Tutorial #6 (for week 9 - March 5, 2004)

6th March 2004

Unfortunately, most of these questions are directly from previous year's tutorials (instead of exams or NEW questions), so the answers are freely available here and there :(

Q1: Code a "*ToolTip*" in Java/Swing.

Answer: *Something like this:*

CODE LISTING **minimaltool.java**

```
public class minimaltool extends javax.swing.JFrame {
    public minimaltool() {
        initComponents();
    }
    private void initComponents() {
        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });
        jButton1.setToolTipText("Hi there!");
        jButton1.setText("HoHo");
        getContentPane().add(jButton1, java.awt.BorderLayout.WEST);
        jButton2.setText("HoHo");
        getContentPane().add(jButton2, java.awt.BorderLayout.CENTER);
        pack();
    }
    private void exitForm(java.awt.event.WindowEvent evt) {
        System.exit(0);
    }
    public static void main(String args[]) {
        new minimaltool().show();
    }
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
}
```

Q2: Code a "*ToolTip*" for a Tk canvas, which puts up an informative box when you hover over any item on the canvas.

Answer: *My copy of Tk has balloon/tooltips already, but different versions don't. I hunted around and found some Tcl/Tk code at:* **http://www.multimania.com/droche/tkballoon/**. *The code tracks the mouse entering and leaving an item. Some versions of Tk allow you to just add a* **-tip "XXX"** *option to every widget. The following source code for a small program for balloons for Canvas widgets is based on Daniel Roche's tkballoon code:*

CODE LISTING **sample.tcl** Page 1/1

```
#!/usr/bin/wish

source "./balloon.tcl"
canvas .c; pack .c;
.c create oval 50 50 100 100 -fill red -tag oval1
.c create rectangle 150 150 200 250 -fill blue -tag rect1
.c bind oval1 <Enter> "balloon %W OvalHello %X %Y"
.c bind oval1 <Leave> "kill_balloon"
.c bind rect1 <Enter> "balloon %W RectangleHello %X %Y"
.c bind rect1 <Leave> "kill_balloon"
```

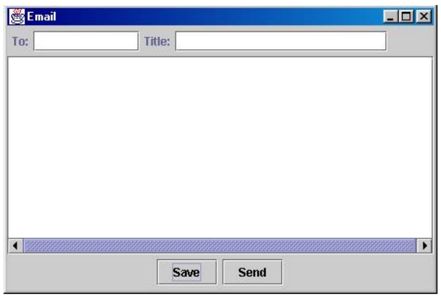
And the balloon routines are:

CODE LISTING **balloon.tcl** Page 1/1

```
proc kill_balloon {} {
    if {[winfo exists .balloon] == 1} {
        destroy .balloon
    }
}

proc balloon {target message {cx 0} {cy 0} } {
        if { $cx == 0 && $cy == 0 } {
            set x [expr [winfo rootx $target] + ([winfo width $target]/2)]
            set y [expr [winfo rooty $target] + [winfo height $target] + 4]
        } else {
            set x [expr $cx + 4]
            set y [expr $cy + 4]
        }
        toplevel .balloon -bg black -screen [winfo screen $target]
        wm overrideredirect .balloon 1
        label .balloon.l \
            -text $message -relief flat \
            -bg #ffffaa -fg black -padx 2 -pady 0 -anchor w
        pack .balloon.l -side left -padx 1 -pady 1
        wm geometry .balloon +${x}+${y}
}
```

Q3: Give Java/Swing layout management code for the following:



Answer: *Something like this:*

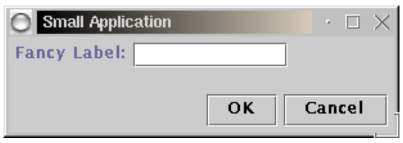
CODE LISTING **TextAreaFrame.java**

```
public class TextAreaFrame extends javax.swing.JFrame {
   public TextAreaFrame () {
      initComponents();
   }
    private void initComponents() {
      javax.swing.JPanel p = new javax.swing.JPanel ();
      saveButton = new javax.swing.JButton ("Save");
      p.add (saveButton);
      sendButton = new javax.swing.JButton ("Send");
      p.add (sendButton);
      getContentPane ().add (p, "South");
      textArea = new javax.swing.JTextArea (8, 40);
      scrollPane = new javax.swing.JScrollPane (textArea);
      getContentPane ().add (scrollPane, "Center");
      javax.swing.JPanel pn = new javax.swing.JPanel ();
      pn.setLayout (new java.awt.FlowLayout (java.awt.FlowLayout.LEFT));
      pn.add (new javax.swing.JLabel ("To:"));
      l_to = new javax.swing.JTextField (10);
      pn.add (l_to);
      pn.add (new javax.swing.JLabel ("Title:"));
      l_title = new javax.swing.JTextField (20);
      pn.add (l_title);
      getContentPane ().add (pn, "North");
      setTitle ("Email");
      setSize (450, 300);
      addWindowListener (new java.awt.event.WindowAdapter () {
         public void windowClosing (java.awt.event.WindowEvent evt) {
            System.exit (0);
         }
      });
   }
   public static void main (String[]args) {
      new TextAreaFrame().show ();
   }
   private javax.swing.JScrollPane scrollPane;
   private javax.swing.JTextArea textArea;
   private javax.swing.JTextField l_to, l_title;
   private javax.swing.JButton saveButton, sendButton;
}
```

Q4: The **`javax.swing.UIManager`** class is used to manipulate the look-and-feel of an application. How can you discover which look-and-feel strategies are implemented in the Java environment?

Answer: *This was just an exercise in looking at the API. The class* **`UIManager.LookAndFeelInfo`** can instantiate **`UIManager.LookAndFeelInfo`** *objects. A method from* **`UIManager`** *class* **`getInstalledLookAndFeels()`** *returns all the look-and-feel strategies installed in the jdk being used. Thanks to Fong Qiyue for this clarification.*

Q5: Write the Java/Swing code for a small application with two buttons, and a text entry box with a label, laid out as shown in this image.



Answer: *Something like this:*

CODE LISTING **SmallFrame.java**

```
public class SmallFrame extends javax.swing.JFrame {
   public SmallFrame () {
      initComponents();
   }
    private void initComponents() {
      javax.swing.JPanel p = new javax.swing.JPanel ();
      p.setLayout (new java.awt.FlowLayout (java.awt.FlowLayout.RIGHT));
      OKButton = new javax.swing.JButton ("OK");
      p.add (OKButton);
      cancelButton = new javax.swing.JButton ("Cancel");
      p.add (cancelButton);
      getContentPane ().add (p, "South");
      javax.swing.JPanel pn = new javax.swing.JPanel ();
      pn.setLayout (new java.awt.FlowLayout (java.awt.FlowLayout.LEFT));
      pn.add (new javax.swing.JLabel ("Fancy Label:"));
      l_to = new javax.swing.JTextField (10);
      pn.add (l_to);
      getContentPane ().add (pn, "North");
      setTitle ("Small Application");
       setSize (300, 75);
      addWindowListener (new java.awt.event.WindowAdapter () {
         public void windowClosing (java.awt.event.WindowEvent evt) {
            System.exit (0);
         }
      });
   }
   public static void main (String[]args) {
      new SmallFrame().show ();
   }
   private javax.swing.JScrollPane scrollPane;
   private javax.swing.JTextArea textArea;
   private javax.swing.JTextField l_to, l_title;
   private javax.swing.JButton OKButton, cancelButton;
}
```