26 Feb 2013 GUIs: Graphical User Interfaces

Their mouse had a mean time between failure of ... a week ... it would jam up irreparably, or ... jam up on the table-- ... It had a flimsy cord whose wires would break. Steve Jobs: "... Xerox says it can't be built for < $400, I want a $10 mouse that will never fail and can be mass produced, because it's going to be the primary interface of the computer ..."

... Dean Hovey ... came back, "I've got some good and some bad news. We'd design a mouse for 10 bucks."

Good news: we've got a new project with Apple. Bad news: I told Steve we'd design a mouse for 10 bucks."

Their mouse had a mean time between failure of ... a week ... it would jam up irreparably, or ... jam up on the table-- ... It had a flimsy cord whose wires would break. Steve Jobs: "... Xerox says it can't be built for < $400, I want a $10 mouse that will never fail and can be mass produced, because it's going to be the primary interface of the computer ..."

We use Java’s two packages for doing GUIs:

- AWT (Abstract or Awful Window Toolkit) — first one
- Swing — a newer one, which builds on AWT as much as possible

Two aspects to making a GUI:
1. Placing components (buttons, text, etc.) in it.
2. Listening/responding to events

The program reacts or programming: The program reacts to events such as button clicks, mouse movement, keyboard input

GUI (Graphical User Interface)

- Provides a friendly interface between user and program
- Allows event-driven or reactive programming: The program reacts to events such as button clicks, mouse movement, keyboard input
- Often is multi-threaded: Different threads of execution can be going on simultaneously

We use Java’s two packages for doing GUIs:

- AWT (Abstract or Awful Window Toolkit) — first one
- Swing — a newer one, which builds on AWT as much as possible

Class JFrame

JFrame object: associated with a window on our monitor.

Generally, a GUI is a JFrame object with various components placed in it

Some methods in a JFrame object

- hide() show() setVisible(boolean)
- getN() getY() (coordinates of top-left point)
- getWidth() getHeight() setLocation(int, int)
- getTitle() setTitle(String)
- getLocation() setLocation(int, int)

Over 100 methods in a JFrame object!

Class JFrame is in package javax.swing

Putting components in a JFrame

import java.awt.*; import javax.swing.*;
/** Demonstrate placement of components in a JFrame.
 Places five components in the five possible areas:
 (1) a JButton in the east.
 (2) a JLabel in the west.
 (3) a JLabel in the south.
 (4) a JTextArea in the north, and
 (5) a JTextField in the center. */

public class ComponentExample extends JFrame {

  public ComponentExample(String t) {

    Container cp = getContentPane();
    cp.add(new JButton("Click me"), BorderLayout.NORTH);
    cp.add(new JLabel("I got up today"), BorderLayout.WEST);
    cp.add(new JTextField("type here", 4, 10), BorderLayout.SOUTH);
    cp.add(new JTextArea("Enter message"), BorderLayout.CENTER);
    cp.add(new JCheckBox("West") , BorderLayout.EAST);
    cp.add(new JCheckBox("South") , BorderLayout.SOUTH);
    cp.add(new JButton("Click me") , BorderLayout.SOUTH);

    setVisible(true);
  }
}

ComponentExample.java

Placing components in a JFrame

Layout manager: Instance controls placement of components.
JFrame layout manager default: BorderLayout.
BorderLayout layout manager: Can place 5 components:

public class C extends JFrame {
  public C() {
    Container cp = getContentPane();
    cp.add(new JButton("Click here"), BorderLayout.CENTER);
    cp.add(new JLabel("label 1"), BorderLayout.NORTH);
    cp.add(new JLabel("label 2"), BorderLayout.EAST);
    cp.add(new JButton("Click me"), BorderLayout.SOUTH);
    cp.add(new JLabel("label 3") , BorderLayout.SOUTH);
    cp.add(new JButton("Click me") , BorderLayout.SOUTH);

    setVisible(true);
  }
}

Frameworks that contain classes that deal with GUIs:

java.awt: Old package. javax.swing: New package.

javax.swing has a better way of listening to buttons, text fields, etc. Components are more flexible.

Component: Something that can be placed in a GUI window. They are instances of certain classes, e.g.

JButton, JButton: Clickable button
JLabel, JLabel: Line of text
JTextField, JTextField: Field into which the user can type:
JTextArea, JTextArea: Many-row field into which the user can type
JPanel, JPanel: Used for graphics; to contain other components
JCheckBox: Checkable box with a title
JComboBox: Menus of items, one of which can be checked
JRadioButton: Same functionality as JCheckBox
Container: Can contain other components
Box: Can contain other components
**Constructor: frame with title t and 3 columns with n, n+1, and n+2 buttons.**

```java
public class BoxDemo2 extends JFrame {
    // Create Box b with n buttons.
    Box b1 = new Box(BoxLayout.Y_AXIS);
    for (int i = 0; i < n; i++)
        b1.add(new JButton(1 + i));
    // Create Box b2 with n+1 buttons.
    Box b2 = ...
    // Create Box b3 with n+2 buttons.
    Box b3 = ...
    // Create horizontal box h containing b1, b2, b3
    Box h = new Box(BoxLayout.X_AXIS);
    h.add(b1);
    h.add(b2);
    h.add(b3);
    Container cp = getContentPane();
    cp.add(h, BorderLayout.CENTER);
    pack();
    show();
}
```

They appear in the order added, taking only one row.

---

**Class Box: a container**

```java
import java.awt.*;
import java.awt.event.*;

public class BoxDemo3 extends JFrame {
    // Create a JFrame.
    public BoxDemo3() {
        super("Box demo");
        Container cp = getContentPane();
        cp.add(new JLabel("west"), BorderLayout.WEST);
        cp.add(new JLabel("east"), BorderLayout.EAST);
        cp.add(new JLabel("center"), BorderLayout.CENTER);
        pack();
    }
}
```

They appear in the order added, taking only one row.

---

**Simulate BoxLayout Manager in a JFrame**

To simulate using a BoxLayout manager for a JFrame, create a Box and place it as the sole component of the JFrame:

```java
JFrame jf = new JFrame("Panel demo");
Box b = new Box(BoxLayout.Y_AXIS);
jf.add(b, BorderLayout.CENTER);
```

1. Start developing a GUI by changing an already existing one. A lot of details. Hard to get all details right when one starts from scratch and has little idea about the Java GUI package.
2. Showed how to place components in a GUI. Next time: how to "listen" to things like button clicks in a GUI.