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. Good news: we've got a new project with Apple. Bad news: I told Steve we'd design a mouse for 10 bucks." ... year later ... we ... filed ... and were granted a patent, on the electro-mechanical-optical mouse of today; ... we ended up ... (making) the mouse as invisible to people as it is today.


Prelim 1

This histogram is for the Corrected Prelim 1.

1. Fluency in Basic Java:Loops, Strings, arrays
2. Fluency with recursion
3. Trees are defined recursively. Therefore recursion is the natural tool for processing trees
4. Strive for simplicity, brevity, clarity, beauty

Answering a question on prelim 1

/** An instance is a node of a binary tree. */

```java
public class TreeNode {

private int val; // Value of node.
private TreeNode left; // Left child --null if none.
private TreeNode right; // Right child --null if none.

/** Return true iff following properties hold:
* 1. All values in the tree with this node as root are >= min.
* 2. All values in the tree with this node as root are <= max.
* 3. This tree is a BST. */
public boolean isBST(int min, int max) {

// Check properties for root value
min  <= val   &&   val  <= max

Look for simple solution

Draw a binary tree for insight

Try for something simple --always

/** Return true iff following properties hold:
* 1. All values in the tree with this node as root are >= min.
* 2. All values in the tree with this node as root are <= max.
* 3. This tree is a BST. */
public boolean isBST(int min, int max) {

Try to keep things simple!

If things work out, may be able to write a single return statement, with each of the 3 points in it.
```

Points 1 and 2 for root value

/** Return true iff following properties hold:
* 1. All values in the tree with this node as root are >= min.
* 2. All values in the tree with this node as root are <= max.
* 3. This tree is a BST. */
public boolean isBST(int min, int max) {

Try to keep things simple!

For points 1 and 2, have to test the root
Points 1 and 2 for subtrees

```java
/** Return true iff following properties hold:
*  1. All values in the tree with this node as
*     root are >= min.
*  2. All values in the tree with this node as
*     root are <= max.
*  3. This tree is a BST. */
public boolean isBST(int min, int max) {
    return min <= val && val <= max
        && (left == null || left.isBST(min, max))
        && (right == null || right.isBST(min, max));
}
```

But the subtree values have to be in same range
(If the subtrees exist!! Always think of this case)

Use recursion

Point 3

```java
/** Return true iff following properties hold:
*  1. All values in the tree with this node as
*     root are >= min.
*  2. All values in the tree with this node as
*     root are <= max.
*  3. This tree is a BST. */
public boolean isBST(int min, int max) {
    return min <= val && val <= max &&
        (left == null || left.isBST(min, max))
        && (right == null || right.isBST(min, max));
}
```

That takes care of points 1, 2. Point 3?
Values in left subtree have to be < val.
Change the argument to isBST. Right subtree similar

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

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

Class JFrame

- JFrame object: associated with a window on your monitor.
- Generally, a GUI is a JFrame object with various components
  placed in it

Some methods in a JFrame object
- hide() show() setVisible(boolean)
- getX() getY() (coordinates of top-left point)
- getWidth() getHeight() setLocation(int, int)
- getTitle() setTitle(String)
- setLocation(int, int)

Over 100 methods in a JFrame object!

Class JFrame is in package javax.swing

Placing components in a JFrame

- Layout manager: Instance controls placement of components.
- JFrame layout manager default: BorderLayout.

```java
public class C extends JFrame {
    public C() {
        Container cp = getContentPane();
        JButton jb = new JButton("Click here");
        JLabel jl = new JLabel("label 2");
        cp.add(jb, BorderLayout.EAST);
        cp.add(jl, BorderLayout.WEST);
        pack();
        setVisible(true);
    }
}
```

Putting components in a JFrame

```java
public class ComponentExample extends JFrame {
    public ComponentExample(String t) {
        Container cp = getContentPane();
        cp.add(new JButton("Click me"), BorderLayout.EAST);
        cp.add(new JTextField("type here", 22), BorderLayout.SOUTH);
        cp.add(new JCheckBox("I got up today"), BorderLayout.CENTER);
        cp.add(new JTextArea("type here", 4, 10), BorderLayout.WEST);
        pack();
    }
}
```
Components that can contain other components

Component

Box
Container
JComponent
JPanel
Panel
Applet
Window
Frame
JFrame
JWindow

Java.awt is the old GUI package.
Java.swing is the new GUI package.
When they wanted to use an old name, they put \texttt{J} in front of it.
(e.g. \texttt{Frame} and \texttt{JFrame})

When constructing java.swing, the attempt was made to rely on the old package as much as possible.
So, \texttt{JFrame} is a subclass of \texttt{Frame}.
But they couldn’t do this with \texttt{JPanel}.

Basic Components

Component: Something that can be placed in a GUI window. These are the basic ones used in GUIs.

Note the use of subclasses to provide structure and efficiency. For example, there are two kinds of JToggleButton, so that class has two subclasses.

import java.awt.*; import javax.swing.*;

public class PanelDemo extends Frame {
  JPanel p = new JPanel();

  // Constructor: a frame with title "Panel demo", labels in east/west, blank label in south, (Panel of 4 buttons in the center)
  public PanelDemo() {
    super("Panel demo");
    p.add(new Button("0"));
    p.add(new Button("1"));
    p.add(new Button("2"));
    p.add(new Button("3"));
    Container cp = getContentPane();
    cp.add(new Label("east"), BorderLayout.EAST);
    cp.add(new Label("west"), BorderLayout.WEST);
    cp.add(new Label("south"), BorderLayout.SOUTH);
    cp.add(new TextComponent(), BorderLayout.CENTER);
    pack();
  }

  Window layout manager default: FlowLayout.
  JPanel as a container

  FlowLayout layout manager: Place any number of components.
  They appear in the order added, taking as many rows as necessary.

public class BoxDemo2 extends Frame {
  public BoxDemo2(int n) {
    super();
    // Create Box b1 with n-1 buttons.
    Box b1 = new Box(BoxLayout.Y_AXIS);
    for (int i = 0; i < n-1; i++)
      b1.add(new Button("button \(^i\) \& \(i\))
    // Create Box b2 with n+1 buttons.
    Box b2 = ...
    // Create Box b3 with n+2 buttons.
    Box b3 = ...
    // Create horizontal box b containing b1, b2, b3
    Box b = new Box(BoxLayout.Y_AXIS);
    b.add(b1);
    b.add(b2);
    b.add(b3);
    Container cp = getContentPane();
    cp.add(b, BorderLayout.CENTER);
    pack();
  }

  // 3 vertical boxes, each a column of buttons, are placed in a horizontal box

  BoxLayout layout manager: Place any number of components.
  They appear in the order added, taking as many rows as necessary.

Boxes within a Box

public class BoxDemo extends Frame {
  // Constructor: frame with title "Box demo", labels in east/west, blank label in south, (Box of 3 columns with n, n+1, and n+2 buttons)
  public BoxDemo(int n) {
    super();
    // Create Box b1 with n buttons.
    Box b1 = new Box(BoxLayout.Y_AXIS);
    for (int i = 0; i < n; i++)
      b1.add(new JButton("button \(i\) \& \(i\))
    // Create Box b2 with n+1 buttons.
    Box b2 = ...
    // Create Box b3 with n+2 buttons.
    Box b3 = ...
    // Create horizontal box b containing b1, b2, b3
    Box b = new Box(BoxLayout.X_AXIS);
    b.add(b1);
    b.add(b2);
    b.add(b3);
    Container cp = getContentPane();
    cp.add(b, BorderLayout.CENTER);
    pack();
  }

  // 3 vertical boxes, each a column of buttons, are placed in a horizontal box

  BoxLayout layout manager: Place any number of components.
  They appear in the order added, taking only one row.

Class Box: a container

import java.awt.*; import javax.swing.*;

public class BoxDemo extends Frame {
  public BoxDemo() {
    super("Box demo");
    Box b = new Box(BoxLayout.Y_AXIS);
    b.add(new JTextField("0"));
    b.add(new JTextField("1"));
    b.add(new JTextField("2"));
    b.add(new JTextField("3"));
    Container cp = getContentPane();
    cp.add(new JLabel("east"), BorderLayout.EAST);
    cp.add(new JLabel("west"), BorderLayout.WEST);
    cp.add(new JLabel("south"), BorderLayout.SOUTH);
    cp.add(new Label(), BorderLayout.CENTER);
    pack();
  }

  Box layout manager default: BoxLayout.
  BoxLayout layout manager: Place any number of components.
  They appear in the order added, taking only one row.

Basic Components

Component: Button, Canvas, Checkbox, Choice, Label, List, Scrollbar, TextArea, Container

JComponent
AbstractButton
Button
JToggleButton
JCheckBox
JRadioButton
JList
JOptionPane
JPanel
JPopupMenu
JScrollBar
JSlider
JTextComponent
JTextComponent
JTextField
JTextArea
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("title");
Box b = new Box(BoxLayout.X_AXIS);
Add components to b;
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.