Today …

- Quiz 4
- GUI Components
  - Frames, Text Areas, and Dialogs
  - Layout Managers
- Lab 3
- Readings:
  - Ch. 7 pages 285-318
GUI Programming in Java

- Early versions of Java used the “AWT”
  - Abstract Window Toolkit
  - Provides an API over the underlying graphics support provided on each platform (Windows, Mac, Linux, etc.)

- “Swing” extends AWT
  - Provides a richer API
  - A more consistent “look and feel”
  - Also supports native “look and feels” as well
  - Plus numerous others (including customized ones)

Swing Components (Widgets)

Some Basic Controls (for input)

- JButton
- JTextField
- JRadioBu5on
- JCheckBox
- JComboBox
- JList
- JSlider
Swing Components (Widgets)

More Complex Displays

Swing Components (Widgets)

Un-editable Information Displays
Swing Components (Widgets)

Top-Level Containers

- JFrame
- JDialog

JApplet (we've seen these)

Swing Components (Widgets)

General Purpose Containers

- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane

Icons
Swing Components (Widgets)

Special-Purpose Containers

JInternalFrame

JLayeredPane

JPanel

- Often just called “panes”
- A general-purpose container for widgets
- Widgets are added using panel.add(…)
- Panels are also widgets …
  … so can be hierarchically nested (can be shared)

- Basic operations:
  - add(…)
  - getComponent(…)
  - remove(…)
  - set the background color
  - set the layout manager (more later)

Here and throughout the lectures I expect you to look these up in the Java API docs and/or the textbook...

... instead of going through all the method details in class
JFrame

A top-level window *not* contained in any other window

```java
import javax.swing.*;
public class SimpleFrameDemo {
    public static void main(String[] args) {
        Runnable r = new Runnable() {
            public void run() {
                JFrame frame = new JFrame("Frame Demo");
                frame.setSize(300, 200);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setVisible(true);
            }
        };
        SwingUtilities.invokeLater(r);
    }
}
```

Basic frame operations

- Set and get the title
- Set and get the close operation
- Pack the window (preferred size of widgets)
- Set the size
- Set the location
- Add widgets to the content pane
- Set the menu bar
- ...
**JFrame Anatomy**

*Each frame has*

- A **root pane**
  - Manages the interior of the pane (the other panes)
- A **layered pane**
  - To put widgets on top of or behind other widgets
- A **menu bar** (optional)
  - For menu items (File -> Save, Edit -> Copy, etc.)
- A **content pane**
  - The visible widgets of the frame
  - Widgets added to this pane from `frame.add(...)`
- A **glass pane**
  - For intercepting events before the content pane
**JTextArea**

A text “field” for multiple lines
— So, supports both input and output

```java
import javax.swing.*;
public class SimpleFrameDemo {
    public static void main(String[] args) {
        Runnable r = new Runnable() {
            public void run() {
                JFrame frame = new JFrame("Frame Demo");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                JTextArea textArea = new JTextArea(20, 40);
                frame.add(textArea);
                frame.pack();
                frame.setVisible(true);
            }
        }
        SwingUtilities.invokeLater(r);
    }
}
```

**JTextArea**

Basic text area operations

- Append a string
- Insert into a specific location
- Replace a certain range
- Set whether to wrap lines
- Get text
- Get selected text
- Set whether it is editable
- …
Adding a Scroll Bar

- We can use a JScrollPane for this

```java
import javax.swing.*;
public class SimpleFrameDemo {
    public static void main(String[] args) {
        Runnable r = new Runnable() {
            public void run() {
                JFrame frame = new JFrame("Frame Demo");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                JTextArea textArea = new JTextArea(20, 40);
                JScrollPane scrollPane = new JScrollPane(textArea);
                frame.add(scrollPane);
                frame.pack();
                frame.setVisible(true);
            }
        };
        SwingUtilities.invokeLater(r);
    }
}
```

JScrollBar

**Basic operations**

- Set horizontal and vertical policy
- E.g., as needed, never, always
- Get the size of the “viewport”
  - ...
Layout Managers

Control how widgets are arranged on a panel

- Border Layout
- Box Layout (* recommended for this class)
- Card Layout
- Flow Layout (* the default)
- GridBag Layout
- Grid Layout
- Group Layout (* used in gui builders, e.g., NetBeans)
- Spring Layout (* also used by gui builders)

• The java tutorials cover these in detail

Flow Layout

• Adds widgets from left to right at their preferred size
  - you can set the preferred size of all widgets ...
    ... using setPreferredSize(…)
  - You can also set the maximum and minimum size ...
    ... using setMaximumSize(…) and setMinimumSize(…)

• This is the default layout manager for panels
  - Layout depends on the size of the frame
  - Should only be used with very simple UI's
**Box Layout**

*I've found this to be the easiest to use for laying out panels by hand (without a builder tool)*

- Widgets placed either vertically (Y or PAGE AXIS)  
  … or horizontally (X or LINE AXIS)

- Can create “**rigid**” areas  
  – Invisible components of a certain dimension

- Can create “**glue**” areas  
  – Invisible components that grow as needed to absorb space  
  – E.g., when the panel is resized

- Can align based on the components “orientation”

---

**Setting the Box Layout**

*Use the `setLayout(…)` method of JPanel*

```java
JPanel panel = new JPanel();
panel.setLayout(new BoxLayout(panel, BoxLayout.X_AXIS));
...
or ...
Panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
```

**Rigid areas**

```java
pane.add(Box.createRigidArea(new Dimension(w, h)));
```

**Glue**

```java
pane.add(Box.createHorizontalGlue());
...
or ...
pane.add(Box.createVerticalGlue());
```
Dialog Boxes (errors, warnings, messages, etc.)

Super easy with JOptionPane

- JOptionPane.showMessageDialog(…)
  - Show message and wait for user to click OK

- JOptionPane.showConfirmDialog(…)
  - Show message and get confirmation (OK, cancel, etc.)

- JOptionPane.showOptionDialog(…)
  - Show message and get a user choice from a set of options

- JOptionPane.showInputDialog(…)
  - Show message and get a line of user input

Lab 3