CASE