Lecture 33

We write a program with a GUI for converting temperatures between centigrade and Fahrenheit.

We learn that some GUI components are containers. They are ones that can have other components added to them. Containers have layout managers that determine the positioning of the components the container contains.

We see more examples of listeners, noting in particular that it is possible that a listener will have an instance variable that refers back to the GUI class.


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

public class TempConversionGUI
   extends JPanel
{
   public TempConversionGUI()
   {  super();
      fahrLabel =
         new JLabel(DEGREES_FAHRENHEIT);
      fahrField =
         new JTextField(INIT_FAHR, TFCOLS);
      centLabel =
         new JLabel(DEGREES_CENTIGRADE);
      centField =
         new JTextField(INIT_CENT, TFCOLS);
      compFahrBtn =
         new JButton(COMPUTE_FAHRENHEIT);
      compCentBtn =
         new JButton(COMPUTE_CENTIGRADE);
```



```
   // Add components to container
   add(fahrLabel);
   add(fahrField);
   add(centLabel);
   add(centField);
   add(compFahrBtn);
   add(compCentBtn);

   // Register listeners
   compFahrBtn.addActionListener(
      new CompFahrBtnLstnr(this));
   compCentBtn.addActionListener(
      new CompCentBtnLstnr(this));
}
```



```
   public void setFahrField(int theFahr)
   {
   }

   public int getFahrField()
   {
   }

   // setCentField and getCentField similarl

   private JLabel fahrLabel;
   private JTextField fahrField;
   private JLabel centLabel;
   private JTextField centField;
   private JButton compFahrBtn;
   private JButton compCentBtn;
```



```
   public static final String
    DEGREES_FAHRENHEIT = "Degrees Fahrenheit
   public static final String
    DEGREES_CENTIGRADE = "Degrees Centigrade
   public static final String
    COMPUTE_FAHRENHEIT = "Compute Fahrenheit
   public static final String
    COMPUTE_CENTIGRADE = "Compute Centigrade

   private static final String
      INIT_FAHR = "212";
   private static final String
      INIT_CENT = "100";
   private static final int TFCOLS = 6;
```



```
   public static void main(String[] args)
   {  TempConversionGUI gui =
               new TempConversionGUI();
      JFrame frm = new JFrame(
            "Temperature Conversions");
      Container contentPane =
                  frm.getContentPane();
      contentPane.add(gui);
      frm.addWindowListener(
             new GenericWindowLstnr());
      frm.setBounds(50, 75, 250, 150);
      frm.setVisible(true);
   }
}
```

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

public class CompFahrBtnLstnr
   implements ActionListener
{
  public CompFahrBtnLstnr(
        TempConversionGUI theGUI)
  {
  }

  public void actionPerformed(ActionEvent ae
  {



  }

  private TempConversionGUI gui;
}
```

`CompCentLstnr` will be very similar.


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

public class GenericWindowLstnr
   implements WindowListener
{
  public GenericWindowLstnr()
  {
  }

  public void windowOpened(WindowEvent we)
  {
  }

  public void windowIconified(WindowEvent we
  {
  }

  // Likewise windowActivated, windowClosed
  // windowDeactivated, windowDeiconified &

  public void windowClosing(WindowEvent we)
  {  System.exit(0);
  }
}
```


Components in our GUI

- Need a subclass of `JPanel`, 2 `JButton`s &...
- `javax.swing.JLabel`
  - allows display of user-uneditable text/images
  - constructor can specify initial text
  - methods `setText`, `getText`, etc.
- `javax.swing.JTextField`
  - allows display, entry & edit of a single line of text (`String`s!)
  - constructor can specify initial value & no. of columns
  - methods `setText`, `getText`, etc.

GUI containers

- In GUI toolkits, such as AWT & Swing, some components are *containers*
  - can have other components added to them
  - e.g. a `JPanel` is a container: it can have, e.g., `JButton`s added to it
  - a container can even have other containers added to it
  - e.g. a `JFrame` is a container and it can have, e.g., two `JPanel`s added to it (to its content pane) and the `JPanel`s are containers that could contain `JButton`s
  - (in fact, in Swing, *all* components are containers — a major increase in flexibility!)

Layout managers

- You can position components in containers using co-ordinates (`setLocation`) but this is insensitive to different platforms and window resizing
- Each container has an instance variable which holds a *layout manager* object, which you can get and set (`get/setLayout`)
- When you add objects to a container, the layout manager positions them and sizes them for you
- `FlowLayout` (the default for `JPanel`) places components sequentially, filling each line from left-to-right and filling the lines from top-to-bottom
  (Very simple, but not usually the best approach)
- There is also `BorderLayout`, `CardLayout`, `GridLayout`, `GridBagLayout` & `BoxLayout` and you could define your own (hard!)

The listeners

- In this program, only the buttons have listeners registered with them (we could have registered listeners with the textfields so that pressing 'Enter' invokes a handler method)
- There's a clean separation of concerns between the GUI class (handles the I/O) and the classes that handle 'real' processing
  — this is achieved by giving the listeners a reference to the GUI class so they can send messages to the GUI class
- We've not done anything about handling illegal user input
- Windows, such as `JFrame`, generate window events (`WindowEvent`); they need a window listener (`addWindowListener`) which implements `WindowListener` (7 methods)