Intro to Graphics

GUIs
GUI Definitions

Graphical User Interface

- A “window” – a smaller screen within your monitor.
  - Usually contains text, buttons, menus, etc.
  - Has a border, close-window “X” button, inside.

How’s the window know which button you push?

- Each click on a window is an “event”.
- Window has a “listener”.
  - Listens for events.
  - Just sits there constantly checking for events.
Java GUI: Swing

Most of the Java GUI is in package called Swing.

- import javax.swing.*;

- The classes have names like
  - JFrame
  - JLabel
  - JButton
  - JMenu
  - etc.

- The J in front distinguishes from an older “AWT” graphics package.
  - We still use many AWT classes that don’t have J.
  - e.g., WindowAdapter
  - import java.awt.*;
Parts of the Java GUI: JFrame

**JFrame class**

- This is the window.
- JFrame myWindow = new JFrame();

- Can set the size in pixels.
  - myWindow.setSize(width, height);

- Must make window visible.
  - myWindows.setVisible(true);
  - Default is invisible.
  - Handy if you want to hide windows temporarily (set to false).
Parts of the Java GUI: Content Pane

Content Pane (or Container)

- The inner part of the window (not the border).
- This is where you put the text, buttons, menus etc.
- `Container myContentPane = myWindow.getContentPane();`

- Can add any GUI Component.
  - `myContentPane.add(…);`

- Must import the Container class
  - `import java.awt.*;`
Parts of the Java GUI: JComponent

- JComponent
  - All labels, buttons, menus, scroll bars, etc.
  - JLabel, JButton, JMenu, JScrollBar, etc.

- e.g.,
  - JLabel myLabel = new JLabel("Yo, how’s life?");
    myContentPane.add(myLabel);
    • Now this label is displayed in the window.

  - JButton myButton = new JButton("Push me");
    myContentPane.add(myButton);
    • Now this button is displayed in the window.
Parts of the Java GUI: WindowAdapter

- WindowAdapter
  - The JFrame has a close-window box.
    - How’s it know when it was clicked?
  - **WindowAdapter listens for window events.**

- But WindowAdapter is only a template.
  - How does it know what you would like to happen when the close-window box is clicked?
    - Maybe you want a confirmation dialog to pop up.
    - Maybe you want to end the program… `System.exit(0);`

- You have to fill in the template.
import java.awt.event.*;

public class WindowEventListener extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
}

This says: I’m using the template “WindowAdapter” to create my class called “WindowEventListener”.

This method was already in “WindowAdapter” but only as a template. It contained no code. So we had to add the code “System.exit(0);” or anything else we want to happen.
Parts of the Java GUI: Using WindowAdapter

After defining our listener in separate class, can use it with JFrame.

```java
JFrame myWindow = new JFrame();
WindowEventListener myListener = new WindowEventListener();
myWindow.addWindowListener(myListener);
```

Now the JFrame listens for a window closing event.
import javax.swing.*;
import java.awt.*;
public class FirstGraphics
{
    public static void main(String[] args)
    {
        JFrame myWindow = new JFrame();
        WindowEventListener myListener = new WindowEventListener();
        JButton myButton = new JButton("Push me");

        myWindow.addWindowListener(myListener);
        myWindow.setSize(300, 400);
        myWindow.setTitle("Test Window");

        Container myContentPane = myWindow.getContentPane();
        myContentPane.add(myButton);

        myWindow.setVisible(true);
    }
}

Note: I assume that we have already written this listener!

Must make visible AFTER adding components! Otherwise doesn’t show the components.
The Result Looks Like…

This entire box is the button!
What’s the Button Do?

- Nothing until we add a Listener!
  - Clicking the button is an event.
  - As with JFrame, there is already a listener “template”.
    - called ActionListener
  - Once we create the listener

```java
JButton myButton = new JButton();
ButtonListener myButtonListener = new ButtonListener();
myButton.addActionListener(myButtonListener);
myContentPane.add(myButton);
```

We have to create this listener.
ActionListener

Similar to last listener, but a different kind of template. Uses “implements” instead of “extends”.

import java.awt.event.*;
import javax.swing.*;
public class ButtonListener implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        JOptionPane.showMessageDialog(null, “You dare to PUSH me?!”);
    }
}
The Result of Our Button Listener

```java
public static void main(String[] args) {
    JFrame myWindow = new JFrame();
    WindowEventListener myListener =
        new WindowEventListener();
    ButtonListener myButtonListener =
        new ButtonListener();
    JButton myButton = new JButton("Push me");
    myButton.addActionListener(myButtonListener);
    myWindow.addWindowListener(myListener);
    myWindow.setSize(300, 400);
    myWindow.setTitle("Test Window");
    Container myContentPane =
        myWindow.getContentPane();
    myContentPane.add(myButton);
    myWindow.setVisible(true);
}
```
Layouts Managers: Adding More Components

To add more components to window, they must be arranged.

- Use one of the layout managers
  - GridLayout (specify a grid size)
  - FlowLayout (in a line next to each other)
  - BorderLayout (north, south, east, west center)
  - GridBagLayout (most flexible)
  - Etc.

- Their only job is arranging the components in the window.
BorderLayout

Five areas

- designated north, south, east, west, and center

If use all five:

If use only N, S, Center:

If use only E, W, Center:
Using BorderLayout

```java
JFrame myWindow = new JFrame();
WindowEventListener myListener = new WindowEventListener();
JLabel myLabel = new JLabel("Yo");
JButton myButton = new JButton("Push me");

myWindow.addWindowListener(myListener);
myWindow.setSize(300, 400);

Container myContentPane = myWindow.getContentPane();

BorderLayout layout = new BorderLayout();
myContentPane.setLayout(layout);
myContentPane.add(myLabel, BorderLayout.WEST);
myContentPane.add(myButton, BorderLayout.EAST);
```
Grid Layout With Two Elements

- Ok, not too snazzy.
  - Will look a little better if we add stuff to the north and south.

- Or use GridBagLayout.
  - complicated, but good
  - Study API or a book.

- Or put layouts inside of layouts.
  - Use JPanels.
  - Study book!

This part of the window is the button!
Note on Structure

In Object-Oriented programming, we’ll learn a better way to write a window.

- We’ll make more use of the “templates”!
- Exact same code.
  - Just put into more classes.

- Called “inheritance”.

Don’t sweat it.

- What I’ve shown you works great.
Want to Learn More?

- Graphics is a big topic.
- But I’ve introduced you to all the basics.
- You should be able to open up a book on Java graphics and learn anything you want.
- Also, remember the Java API!