PRG 421
Java Programming II

Rosa Williams
Facilitator

Class Topics

Review

Week One: User Interfaces

Week Two: GUI Components

Week Three: Files

Week Four: Applets and Graphics

Week Five: Database Connectivity and Mobile Computing

Week One: User Interfaces

	Details	Due	Points
Objectives	1. Design, implement, test, and debug a Java® program that incorporates a graphical user interface (GUI) and processes events. 2. Apply layout managers.		
Learning Team Instructions Fundraiser Program	**A city is sponsoring a run to support local charities and would like an application to track the pledges. The result will be a database that holds data on individuals, total pledges obtained, and the charity for which the donation is designated.** **Design and implement a GUI-based program to accept a participant's name, the amount pledged, and the designated charity's name. The program will output this information to the GUI. The project will be completed in several stages, with the first deliverable due in Week Two.**		
Individual Review Exercise	For week one the assignment is to complete the quiz located in the CourseMaterials Tab. The posting is entitled "Week One Java Concepts Quiz". This should be completed and submitted to the Assignment Tab.	Assignment Tab Tuesday, November 26, 2013 6:00 PM	5

Class Topics

- Week One: User Interfaces
- ➡ Week Two: GUI Components
- Week Three: Files
- Week Four: Applets and Graphics
- Week Five: Database Connectivity and Mobile Computing

Week Two: GUI Components

	Details	Due	Points
Objectives	Design, implement, test, and debug a GUI program that includes lists, combo boxes, and menus.		
Reading	Read the Week Two Read Me First.		
Reading	Read Ch. 12, "GUI Applications – Part 2," of Starting Out With Java.		
Video	Watch the Deitel video, "Figure 14-21-22: ComboBoxFrame.java."		
Participation	Participate in class discussion.	In class Tuesday, December 3, 2013	2
Supporting Activity Individual Graded In-Class Activity	This will be an in-class activity reviewing the week's discussion.	Assignment Tab or Submit to Facilitator Tuesday, December 3, 2013 10:00 PM	2

Week Two: GUI Components			
Learning Team Instructions Learning Team Charter	Create a Learning Team Charter and post it in the Learning Team forum.		
Learning Team Initial Project Plan	Develop a project plan for the fundraiser program due in Week Five. • The project plan should describe the layout of the GUI. • The project plan should also include individual task assignments. Submit the project plan to your facilitator.	Assignment Tab Tuesday, December 3, 2013 6:00 PM	5

Week Two: GUI Components			
Individual Hello World Program	Design, implement, test, and debug a GUI-based version of a "Hello, World!" program. Create a JFrame that includes two JLabels. One should reads "Hello, World!" The other should read "Week 2 Assignment". The title of the frame should be your name. Use a layout manager of your choice. Include an Exit button to close the program. Submit the .java source code files with the correct file names for all classes used. The program must consist of at least one public class. Programs are expected to compile and run/execute correctly. There should be proper documentation (comments) in the source code. The documentation should include a block containing the name of the program and the name of the student, and a short description of the program.	Assignment Tab Tuesday, December 3, 2013 6:00 PM	5

Questions?

Week II Topics

- Displaying Text in a Dialog
- Preview of Week 2
- Graded In Class Exercise

Week Two: GUI Components

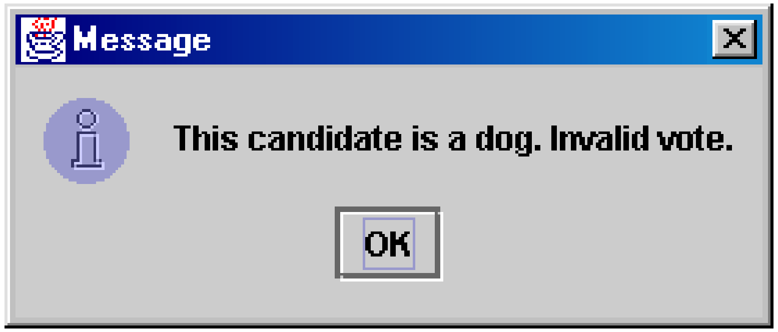
Displaying Text in a Dialog

Displaying Text in a Dialog Box

- Windows and dialog boxes
  - In PRG 420 output has been to the console
  - Many Java applications use dialog boxes to display output
  - Package is javax.swing – contains classes for creation of GUIs.
  - `JOptionPane` class provides prepackaged dialog boxes called message dialogs
  - showMessageDialog() – displays a dialog box displaying a message. Two parameters , position of the dialog box and the message.

JOptionPane

An option pane is a simple dialog box for graphical input/output



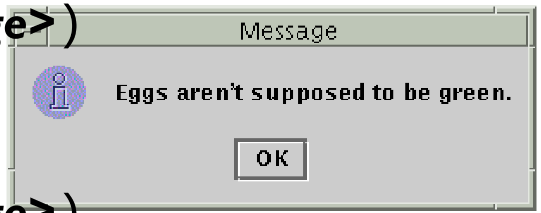
This candidate is a dog. Invalid vote.

- advantages:
  - simple
  - flexible (in some ways)
  - looks better than the black box of death

- disadvantages:
  - created with static methods; not very object-oriented
  - not very powerful (just simple dialog boxes)

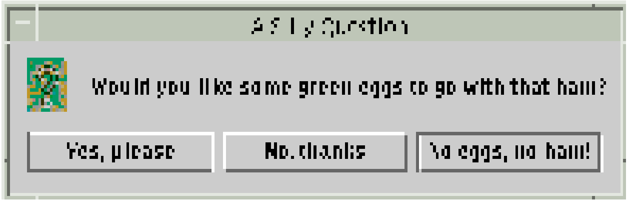
Types of `JOptionPane`s

- `showMessageDialog(`***<parent>***`,` ***<message>***`)`
  Displays a message on a dialog
  with an OK button. – Using in this class



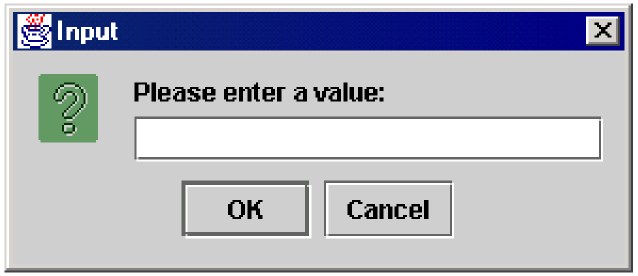
Eggs aren't supposed to be green.

- `showConfirmDialog(`***<parent>***`,` ***<message>***`)`
  Displays a message and list of
  choices Yes, No, Cancel;
  returns user's choice as an `int` with one
  of the following values:
  - `JOptionPane.YES_OPTION`
  - `JOptionPane.NO_OPTION`
  - `JOptionPane.CANCEL_OPTION`



Would you like some green eggs to go with that ham?

- `showInputDialog(`***<parent>***`,` ***<message>***`)`
  Displays a message and text
  field for input; returns the user's
  value entered as a `String`.



Please enter a value:

- Can pass null for the parent to all methods

Displaying Text in a Dialog Box

- **showMessageDialog()** – displays a dialog box displaying a message. Two parameters , position of the dialog box and the message.

- **showMessageDialog()** is a class method of JOptionPane class , it is a static method.

- Such methods often define frequently used tasks that do not explicitly require the creation of an object.

- A static method typically is called by using its class name dot operator and the method name.

```
JOptionPane.showMessageDialog( null, "Welcome\nto\nJava" );
```

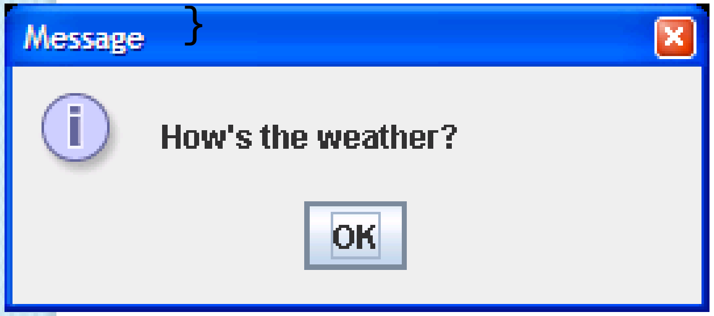
- Note we did not have to create an instance of the class JoptionPane to use the showMessageDialog() method.

JOptionPane

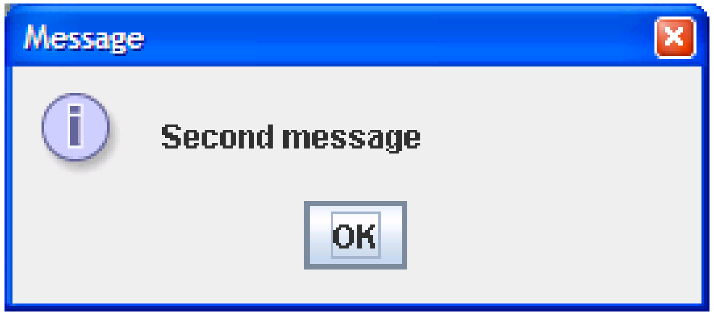
- `showMessageDialog` analogous to `System.out.println` to display a message

```
import javax.swing.*;

public class MessageDialogExample {
    public static void main(String[] args) {
        JOptionPane.showMessageDialog(null,"How's the
  weather?");
        JOptionPane.showMessageDialog(null, "Second
  message");
    }
}
```



How's the weather?



Second message

Week Two: GUI Components

Events

Event-Driven Programming

- *Procedural programming* is executed in procedural order.

- In *event-driven programming*, code is executed upon activation of events.

Events

- An *event* can be defined as a type of signal to the program that something has happened.

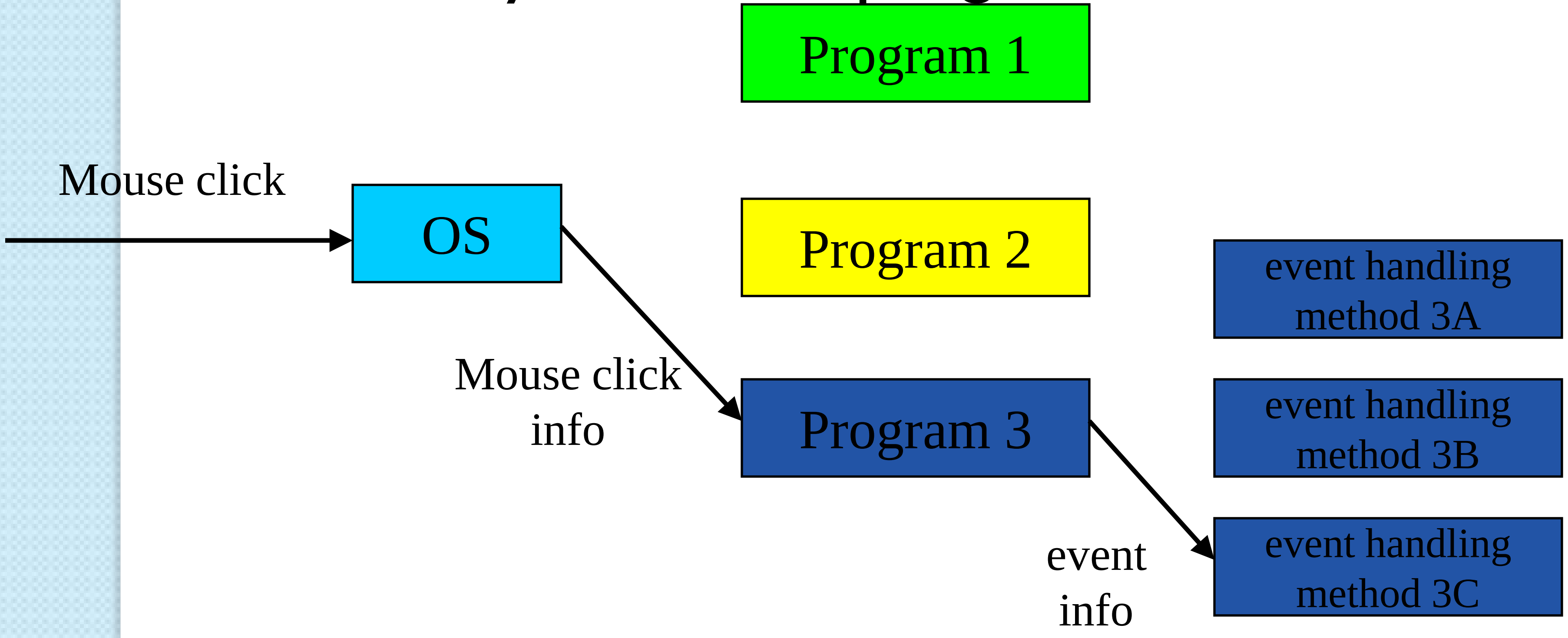
- The event is generated by external user actions such as mouse movements, mouse button clicks, and keystrokes, or by the operating system, such as a timer.

Event Information

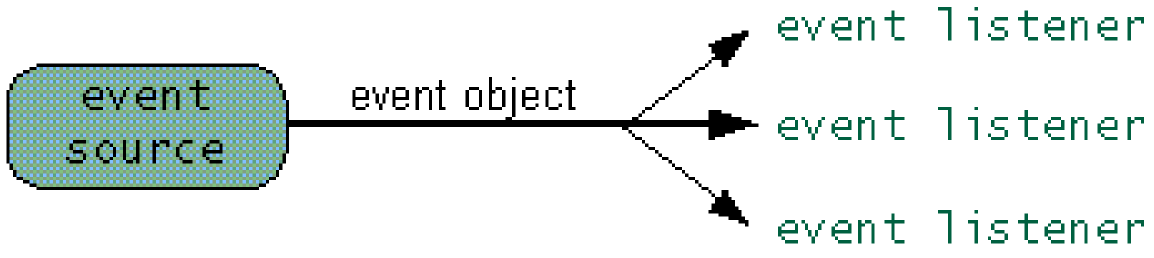
- `id`: A number that identifies the event.
- `target`: The source component upon which the event occurred.
- `arg`: Additional information about the source components.
- `x, y coordinates`: The mouse pointer location when a mouse movement event occurred.
- `clickCount`: The number of consecutive clicks for the mouse events. For other events, it is zero.
- `when`: The time stamp of the event.
- `key`: The key that was pressed or released.

Event Programming

- Operating Systems monitor events
  - Ex: keystrokes, mouse clicks, etc…
- OS relay events to programs



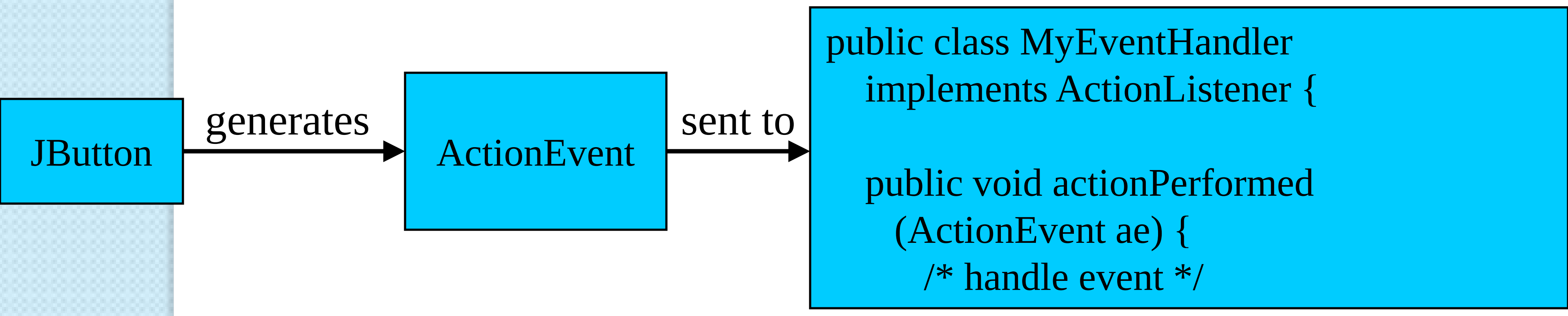
AWT Event Handling



- **Event Sources, Objects, and Listeners**
  - interacting with an event source triggers an event
    - e.g. **`JButton`**
  - when an event occurs, an event object is created
    - e.g. **`ActionEvent`**
    - contains information about the event
    - like what?
      - e.g., source of event
  - the event object is sent to all registered event listeners
    - e.g., a class that implements **`ActionListener`**
    - listener methods provide respond to user interactions

What types of interactions will your GUI provide?

- Different types of event sources are interacted with differently
  - e.g. button vs. combo box
- So, different types of event sources:
  - generate different types of event objects
  - register different types of listeners



```
public class MyEventHandler
   implements ActionListener {

   public void actionPerformed
      (ActionEvent ae) {
         /* handle event */
```

Listeners: Methods responding to Events

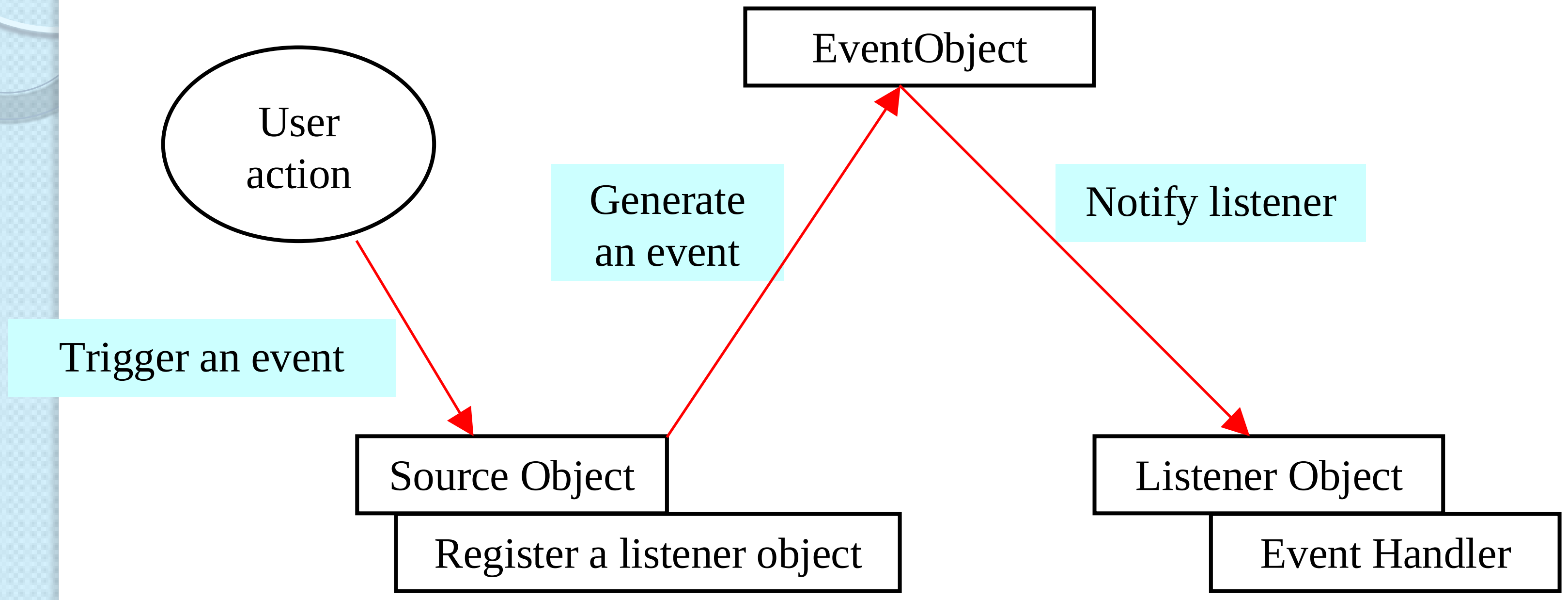
Examples

- **MouseMotionListener** – respond to mouse movements
  - Add "implements MouseMotionListener" to the GUI class
  - Code listener methods (e.g. mouseMoved()) and attach to the GUI object
- **ActionListener** – Recponds once to button selections
  - Add "implements ActionListener" to the GUI class
  - Code the "actionPerformed" method and attach to the GUI object
- **ItemListener** – Responds multiple times to changes to a component
  - Add "implements ItemListener" to the GUI class
  - Code the "itemStateChanged" method
  - Attach the ItemListener to the GUI object
- **Window Listener** – respond to clicks of a frame's X button
  - Create a class that extends WindowAdapter
  - Code the WindowListener methods and attach to the frame

Selected User Actions

User Action	Source Object	Event Type Generated
Clicked on a button	`JButton`	`ActionEvent`
Changed text	`JTextComponent`	`TextEvent`
Double-clicked on a list item	`JList`	`ActionEvent`
Selected or deselected an item with a single click	`JList`	`ItemEvent`
Selected or deselected an item	`JComboBox`	`ItemEvent`

The Delegation Model



Selected Event Handlers

Event Class	Listener Interface	Listener Methods (Handlers)
`ActionEvent`	`ActionListener`	`actionPerformed(ActionEvent)`
`ItemEvent`	`ItemListener`	`itemStateChanged(ItemEvent)`
`WindowEvent`	`WindowListener`	`windowClosing(WindowEvent)`
		`windowOpened(WindowEvent)`
		`windowIconified(WindowEvent)`
		`windowDeiconified(WindowEvent)`
		`windowClosed(WindowEvent)`
		`windowActivated(WindowEvent)`
		`windowDeactivated(WindowEvent)`

Events

- Events are central to programming for a graphical user interface
- The main() routine of a GUI program does not outline what will happen when the program is run, in a step-by-step process from beginning to end
- The program must be prepared to respond to various kinds of events that can happen at unpredictable times and in an order that the program doesn't control
- Why are events unpredictable?

Mouse and Keyboard

- The most basic kinds of events are generated by the mouse and keyboard
- The user can press any key on the keyboard, move the mouse, or press a button on the mouse
- The user can do any of these things at any time, and the computer has to respond appropriately.

Event Objects

- In Java, events are represented by objects
- When an event occurs, the system collects all the information relevant to the event and constructs an object to contain that information
- Different types of events are represented by objects belonging to different classes
- For example, when the user presses one of the buttons on a mouse, an object belonging to a class called MouseEvent is constructed

Event Objects (cont)

- When the user presses a key on the keyboard, a KeyEvent is created
- After the event object is constructed, it is passed as a parameter to a designated method
- By writing that method, the programmer says what should happen when the event occurs.

Event Processing

- As a Java programmer, you get a fairly high-level view of events
- There is a lot of processing that goes on between the time that the user presses a key or moves the mouse and the time that a subroutine in your program is called to respond to the event
- Fortunately, you don't need to know much about that processing

Event Loop

- But you should understand this much: Even though your GUI program doesn't have a main() routine, there is a sort of main routine running somewhere that executes a loop of the form – the Event Loop
  - while the program is still running:
  - Wait for the next event to occur
  - Call a subroutine to handle the event
- This loop is called an event loop
- Every GUI program has an event loop
- In Java, you don't have to write the loop

Event Handling

- For an event to have any effect, a program must detect the event and react to it
- In order to detect an event, the program must "listen" for it
- Listening for events is something that is done by an object called an event listener
- An event listener object must contain instance methods for handling the events for which it listens

Another Meaning of Interface

- An `interface` in Java is just a list of instance methods
- A class can "implement" an interface
- `public class MyClass extends JFrame implements ActionListener"`

How do we use AWT?

- We must define an event listener class
  - a class that implements a listener interface
  - responds to interactions inside event handling methods
  - listener objects use the information in the event object to determine their reaction to the event

How do we use AWT?

To build a GUI:

- construct an event source
  - `JButton okButton = new JButton("OK");`
- register the event listener with the event source
  - `okButton.addActionListener(this);`

Week Two: GUI Components

JButtons and JLabels

JButton

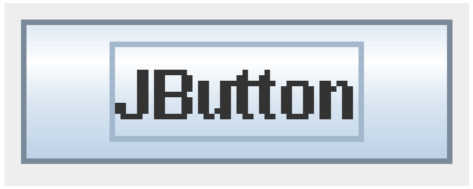
- Constructor – Creates JButton:
  - Declare and Create JButton
  - JButton exitButton = new JButton (“Exit”);
  - JButton myButton = new JButton (“Just a Button”);
- Event for JButton:
  - When the user clicks on a button, the button generates an event of type ActionEvent.
  - This event is sent to any listener that has been registered with the button as an ActionListener.

JButton

- Listeners
  - An object that wants to handle events generated by buttons must implement the ActionListener interface. This interface defines just one method,
  - public void actionPerformed(ActionEvent evt)
  - Registration of Listeners:
  - In order to actually receive notification of an event from a button, an ActionListener must be registered with the button. This is done with the button's addActionListener() method.
  - exitButton.addActionListener(buttonHandler);

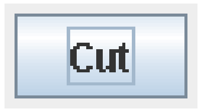
Other Things to Do with JButtons

- JButtons also have all the general Component methods
- setEnabled()
- setFont().
- setText()



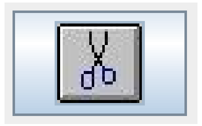
- Constructors
  - button with text

```
JButton cutButton = new JButton("Cut");
```



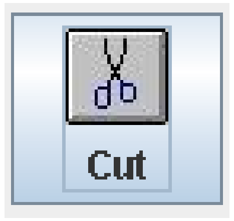
- button with an icon

```
JButton cutButton = new JButton(
              new ImageIcon("cut.gif"));
```



- button with an icon and text

```
JButton cutButton = new JButton(
         "Cut", new ImageIcon("cut.jpg"));
cutButton.setVerticalTextPosition(JButton.BOTTOM);
cutButton.setHorizontalTextPosition(JButton.CENTER);
```



JLabel

- The simplest type of component. An object of type JLabel exists just to display a line of text. The text cannot be edited by the user, although it can be changed by your program. The constructor for a JLabel specifies the text to be displayed

JLabel

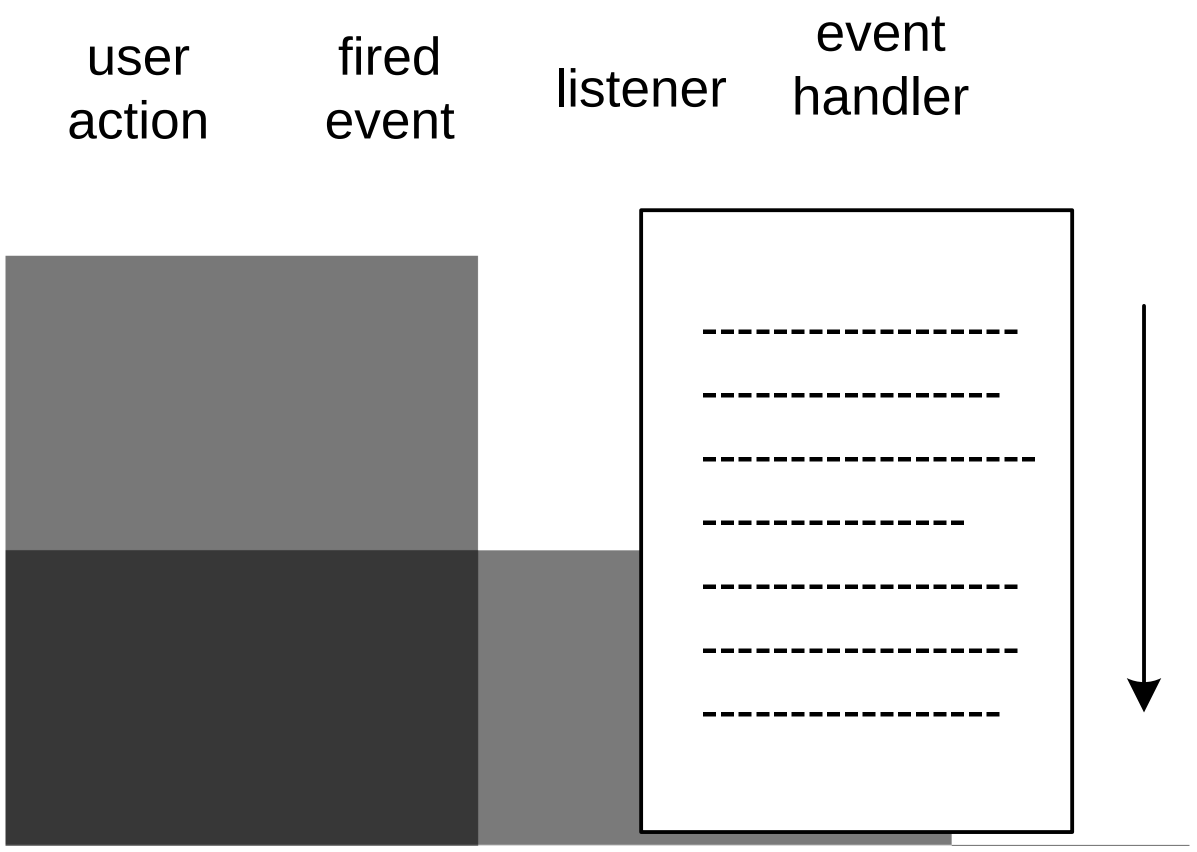
- Constructor – Creates JLabel:
  - Declare and Create JLabel
    - JLabel message = new JLabel("Hello World!");
    - JLabel message = new JLabel("Hello World!", JLabel.CENTER);
    - alignments are given by the constants JLabel.LEFT, JLabel.CENTER, and JLabel.RIGHT.

Other Things to Do with JLabels

- JLabels also have the following methods:
  - Label message = new JLabel("Hello World!", JLabel.CENTER);
  - message.setForeground(Color.RED);   // Display red text...
  - message.setBackground(Color.BLACK); //   on a black background...
  - message.setFont(new Font("Serif",Font.BOLD,18)); // in a big bold font.
  - message.setOpaque(true); // Make sure background is filled in.

Event-Driven Programming Basics

- What happens when a button is pressed:



Example
Handling Simple Action Events

- Objective: Display two buttons OK and Cancel in the window. A message is displayed to indicate which button is clicked, when a button is clicked.

Microsoft Word Document

Team Exercise

- Modify TestActionEvent.
- Add an Exit button that closes the Window
- The command for closing an application is

  System.out.exit(0);

Week Two: GUI Components

JTextFields

JTextField

- Used to collect and display a single line of text
- Constructor
  - JTextField(String text, int columns)
- Some methods
  - addActionListener(ActionListener a)
  - getSelectedText()
  - getText()
  - setEditable(boolean b)
  - setFont(Font f)
  - setHorizontalAlignment(int alignment)

JTextField

- JTextField class represents a component that contain text that can be edited by the user.
- A JTextField holds a single line of text.

JTextField Constructors

- **JTextField()**
- **JTextField(int columns)**
  - **where columns is an integer that specifies the number of characters that should be visible in the text field. This is used to determine the preferred width of the text field.**
  - **(Because characters can be of different sizes and because the preferred width is not always respected, the actual number of characters visible in the text field might not be equal to columns.)**
  - **If don't have to specify the number of columns the Layout Manager sizes the text field**

JLabels & JTextFields

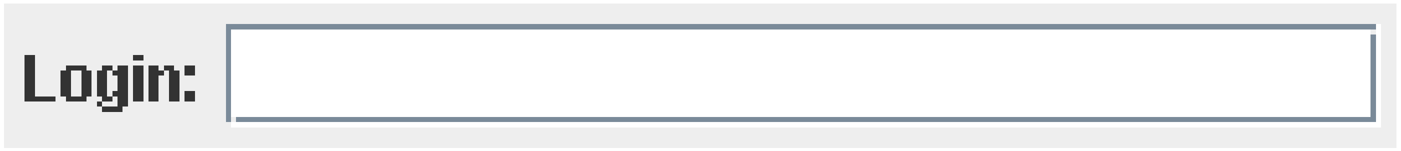
- Commonly paired
- Ex:

```
JLabel loginName = new  JLabel("Login:");
JTextField loginTF = new JTextField(20);
add(loginName);
add(loginTF);
```

Microsoft Word Document

Microsoft Word Document

Microsoft Word Document



Individual Exercise

- Modify SwingGrossPay.java
- If the employee works more than 40 hours, the pay should be time and a half for all overtime hours
- Add a "Clear" a button that clears out all text fields
- Add an Exit Jbutton
- **This is not Graded**

Questions?

Class Topics

Week One: User Interfaces

Week Two: GUI Components

Preview

Week Three: Files

Week Four: Applets and Graphics

Week Five: Database Connectivity and Mobile Computing

Week Three: Files			
	Details	Due	Points
Learning Team Algorithm	Create an algorithm for your Week 5 program. The algorithm can be either in the form of a flow chart or pseudo code. Submit the algorithm to your facilitator.	Assignment Tab Tuesday, December 10, 2013 6:00 PM	5

Week Three: Files

	Details	Due	Points
Individual Income Tax Program	Individual Assignment: This assignment should be an application, not an applet, written in Java with a graphical user interface. Write an application that contains a JButton, and three JTextFields. This program should calculate Income Tax. The user enters gross pay and total deductions in two of the text fields. When the user presses the button, the income tax due should be displayed in the third text field. To calculate the income tax, first subtract the deductions from the gross pay. This difference is the net pay. If the net pay is less than 10 thousand dollars, then the income tax due is zero. If the net pay is greater than or equal to 10 thousand dollars, then the tax due is twenty five percent of the net pay. All input and output should be through the GUI not via the console. Submit the .java source code files with the correct file names for all classes used. The program must consist of at least one public class. Programs are expected to compile and run/execute correctly. There should be proper documentation (comments) in the source code. The documentation should include a block containing the name of the program and the name of the student, and a short description of the program.	Assignment Tab Tuesday, December 10, 2013 6:00 PM	15

Questions?

Questions?

See You Next Week !!