Java GUI Programming

Building the GUI for the Microsoft Windows Calculator
Lecture 2

Des Traynor 2005
So, what are we up to

• Today, we are going to create the GUI for the calculator you all know and love.

• It will basically be a crash course in putting together simple interfaces using Swing.

• You will learn...
  – Making and placing GUI components
  – Creating GUIs using JPanel to break down the design
  – Decorating GUI Components, via Fonts, Colors (sic) and Insets
  – Using Layout Managers
  – JMenuBar, and JMenu
The Calculator
The Calculator, broken into 3 JPanels
So What?

• Well, we have now gone from 1 large problem, to three small ones.
• We will do our best to create each JPanel, one at a time.
• So, starting with the simplest, the JPanel containing the single JTextField.
class textPanel extends JPanel {

    JTextField output;

    textPanel() {
        // set size, initial text, font and text alignment
        output = new JTextField("0.", 28);
        output.setFont(new Font("Courier", Font.PLAIN, 16));
        output.setHorizontalAlignment(4);
        //add to panel
        this.add(output);
        // change background color to be that gray color
        Color background_color = new Color(241, 237, 222);
        this.setBackground(background_color);
    }

    // main removed.
}
So lets have a look at that

- (goto text editor, show main etc )

- There were some new methods there, that you wouldn't have seen before. You do not need to learn these off by heart, they are all available at http://java.sun.com

- You can look up this site all the time during labs, but not during your lab exam, or your final exam. I will provide the signatures of any of the necessary methods in these occasions (I think).
Happy Enough with textPanel? Great, next is the first row of buttons

• These bad boys...

• So, what we have here is 4 buttons, 3 with red text, and 2 of them are heavily padded
• Normally buttons shrink/expand to fit their label. CE and C aren't doing that.
• The first yoke, is actually a disabled button (I worked that out after a while)
/* The code for the whole Panel won't fit in a slide, so I'll explain one JButton and hopefully you genii can work out the rest */

JButton CE;
// This is RGB value of the "windows button color"
Color background_color = new Color(241, 237, 222);
/* The string passed in to constructor is the label that appears on the JButton */
CE = new JButton("CE");
// This bit sets the button text to be red.
CE.setForeground(Color.red);
// This sets the background color to be the color above
CE.setBackground(buttons_color);
/* This sets the borders, we'll be talking about this in a second */
CE.setMargin(new Insets(6, 29, 6, 29));
// Then add it to buttonPanel1
add(CE);
Insets ?

- Insets is the amount of space between the component, and its border.

- For a button it's like

- You create them like so

Insets(int top, int left, int bottom, int right)

- All distances are in pixels.
So let's have a look at buttonPanel1.java

• (goto text editor and explain)

• (this is the bit where you realise that attending lectures was probably a good idea)
2 down, one to go.

- Now for the slight challenge, the 24 buttons
- Some buttons are blue, some aren't, lot of copying and pasting to specify this.

Basically every button has the following things...

```java
Insets buttonMargin = new Insets(5,1,5,1);
MC = new JButton("MC"); //name
MC.setBackground(buttons_color); //color fg
MC.setForeground(Color.red); // color bg
MC.setMargin(buttonMargin); //margins
```
The GridLayout we are using

- `setLayout(new GridLayout(4,6,5,5));`

- GridLayout places components as you add them, starting top right, ending bottom left. You can not directly place a component in a specific place easily.

- The 4 and 6 relate the to the 4x6 layout, the 5 and 5 relate to the horizontal and vertical gap between components (in pixels.)

- Again, lets take a look at the code itself. (buttonPanel2.java)
Putting it all together

• Well, we have built the 3 sub components, so we now need to build the Frame that contains the three of them, in the correct fashion.

• We will also add a JMenu bar to complete the look of the program.
class CalcFrame extends JFrame {

/* These are our three panels that we already wrote */
textPanel tp;
buttonPanel1 bp1;
buttonPanel2 bp2;

/** This is the menu stuff **/
JMenuBar jmb;
JMenu  jmi1,jmi2,jmi3;
ClassFrame.java (the constructor)

CalcFrame()
{
    super("Calculator"); // set title to Calculator
    Container cp = getContentPane(); // we saw this already
    cp.setLayout(new FlowLayout()); // we have to use flow
    cp.setBackground(new Color(241,237,222));
    tp = new textPanel(); // create our three components
    bp1  = new buttonPanel1();
    bp2  = new buttonPanel2();
    cp.add(tp);
    cp.add(bp1); // and add them
    cp.add(bp2); // in order
    .... // not finished
The JMenuBar is the top level line. It is initially empty.

A JMenu is a single menu (e.g. File, or Edit)

A JMenuItem is an item inside a JMenu, (e.g. Open, Cut)

To get a submenu, simply add a JMenu to a JMenu.
JMenus and their associates

One JMenuBar
5 JMenus
7 JMenuItems
So let's have a look at our Menu

- Goto Text Editor
- Explain stuff.
The final bits of code

this.setDefaultCloseOperation(EXIT_ON_CLOSE);
this.setSize(340,320); // I worked out the size afterward
setVisible(true);
setResizable(false); // This prevents it getting messed up

• To make the actual CalcFrame appear...

public static void main(String args[])
{
    CalcFrame cf = new CalcFrame();
}
So, how did we do?

Microsoft Windows Calculator
Version 5.1

The CS211 Calculator
Good enough

• It's close. Certainly not perfect.

• We could have set the Look And Feel to be “MS Windows”, that would get the buttons spot on. (If we have time, I'll elaborate)

• Its off a few pixels here and there, but nothing major. If we spent longer it'd be a waste of time.

• So, you can now build uncomplicated GUIs. Cant you?
What do you mean “Look and Feel”

• Java has 3 looks, (by looks I mean how the components look). Windows, Motif, and Metal.

• These are good/bad. If you use them, you end up with a few difficulties when you go cross platform.

• But seeing as its only 2 lines of code I'll give you all a look.
Altering Look and Feel

// First an import statement
import com.sun.java.swing.plaf.windows.WindowsLookAndFeel;
/* And then, in your Frame class (whatever your base container is (calcFrame.java in this case) */

try
{
// This line sets the look and feel
UIManager.setLookAndFeel
("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
}
catch(Exception cnfe) // you have to catch the exceptions
{
System.out.println("Error changing look and feel");
}
// This line tells the application to update itself
SwingUtilities.updateComponentTreeUI(getContentPane());

// and thats it!
Lets look at the results!

If it wasn't for the little java icon, you'd be struggling!
There you have it a (near) perfect clone.
End of Lecture

• So, what did we learn?
  − How to break a user interface into sub components, making it easier to create
  − How to decorate buttons and text fields so they look appropriate
  − How to use Insets to change the padding on components
  − How JMenuBars work
  − GridLayout and FlowLayout
  − How to alter the Look and Feel of an GUI
Your Lab this week...

- Needs to be decided upon.