GUI Programming with Swing

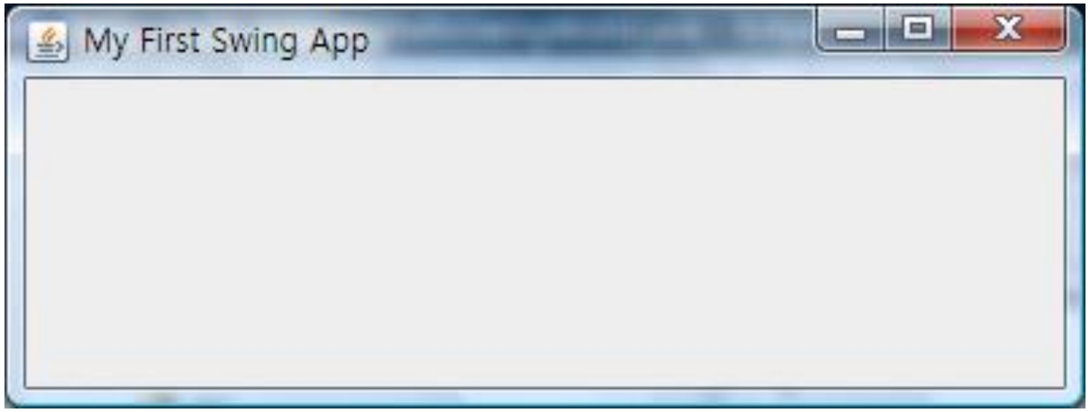
Core Java Volume I – Fundamentals

- Chapter 10. Graphics Programming
- Chapter 11. Event Handling
- Chapter 12. User Interface Components with Swing

Swing

- **Swing** is a widget toolkit for Java.
- It is part of Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.
- Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT(Abstract Window Toolkit).
- It can be compared with MFC and WinForm in MS Windows Platform.

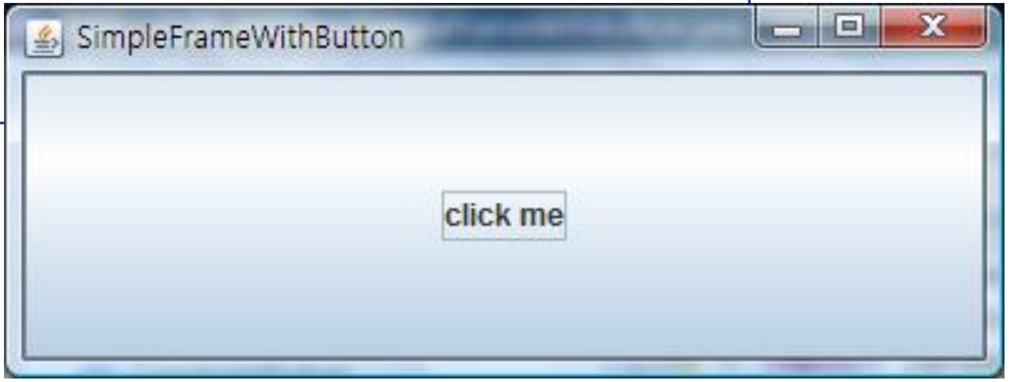
Your First Swing Application



```
import javax.swing.JFrame;

public class HelloSwingWorld {
  public static void main(String[] args) {
     JFrame frame = new JFrame("My First Swing App");
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setVisible(true);
     frame.setSize(400, 150) ;
  }
}
```

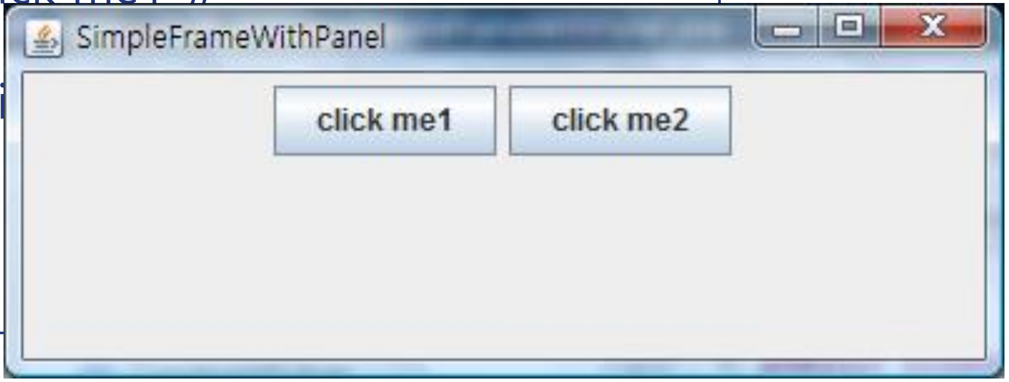
Frame with Button

```
import javax.swing.*;

public class HelloSwingWorld {
  public static void main(String[] args) {
     JFrame frame = new JFrame("SimpleFrameWithButton");
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setVisible(true);
     frame.setSize(400, 150) ;

     JButton button = new JButton("click me");
     frame.add(button);
  }
}
```



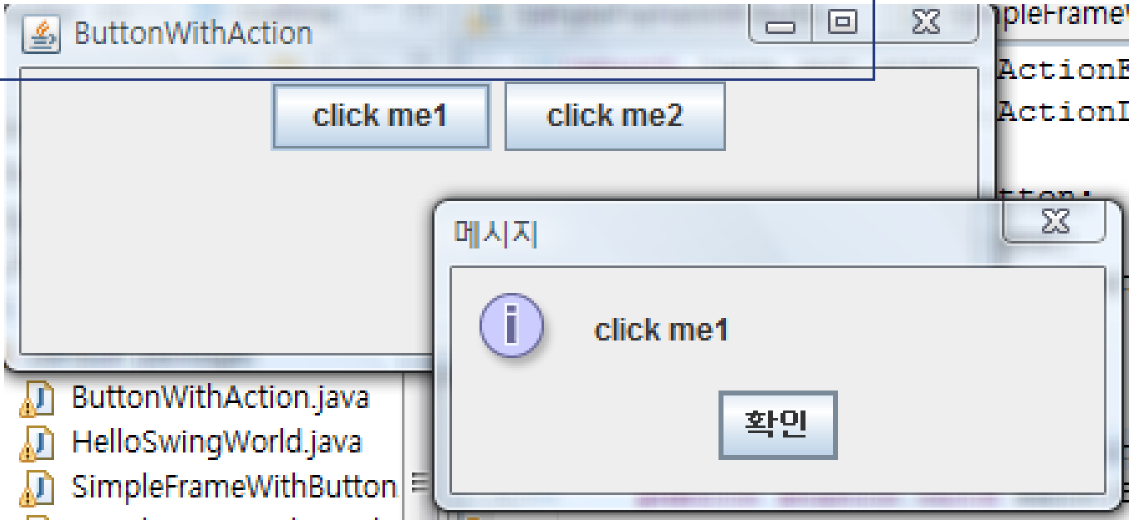
Frame with Panel

```
import javax.swing.*;

public class HelloSwingWorld {
  public static void main(String[] args) {
     JFrame frame = new JFrame("SimpleFrameWithButton");
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setVisible(true);
     frame.setSize(400, 150) ;

     JPanel panel = new JPanel() ;
     frame.add(panel) ;
     JButton button1 = new JButton("click me1");
     panel.add(button1);
     JButton button2 = new JButton("cli
     panel.add(button2);
  }
}
```



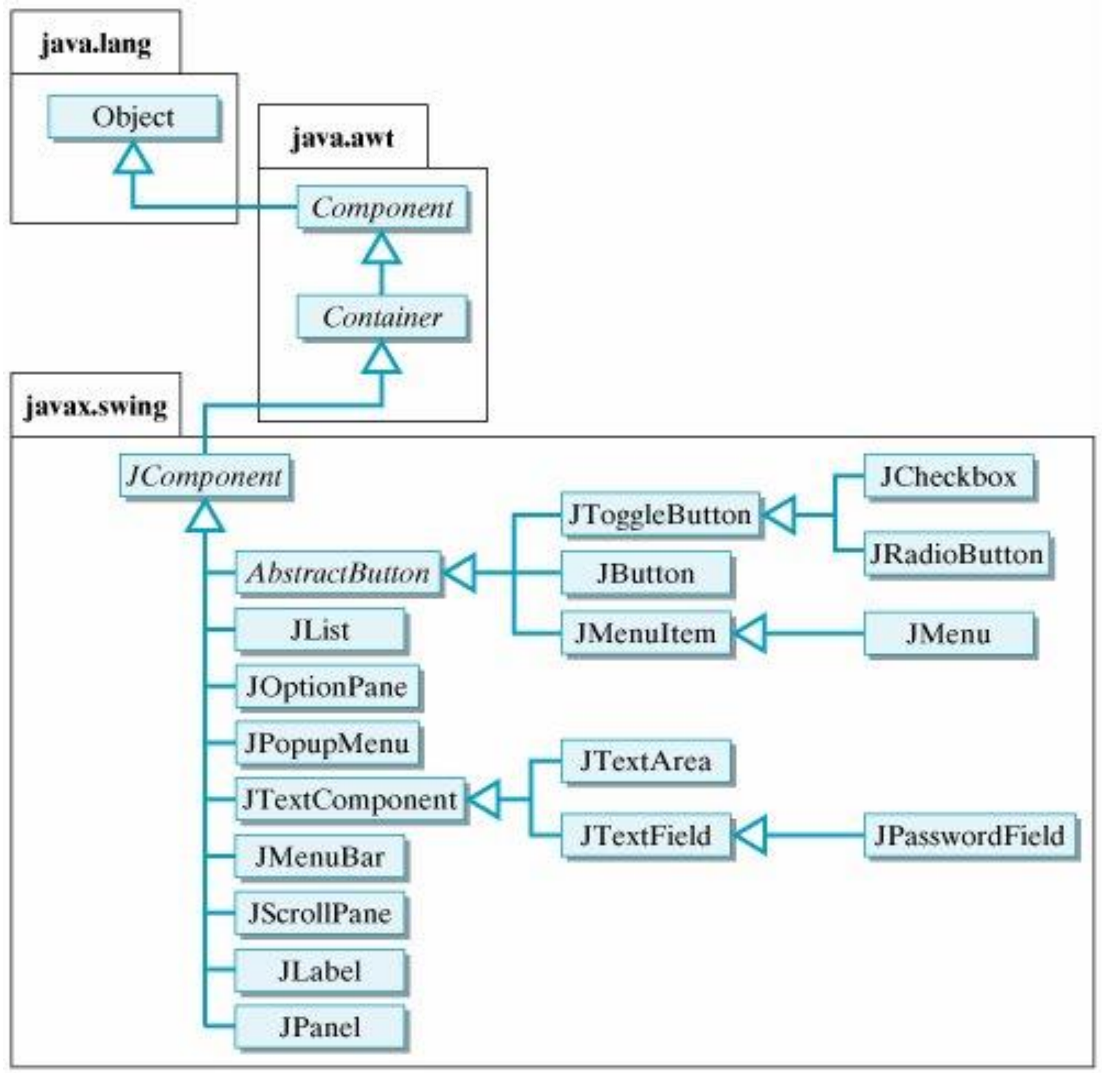
Button with Handler

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class ButtonWithAction {
    public static void main(String[] args) {
        MyFrame frame = new MyFrame("ButtonWithAction");
    }
}
```

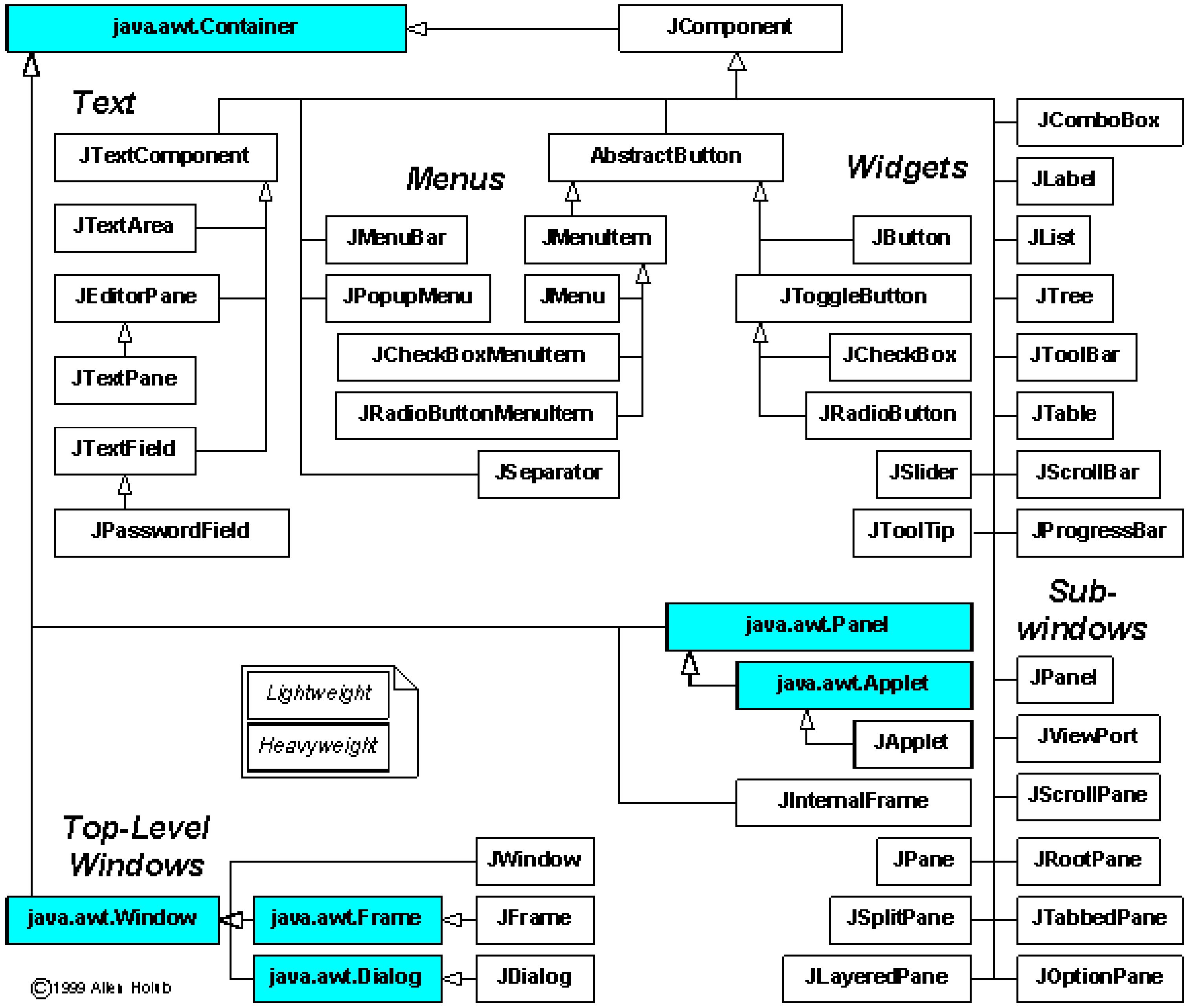


```
class MyFrame extends JFrame implements ActionListener {
    public MyFrame(String title) {
        setTitle(title) ;           setVisible(true);    setSize(400, 150) ;

        JPanel panel = new JPanel() ;
        add(panel) ;
        JButton button1 = new JButton("click me1");
        button1.addActionListener(this) ;
        panel.add(button1);
        JButton button2 = new JButton("click me2");
        button2.addActionListener(this) ;
        panel.add(button2);
    }

    public void actionPerformed(ActionEvent event) {
        System.out.println(event) ;

        String cmd = event.getActionCommand() ;
        //String cmd = ((JButton) event.getSource()).getText() ;

        JOptionPane.showMessageDialog(null, cmd) ;
    }
}
```

Swing Classes



Swing Classes: Details



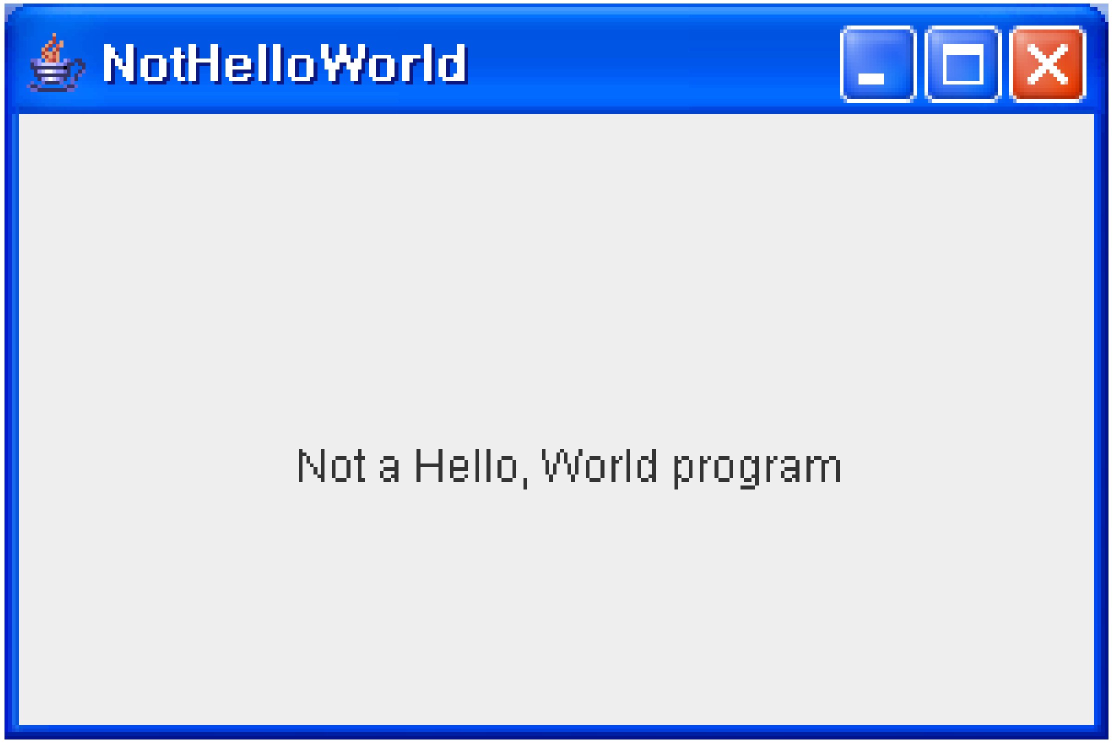
Creating & Positioning a Frame

- A frame window so that
  - Its area is one-fourth that of the whole screen
  - It is centered in the middle of the screen

```
import java.awt.*;

import javax.swing.*;

public class CenteredFrameTest {
   public static void main(String[] args) {
      CenteredFrame frame = new CenteredFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

```
class CenteredFrame extends JFrame {
   public CenteredFrame() {
      // get screen dimensions

      Toolkit kit = Toolkit.getDefaultToolkit(); // java.awt.Toolkit
      Dimension screenSize = kit.getScreenSize();
      int screenHeight = screenSize.height;
      int screenWidth = screenSize.width;

      // center frame in screen

      setSize(screenWidth / 2, screenHeight / 2);
      setLocation(screenWidth / 4, screenHeight / 4);

      // set frame icon and title

      Image img = kit.getImage("icon.gif");
      setIconImage(img);
      setTitle("CenteredFrame");
   }
}
```

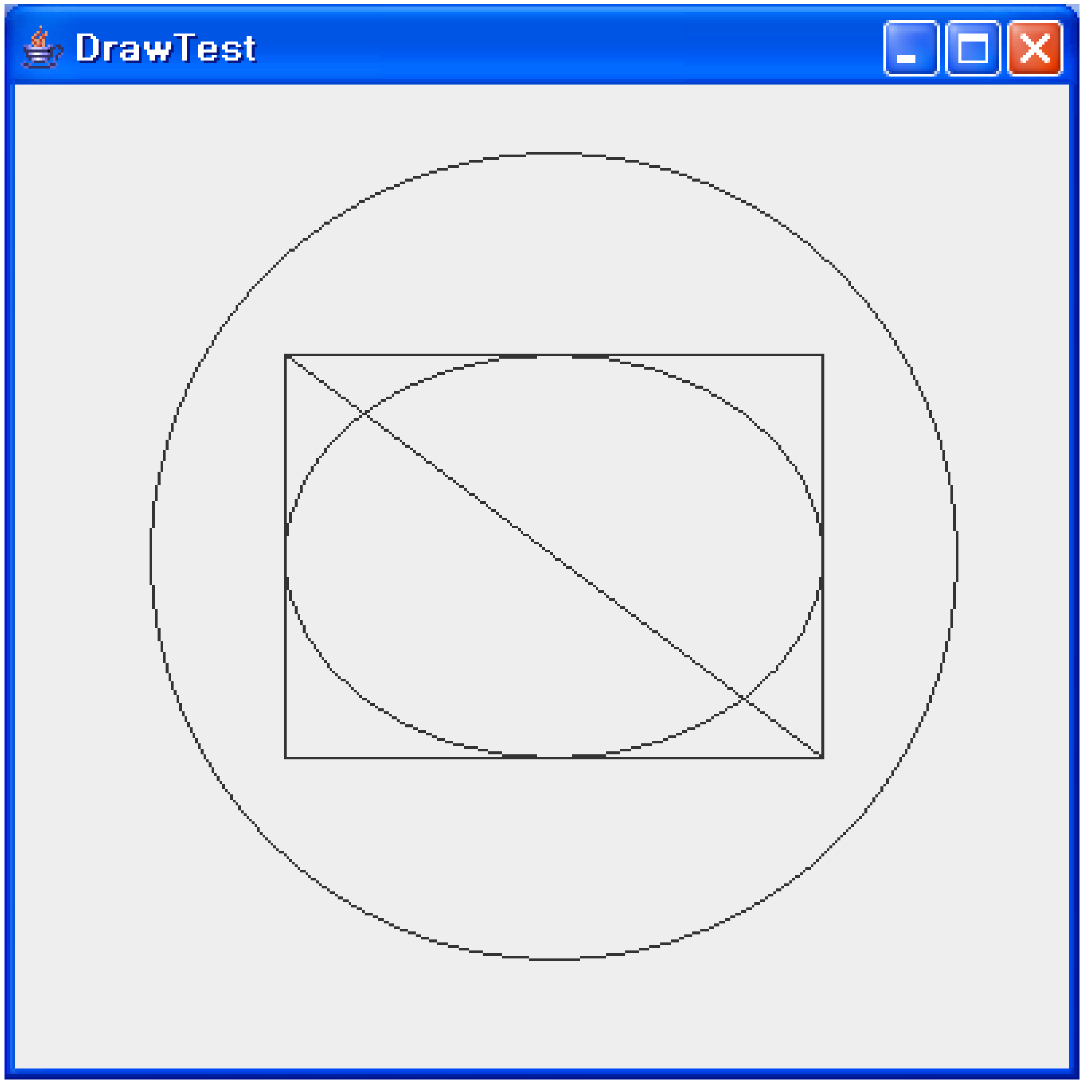
Displaying Information in a Panel



```
Not a Hello, World program
```

```
import javax.swing.*;
import java.awt.*;
public class NotHelloWorld {
   public static void main(String[] args) {
      NotHelloWorldFrame frame = new NotHelloWorldFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class NotHelloWorldFrame extends JFrame {
   public NotHelloWorldFrame() {
      setTitle("NotHelloWorld");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      // add panel to frame
      NotHelloWorldPanel panel = new NotHelloWorldPanel();
      add(panel);
   }
   public static final int DEFAULT_WIDTH = 300, DEFAULT_HEIGHT = 200 ;
}
class NotHelloWorldPanel extends JPanel {
   public void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.drawString("Not a Hello, World program", MESSAGE_X, MESSAGE_Y);
   }
   public static final int MESSAGE_X = 75, MESSAGE_Y = 100;
}
```

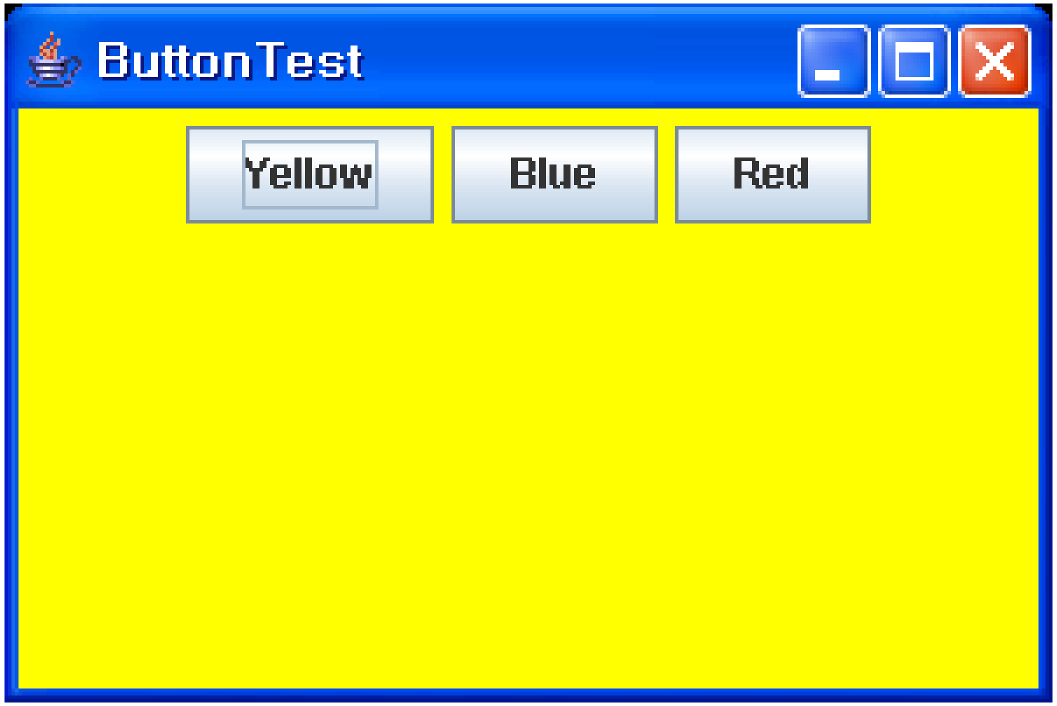
Working with 2D Shapes



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

public class DrawTest {
   public static void main(String[] args) {
      DrawFrame frame = new DrawFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class DrawFrame extends JFrame {
   public DrawFrame() {
      setTitle("DrawTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      DrawPanel panel = new DrawPanel();
      add(panel);
   }
   public static final int DEFAULT_WIDTH = 400;
   public static final int DEFAULT_HEIGHT = 400;
}
```

```
class DrawPanel extends JPanel {
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2 = (Graphics2D) g;

    // draw a rectangle
    double leftX = 100;
    double topY = 100;
    double width = 200;
    double height = 150;
    Rectangle2D rect = new Rectangle2D.Double(leftX, topY, width, height);
    g2.draw(rect);

    // draw the enclosed ellipse
    Ellipse2D ellipse = new Ellipse2D.Double();
    ellipse.setFrame(rect);
    g2.draw(ellipse);

    // draw a diagonal line
    g2.draw(new Line2D.Double(leftX, topY, leftX + width, topY + height));

    // draw a circle with the same center
    double centerX = rect.getCenterX();
    double centerY = rect.getCenterY();
    double radius = 150;

    Ellipse2D circle = new Ellipse2D.Double();
    circle.setFrameFromCenter(centerX, centerY, centerX + radius, centerY + radius);
    g2.draw(circle);
  }
}
```

Basics of Event Handling



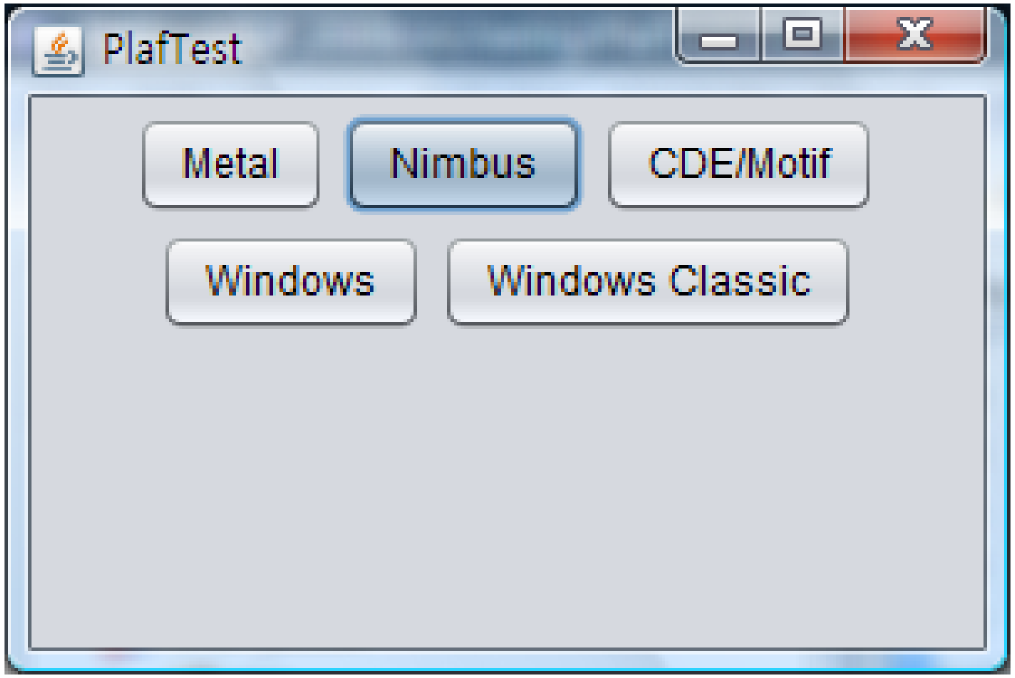
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ButtonTest {
   public static void main(String[] args) {
      ButtonFrame frame = new ButtonFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

class ButtonFrame extends JFrame {
   public ButtonFrame() {
      setTitle("ButtonTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

      ButtonPanel panel = new ButtonPanel();
      add(panel);
   }
   public static final int DEFAULT_WIDTH = 300;
   public static final int DEFAULT_HEIGHT = 200;
}
```

```
class ButtonPanel extends JPanel {
  public ButtonPanel() {
    // create buttons
    JButton yellowButton = new JButton("Yellow");
    JButton blueButton = new JButton("Blue");
    JButton redButton = new JButton("Red");

    // add buttons to panel
    add(yellowButton); add(blueButton); add(redButton);

    // create button actions
    ColorAction yellowAction = new ColorAction(Color.YELLOW);
    ColorAction blueAction = new ColorAction(Color.BLUE);
    ColorAction redAction = new ColorAction(Color.RED);

    // associate actions with buttons
    yellowButton.addActionListener(yellowAction);
    blueButton.addActionListener(blueAction);
    redButton.addActionListener(redAction);
  }
  private class ColorAction implements ActionListener {
    public ColorAction(Color c) { backgroundColor = c; }
    public void actionPerformed(ActionEvent event) {
       setBackground(backgroundColor);
    }
    private Color backgroundColor;
  }
}
```

```
class ButtonPanel extends JPanel implements ActionListener {
  …
  public void actionPerformed(ActionEvent  event) {
     Object source = event.getSource() ;
     if ( source == yellowButton ) {
         setBackground(Color.YELLOW) ;
     …
  }
  …
  yellowButton.addActionListener(this) ;
  …
```

```
class ButtonPanel extends JPanel implements ActionListener {
  …
  public void actionPerformed(ActionEvent  event) {
     String command = event.getActionCommand() ;
     if ( command.equals("Yellow") ) {
         setBackground(Color.YELLOW) ;
     …
  }
  …
  yellowButton.addActionListener(this) ;
  …
```

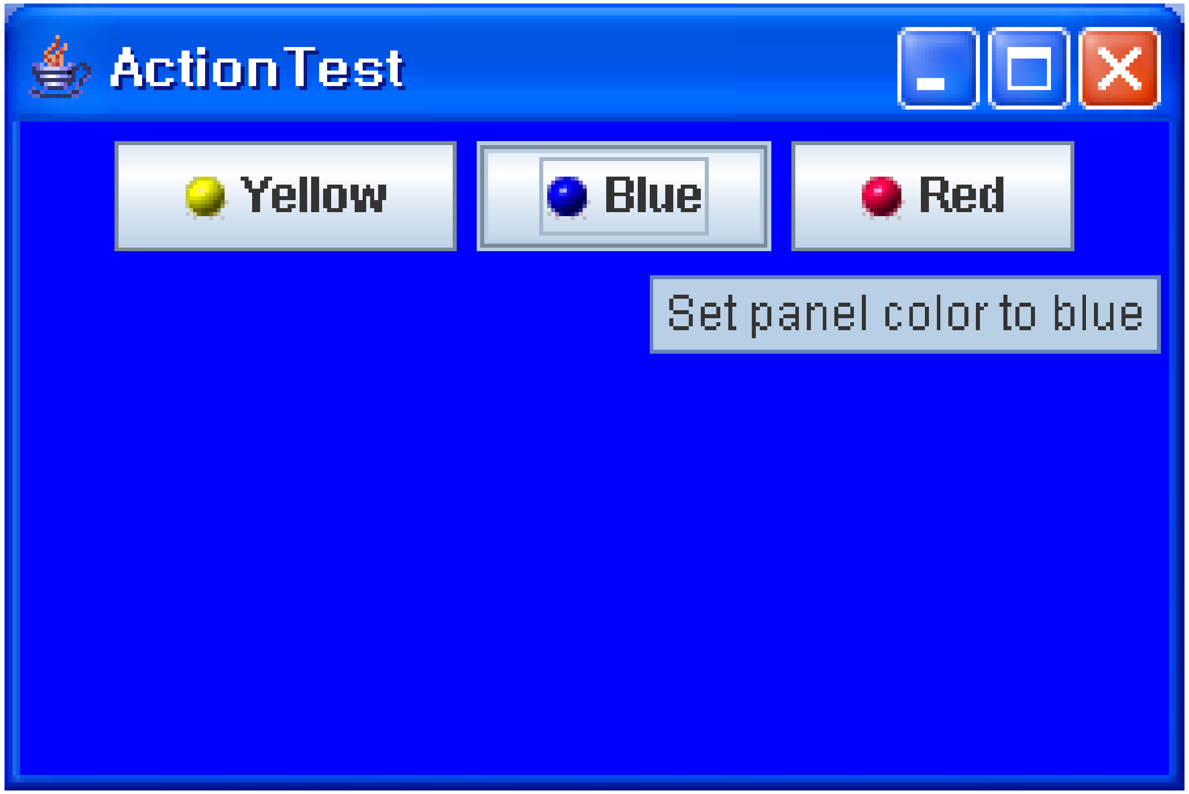
Changing the Look and Feel



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

public class PlafTest {
   public static void main(String[] args) {
      PlafFrame frame = new PlafFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class PlafFrame extends JFrame {
   public PlafFrame() {
      setTitle("PlafTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      PlafPanel panel = new PlafPanel();
      add(panel);
   }
   public static final int DEFAULT_WIDTH = 300;
   public static final int DEFAULT_HEIGHT = 200;
}
```

```
class PlafPanel extends JPanel {
  public PlafPanel() {
    UIManager.LookAndFeelInfo[] infos = UIManager.getInstalledLookAndFeels();
    // javax.swing.UIManager: keeps track of the current look and feel and its defaults

    for (UIManager.LookAndFeelInfo info : infos)
      makeButton(info.getName(), info.getClassName());
  }

  void makeButton(String name, final String plafName) {
    // add button to panel
    JButton button = new JButton(name);
    add(button);

    // set button action
    button.addActionListener(new  ActionListener() {
        public void actionPerformed(ActionEvent event) {
          // button action: switch to the new look and feel
          try {
            UIManager.setLookAndFeel(plafName);
            SwingUtilities.updateComponentTreeUI(PlafPanel.this);
          } catch(Exception e) { e.printStackTrace(); }
        }
      });
  }
}
```

Actions



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

public class ActionTest {
   public static void main(String[] args) {
      ActionFrame frame = new ActionFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class ActionFrame extends JFrame {
   public ActionFrame() {
      setTitle("ActionTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      ActionPanel panel = new ActionPanel();
      add(panel);
   }
   public static final int DEFAULT_WIDTH = 300;
   public static final int DEFAULT_HEIGHT = 200;
}
```

```
class ActionPanel extends JPanel {
   public ActionPanel() {
      // define actions
      // interface javax.swing.Action
      Action yellowAction =
          new ColorAction("Yellow", new ImageIcon("yellow-ball.gif"), Color.YELLOW);
                              // name, icon, color

      Action blueAction =
          new ColorAction("Blue", new ImageIcon("blue-ball.gif"), Color.BLUE);

      Action redAction =
          new ColorAction("Red", new ImageIcon("red-ball.gif"), Color.RED);

      // add buttons for these actions
      add(new JButton(yellowAction));
      add(new JButton(blueAction));
      add(new JButton(redAction));
      // JButton(Action a):
      // Creates a button where properties are taken from the Action supplied.
```

```
  // javax.swing.InputMap and javax.swing.ActionMap
  InputMap imap =
     getInputMap(JComponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT);

  // public static final InputMap JComponent.getInputMap(int condition)
  // Returns the InputMap that is used during condition.
  // condition - one of WHEN_IN_FOCUSED_WINDOW, WHEN_FOCUSED,
  //                    WHEN_ANCESTOR_OF_FOCUSED_COMPONENT

  imap.put(KeyStroke.getKeyStroke("ctrl Y"), "panel.yellow");
  imap.put(KeyStroke.getKeyStroke("ctrl B"), "panel.blue");
  imap.put(KeyStroke.getKeyStroke("ctrl R"), "panel.red");
  // void put(KeyStroke keyStroke, Object actionMapKey)
  // Adds a binding for keyStroke to actionMapKey.

  // ActionMap provides mappings from Objects (called keys or Action names) to Actions
  ActionMap amap = getActionMap();
  amap.put("panel.yellow", yellowAction);
  amap.put("panel.blue", blueAction);
  amap.put("panel.red", redAction);
  // put(Object key, Action action): Adds a binding for key to action.
}
```

```
  public class ColorAction extends AbstractAction {
    //AbstractAction implements all methods of interface Action except for actionPerformed

    public ColorAction(String name, Icon icon, Color c) {
      putValue(Action.NAME, name);    // displayed on buttons and menu items
      putValue(Action.SMALL_ICON, icon);
      putValue(Action.SHORT_DESCRIPTION, // for display in a tooltip
          "Set panel color to " + name.toLowerCase());
      putValue("color", c);
    }

    public void actionPerformed(ActionEvent event) {
      Color c = (Color) getValue("color");
      setBackground(c);
    }
  }
}
```

KeyStroke

KeyStroke **getKeyStroke**(int keyCode, int modifiers, boolean onKeyRelease)

The "virtual key" constants defined in **java.awt.event.KeyEvent** can be used to specify the key code. For example:

- java.awt.event.KeyEvent.VK_ENTER
- java.awt.event.KeyEvent.VK_TAB
- java.awt.event.KeyEvent.VK_SPACE

The modifiers consist of any combination of:

- java.awt.event.InputEvent.SHIFT_MASK (1)
- java.awt.event.InputEvent.CTRL_MASK (2)
- java.awt.event.InputEvent.META_MASK (4)
- java.awt.event.InputEvent.ALT_MASK (8)

"INSERT"	getKeyStroke(KeyEvent.VK_INSERT, 0);
"control DELETE"	getKeyStroke(KeyEvent.VK_DELETE, InputEvent.CTRL_MASK);
"alt shift X"	getKeyStroke(KeyEvent.VK_X, InputEvent.ALT_MASK | InputEvent.SHIFT_MASK);
"alt shift released X"	getKeyStroke(KeyEvent.VK_X, InputEvent.ALT_MASK | InputEvent.SHIFT_MASK, true);
"typed a"	getKeyStroke('a');

Java Tutorial: Creating a GUI with JFC/Swing

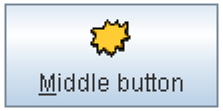
https://docs.oracle.com/javase/tutorial/uiswing/index.html

Using Swing Components: Example

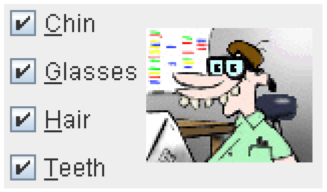
https://docs.oracle.com/javase/tutorial/uiswing/examples/components/index.html

Basic Controls

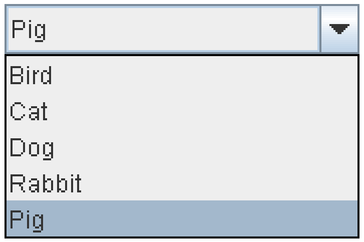
- Simple components get input from the user



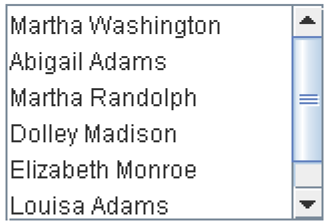
JButton



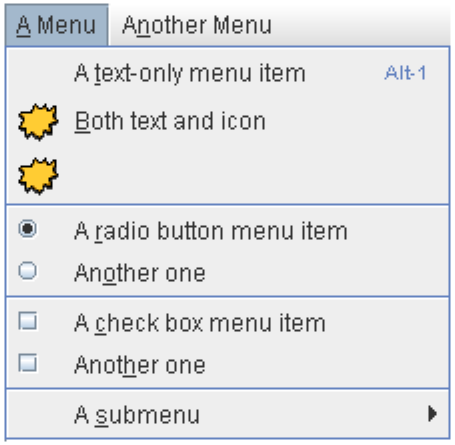
JCheckBox



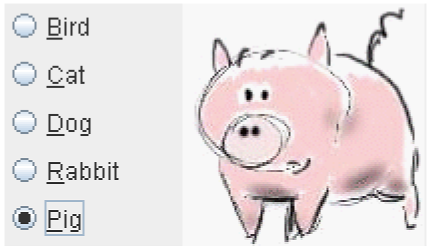
JComboBox



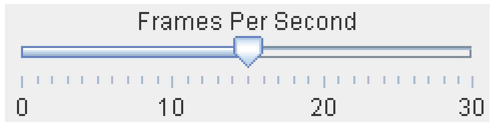
JList



JMenu



JRadioButton



JSlider

Date: 07/2006

JSpinner

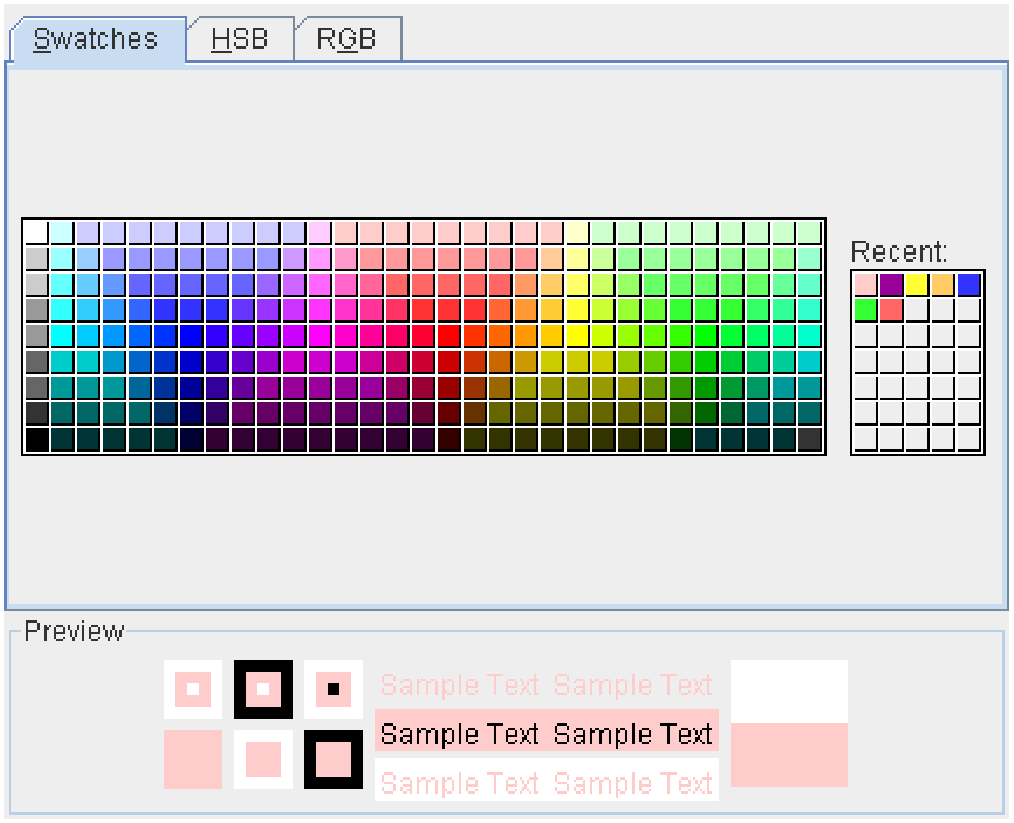
City: Santa Rosa

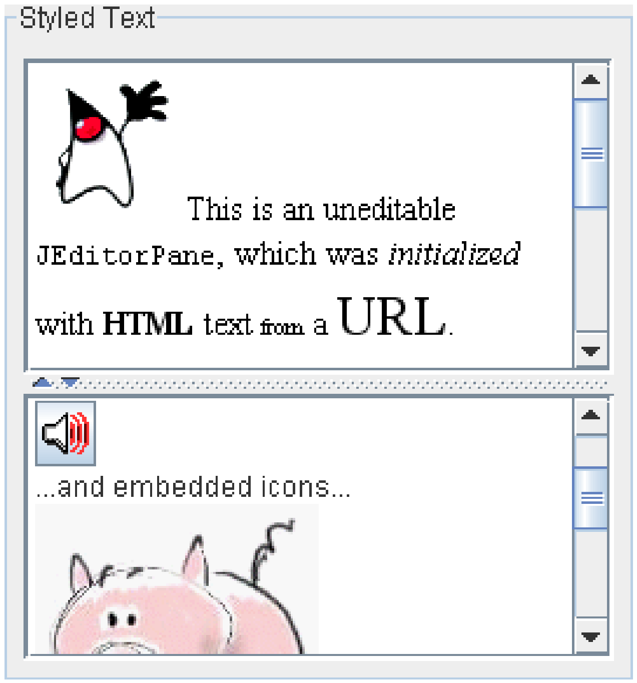
JTextField

Enter the password: •••••••

JPasswordField

Interactive Displays of Highly Formatted Information





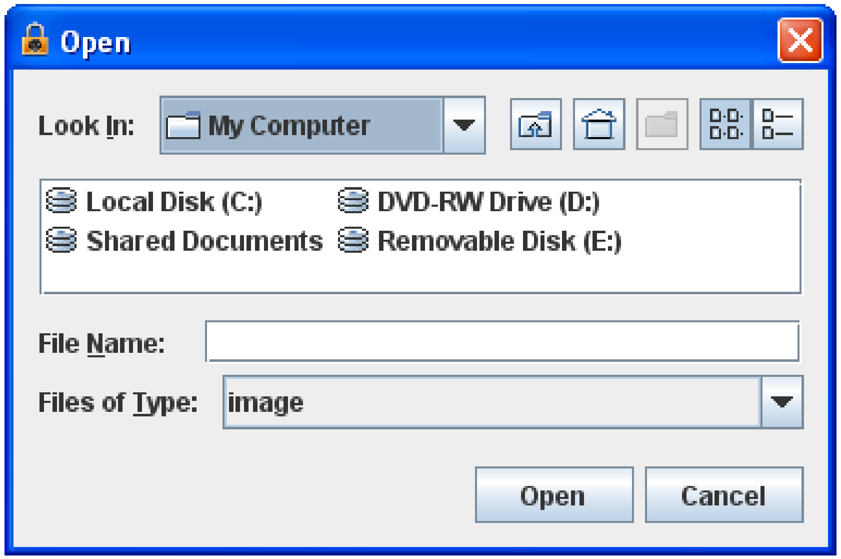
This is an uneditable JEditorPane, which was *initialized* with **HTML** text from a URL.

...and embedded icons...

JColorChooser

JEditorPane and JTextPane

Interactive Displays of Highly Formatted Information



Open

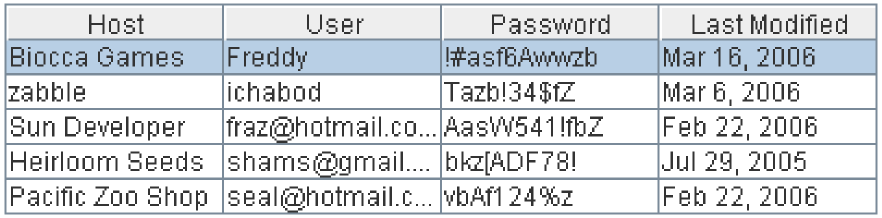
Look In: My Computer

Local Disk (C:)
DVD-RW Drive (D:)
Shared Documents
Removable Disk (E:)

File Name:

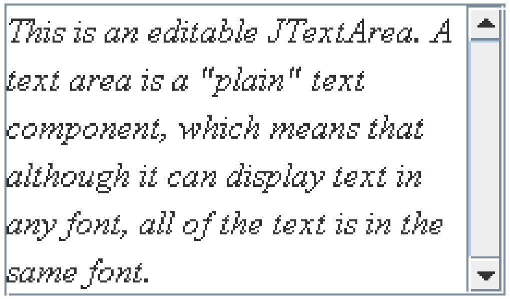
Files of Type: image

JFileChooser



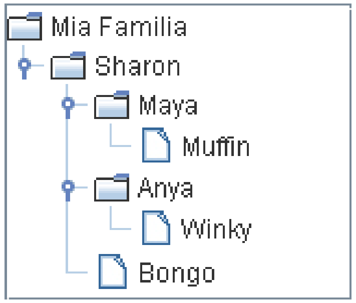
Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail....	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

JTable



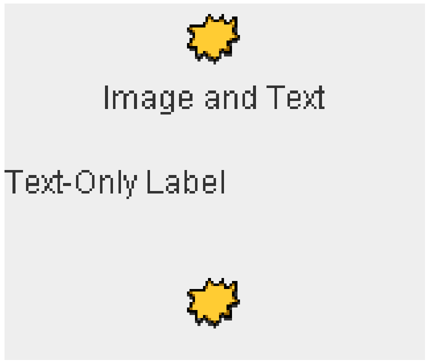
This is an editable JTextArea. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.

JTextArea

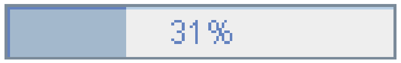


JTree

Uneditable Information Displays

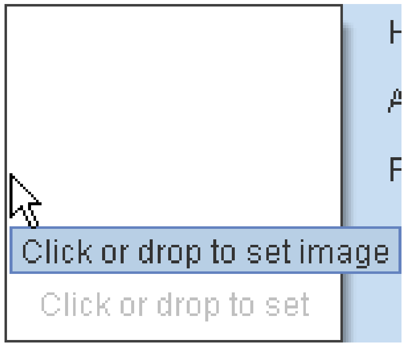


JLabel



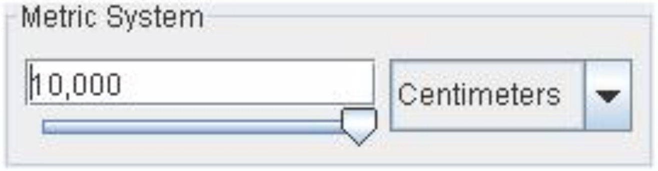
JProgressBar

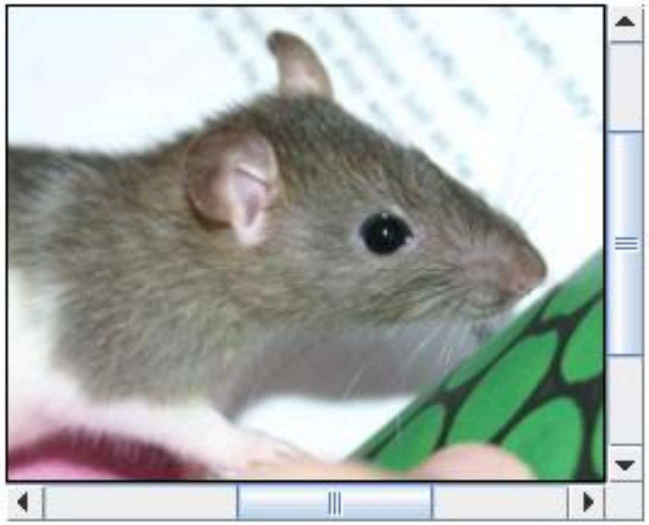
JSeparator



JToolTip

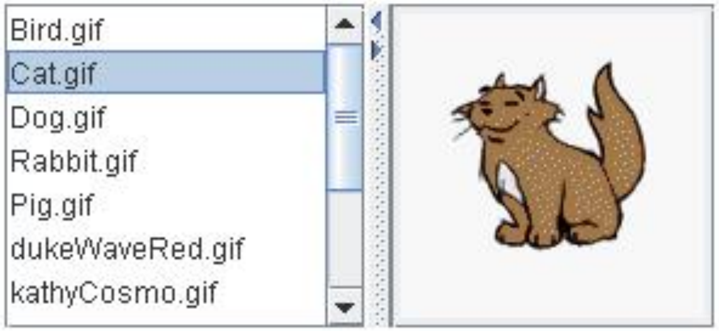
Containers

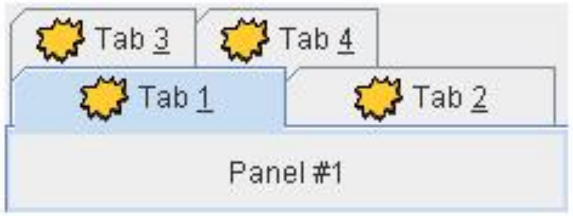




JPanel

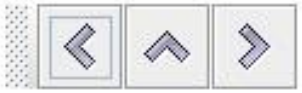
JScrollPane





JSplitPane

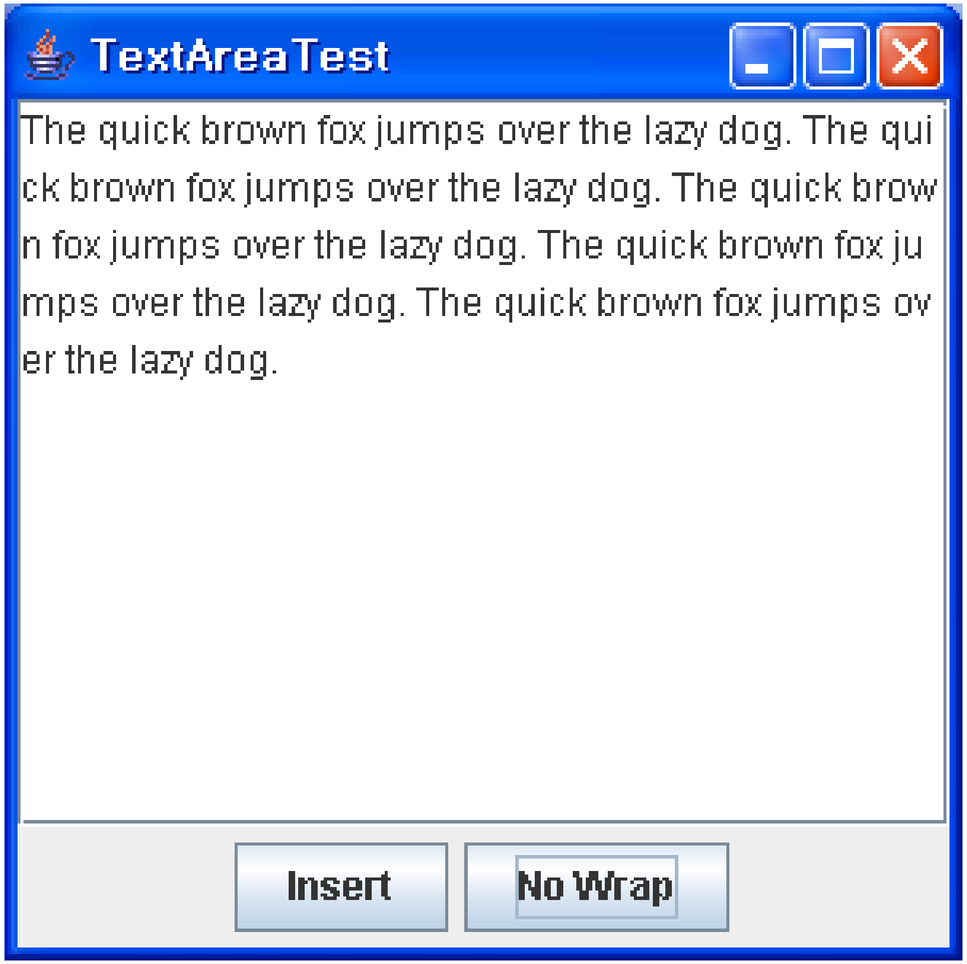
JTabbedPane



JToolBar

Contents

Text Area



The quick brown fox jumps over the lazy dog. The qui
ck brown fox jumps over the lazy dog. The quick brow
n fox jumps over the lazy dog. The quick brown fox ju
mps over the lazy dog. The quick brown fox jumps ov
er the lazy dog.

How to Use Text Areas in Java Tutorial
https://docs.oracle.com/javase/tutorial/uiswing/components/textarea.html

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class TextAreaTest  {
   public static void main(String[] args) {
      TextAreaFrame frame = new TextAreaFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class TextAreaFrame extends JFrame {
   public TextAreaFrame() {
      setTitle("TextAreaTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      buttonPanel = new JPanel();

      JButton insertButton = new JButton("Insert");
      buttonPanel.add(insertButton);
      insertButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent event) {
             textArea.append("The quick brown fox jumps over the lazy dog. ");
          }
       });
```

```
insertButton.addActionListener(
  (ActionEvent event) ->
  textArea.append("The quick brown fox jumps over the lazy dog. ")
);
```

```
    wrapButton = new JButton("Wrap");  buttonPanel.add(wrapButton);
    wrapButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent event) {
          final boolean wrap = !textArea.getLineWrap();
          textArea.setLineWrap(wrap);
          wrapButton.setText(wrap ? "No Wrap" : "Wrap");
       }
     });
```

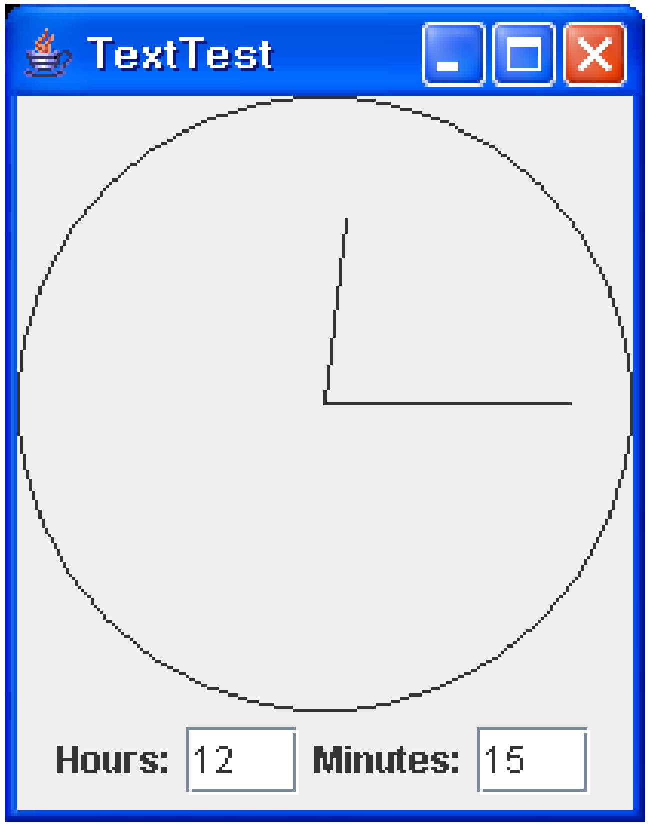
```
wrapButton.addActionListener(
   (ActionEvent event) -> {
      final boolean wrap = !textArea.getLineWrap();
      textArea.setLineWrap(wrap);
      wrapButton.setText(wrap ? "No Wrap" : "Wrap");
   }
);
```

```
    add(buttonPanel, BorderLayout.SOUTH);
    textArea = new JTextArea(8, 40); // JTextArea(int rows, int columns)
    scrollPane = new JScrollPane(textArea);
    add(scrollPane, BorderLayout.CENTER);
  }
  public static final int DEFAULT_WIDTH = 300, DEFAULT_HEIGHT = 300;
  private JTextArea textArea;
  private JScrollPane scrollPane;
  private JPanel buttonPanel;
  private JButton wrapButton;
}
```

Choice Components

- Checkboxes
- Radio Buttons
- Borders
- Combo Boxes
- Sliders

Text Input



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

public class TextTest
{
   public static void main(String[] args)
   {
      TextTestFrame frame = new TextTestFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

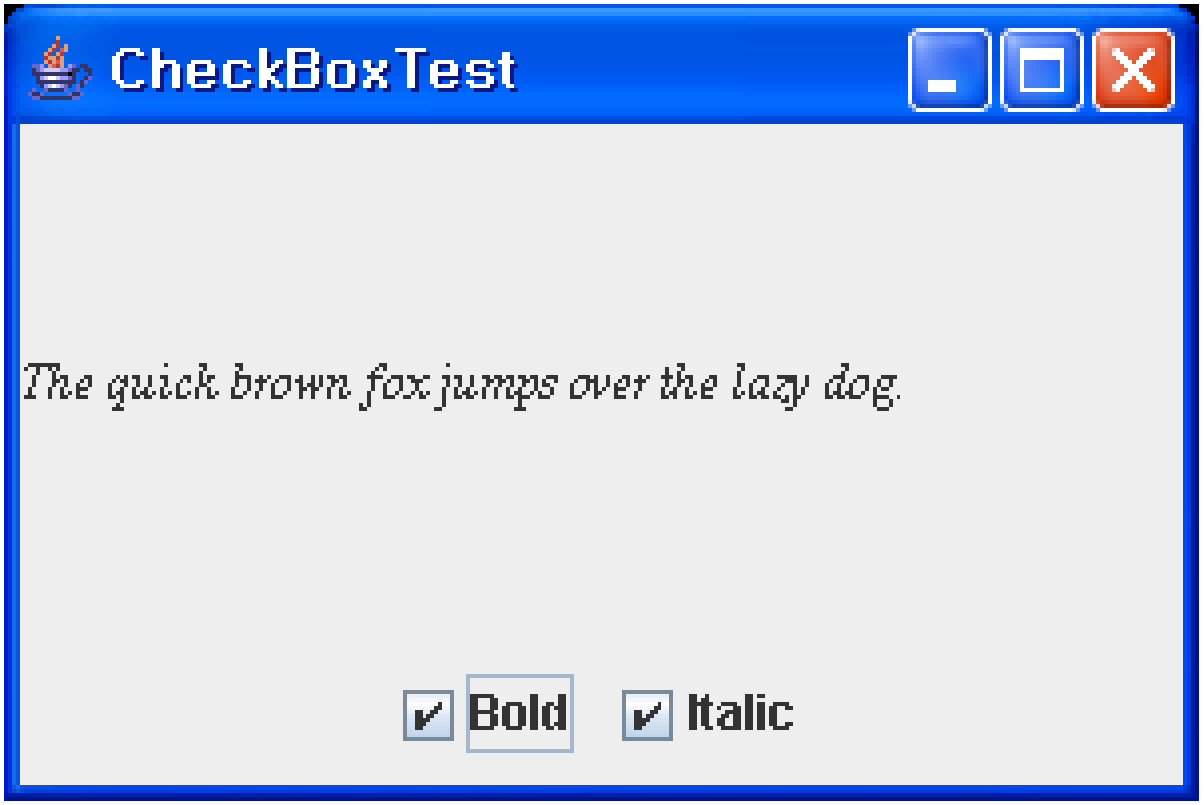
```
class TextTestFrame extends JFrame {
  public TextTestFrame() {
    setTitle("TextTest");
    DocumentListener listener = new ClockFieldListener();
    // interface javax.swing.DocumentListener

    JPanel panel = new JPanel();

    panel.add(new JLabel("Hours:"));
    hourField = new JTextField("12", 3);
    // JTextField(String text, int columns)
    panel.add(hourField);

    hourField.getDocument().addDocumentListener(listener);

    panel.add(new JLabel("Minutes:"));
    minuteField = new JTextField("00", 3);
    panel.add(minuteField);
    minuteField.getDocument().addDocumentListener(listener);

    add(panel, BorderLayout.SOUTH);
    // Note that the default layout manager of the content page is Border Layout

    clock = new ClockPanel();  add(clock, BorderLayout.CENTER);
    pack();
  }
```

you should ask the document to notify you whenever the data have changed

```
  public void setClock() {
    try {
      int hours = Integer.parseInt(hourField.getText().trim());
      int minutes = Integer.parseInt(minuteField.getText().trim());
      clock.setTime(hours, minutes);
    }
    catch (NumberFormatException e) {}
    // don't set the clock if the input can't be parsed
  }
  private JTextField hourField;
  private JTextField minuteField;
  private ClockPanel clock;

  private class ClockFieldListener implements DocumentListener {
    public void insertUpdate(DocumentEvent event) { setClock(); }
    public void removeUpdate(DocumentEvent event) { setClock(); }
    public void changedUpdate(DocumentEvent event) {} // when attributes changed

    // interface javax.swing.DocumentEvent
    // getDocument()
    // getLength(), getOffset(), getType()
  }
}
```

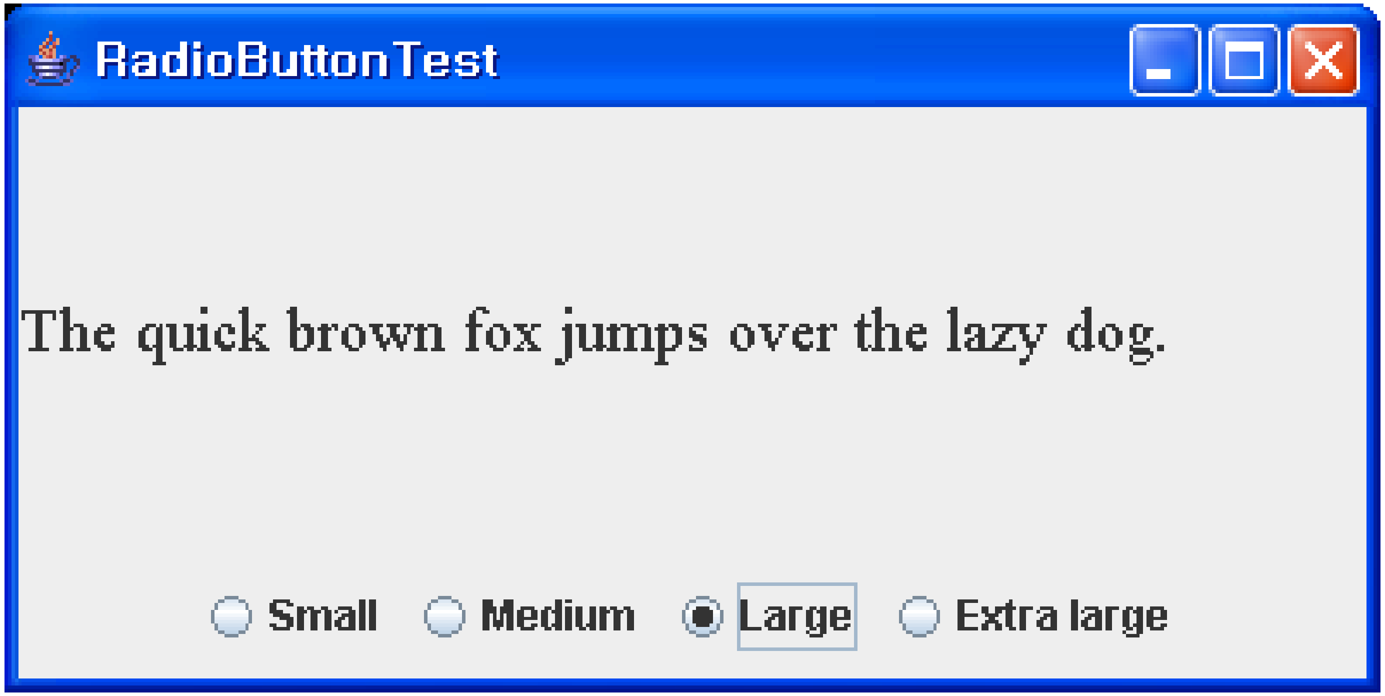
```
class ClockPanel extends JPanel {
  public ClockPanel() {
    setPreferredSize(new Dimension(2 * RADIUS + 1, 2 * RADIUS + 1));
  }
  public void paintComponent(Graphics g) {
    // draw the circular boundary
    super.paintComponent(g);

    Graphics2D g2 = (Graphics2D) g;
    Ellipse2D circle = new Ellipse2D.Double(0, 0, 2 * RADIUS, 2 * RADIUS);
    g2.draw(circle);

    // draw the hour hand

    double hourAngle = Math.toRadians(90 - 360 * minutes / (12 * 60));
    drawHand(g2, hourAngle, HOUR_HAND_LENGTH);

    // draw the minute hand

    double minuteAngle = Math.toRadians(90 - 360 * minutes / 60);
    drawHand(g2, minuteAngle, MINUTE_HAND_LENGTH);
  }
  public void drawHand(Graphics2D g2, double angle, double handLength) {
     Point2D end = new Point2D.Double(
        RADIUS + handLength * Math.cos(angle),
        RADIUS - handLength * Math.sin(angle));
     Point2D center = new Point2D.Double(RADIUS, RADIUS);
     g2.draw(new Line2D.Double(center, end));
  }

  /**
     Set the time to be displayed on the clock
     @param h hours
     @param m minutes
  */
  public void setTime(int h, int m) {
     minutes = h * 60 + m;
     repaint();
  }

  private double minutes = 0;
  private int RADIUS = 100;
  private double MINUTE_HAND_LENGTH = 0.8 * RADIUS;
  private double HOUR_HAND_LENGTH = 0.6 * RADIUS;
}
```

Checkboxes


CheckBoxTest
The quick brown fox jumps over the lazy dog.
Bold
Italic

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class CheckBoxTest {
  public static void main(String[] args) {
    CheckBoxFrame frame = new CheckBoxFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
class CheckBoxFrame extends JFrame  {
  public CheckBoxFrame() {
    setTitle("CheckBoxTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    label = new JLabel("The quick brown fox jumps over the lazy dog.");
    label.setFont(new Font("Serif", Font.PLAIN, FONTSIZE));
    // Font(String name, int style, int size)
    // Creates a new Font from the specified name, style and point size.
    //  GraphicsEnvironment.getAvailableFontFamilyNames()
    add(label, BorderLayout.CENTER);
    ActionListener listener = new ActionListener() {
        public void actionPerformed(ActionEvent event) {
          int mode = 0;
          if (bold.isSelected()) mode += Font.BOLD;
          if (italic.isSelected()) mode += Font.ITALIC;
          label.setFont(new Font("Serif", mode, FONTSIZE));
        }
      };
    JPanel buttonPanel = new JPanel();

    bold = new JCheckBox("Bold");
    bold.addActionListener(listener);
    buttonPanel.add(bold);

    italic = new JCheckBox("Italic");
    italic.addActionListener(listener);
    buttonPanel.add(italic);

    add(buttonPanel, BorderLayout.SOUTH);
  }

  public static final int DEFAULT_WIDTH = 300;
  public static final int DEFAULT_HEIGHT = 200;

  private JLabel label;
  private JCheckBox bold;
  private JCheckBox italic;

  private static final int FONTSIZE = 12;
}
```

Radio Buttons



The quick brown fox jumps over the lazy dog.

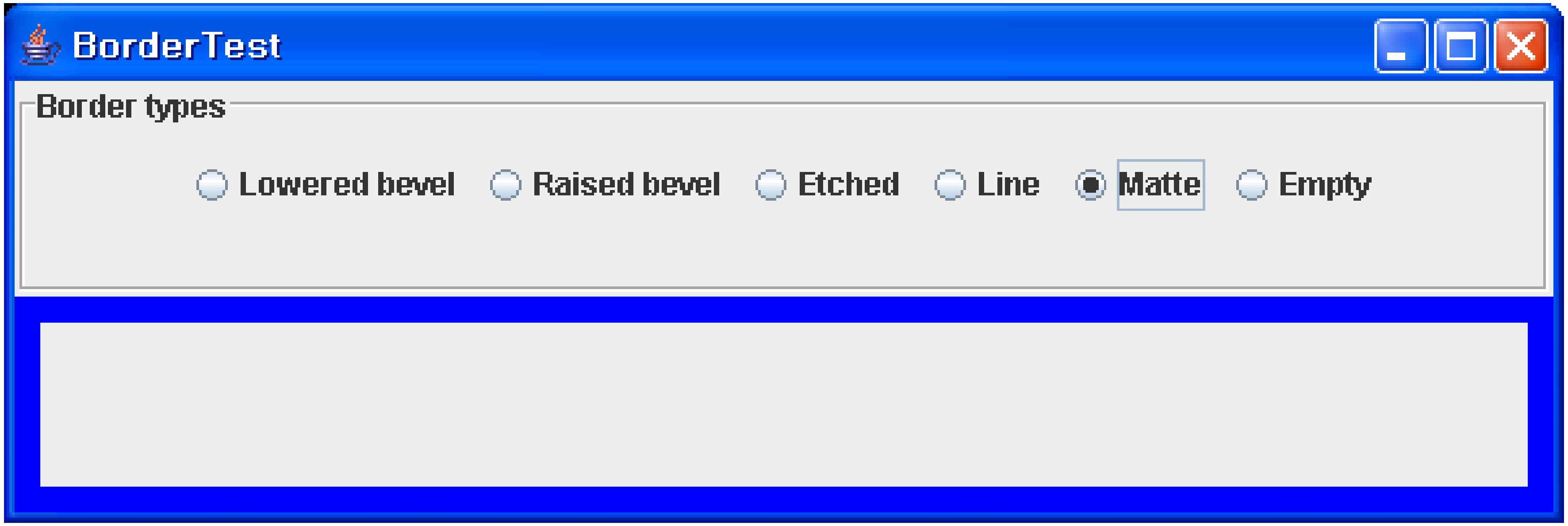
○ Small ○ Medium [✓ Large] ○ Extra large

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
 public class RadioButtonTest {
   public static void main(String[] args) {
      RadioButtonFrame frame = new RadioButtonFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class RadioButtonFrame extends JFrame  {
   public RadioButtonFrame() {
      setTitle("RadioButtonTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      label = new JLabel("The quick brown fox jumps over the lazy dog.");
      label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
      add(label, BorderLayout.CENTER);
      buttonPanel = new JPanel();
      group = new ButtonGroup();
      addRadioButton("Small", 8);
      addRadioButton("Medium", 12);
      addRadioButton("Large", 18);
      addRadioButton("Extra large", 36);
      add(buttonPanel, BorderLayout.SOUTH);
   }
```

```
  public void addRadioButton(String name, final int size) {
     boolean selected = (size == DEFAULT_SIZE);
     JRadioButton button = new JRadioButton(name, selected);
     group.add(button);
     buttonPanel.add(button);

    ActionListener listener = new  ActionListener() {
         public void actionPerformed(ActionEvent event) {
            // size refers to the final parameter of the addRadioButton method
            label.setFont(new Font("Serif", Font.PLAIN, size));
         }
      };
    button.addActionListener(listener);

  }
  public static final int DEFAULT_WIDTH = 400, DEFAULT_HEIGHT = 200;
  private JPanel buttonPanel;
  private ButtonGroup group;
  private JLabel label;
  private static final int DEFAULT_SIZE = 12;
}
```

```
button.addActionListener(
  (ActionEvent e) ->
  label.setFont(new Font("Serif", Font.PLAIN, size))
);
```

Borders


BorderTest

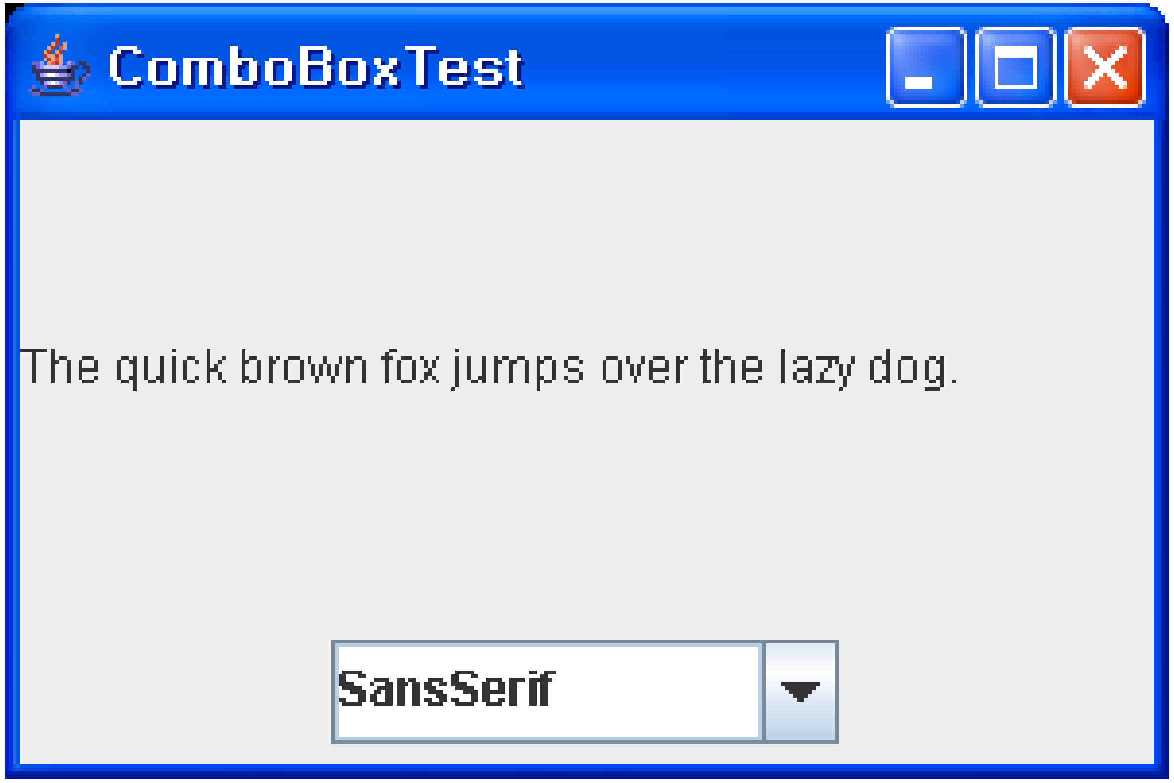
Border types

( ) Lowered bevel ( ) Raised bevel ( ) Etched ( ) Line [✓ Matte] ( ) Empty

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
public class BorderTest {
  public static void main(String[] args) {
    BorderFrame frame = new BorderFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
class BorderFrame extends JFrame  {
  public BorderFrame() {
    setTitle("BorderTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    demoPanel = new JPanel();
    buttonPanel = new JPanel();
    group = new ButtonGroup();
    addRadioButton("Lowered bevel", BorderFactory.createLoweredBevelBorder());
    addRadioButton("Raised bevel", BorderFactory.createRaisedBevelBorder());
    addRadioButton("Etched", BorderFactory.createEtchedBorder());
    addRadioButton("Line", BorderFactory.createLineBorder(Color.BLUE));
    addRadioButton("Matte",
       BorderFactory.createMatteBorder(10, 10, 10, 10, Color.BLUE));
    addRadioButton("Empty", BorderFactory.createEmptyBorder());
    Border etched = BorderFactory.createEtchedBorder();
    Border titled = BorderFactory.createTitledBorder(etched, "Border types");
    buttonPanel.setBorder(titled);

    setLayout(new GridLayout(2, 1));
    add(buttonPanel);
    add(demoPanel);
  }
  public void addRadioButton(String buttonName, final Border b) {
    JRadioButton button = new JRadioButton(buttonName);
    button.addActionListener(new ActionListener() { // anonymous inner class
        public void actionPerformed(ActionEvent event) {
          demoPanel.setBorder(b);
        }
      });

    group.add(button);
    buttonPanel.add(button);
  }
  public static final int DEFAULT_WIDTH = 600, DEFAULT_HEIGHT = 200;
  private JPanel demoPanel;
  private JPanel buttonPanel;
  private ButtonGroup group;
}
```

```
button.addActionListener(
  (ActionEvent e) -> demoPanel.setBorder(b)
);
```

Combo Boxes



The quick brown fox jumps over the lazy dog.

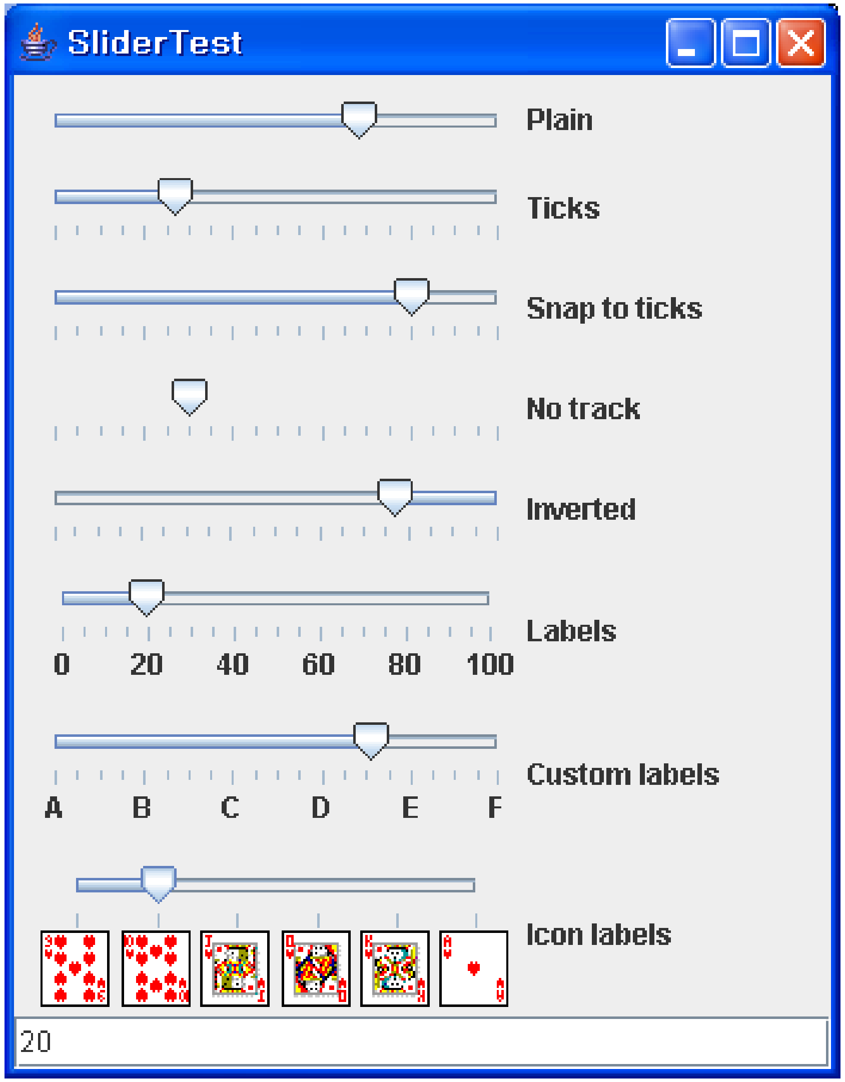
SansSerif

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ComboBoxTest {
  public static void main(String[] args) {
    ComboBoxFrame frame = new ComboBoxFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
class ComboBoxFrame extends JFrame  {
  public ComboBoxFrame() {
    setTitle("ComboBoxTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);l

    label = new JLabel("The quick brown fox jumps over the lazy dog.");
    label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
    add(label, BorderLayout.CENTER);
    faceCombo = new JComboBox<>();
    faceCombo.setEditable(true);
    faceCombo.addItem("Serif");
    faceCombo.addItem("SansSerif");
    faceCombo.addItem("Monospaced");
    faceCombo.addItem("Dialog");
    faceCombo.addItem("DialogInput");
faceCombo.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent event) {
      label.setFont(new Font(
        (String) faceCombo.getSelectedItem(),
        Font.PLAIN, DEFAULT_SIZE));
    }
  });
```

```
faceCombo.addActionListener(
  (ActionEvent event) ->
    label.setFont(new Font(
      (String) faceCombo.getSelectedItem(),
      Font.PLAIN,   DEFAULT_SIZE))
);
```

```
    JPanel comboPanel = new JPanel();
    comboPanel.add(faceCombo);
    add(comboPanel, BorderLayout.SOUTH);
  }

  public static final int DEFAULT_WIDTH = 300;
  public static final int DEFAULT_HEIGHT = 200;

  private JComboBox<String> faceCombo;
  private JLabel label;
  private static final int DEFAULT_SIZE = 12;
}
```

Sliders


SliderTest
Plain
Ticks
Snap to ticks
No track
Inverted
Labels
0
20
40
60
80
100
Custom labels
A
B
C
D
E
F
Icon labels
20

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.event.*;
public class SliderTest {
   public static void main(String[] args) {
      SliderTestFrame frame = new SliderTestFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class SliderTestFrame extends JFrame {
   public SliderTestFrame() {
      setTitle("SliderTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
      sliderPanel = new JPanel();
      sliderPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
      listener = new  ChangeListener() {
            public void stateChanged(ChangeEvent event) {
               // update text field when the slider value changes
               JSlider source = (JSlider) event.getSource();
               textField.setText("" + source.getValue());
            }
         };
```

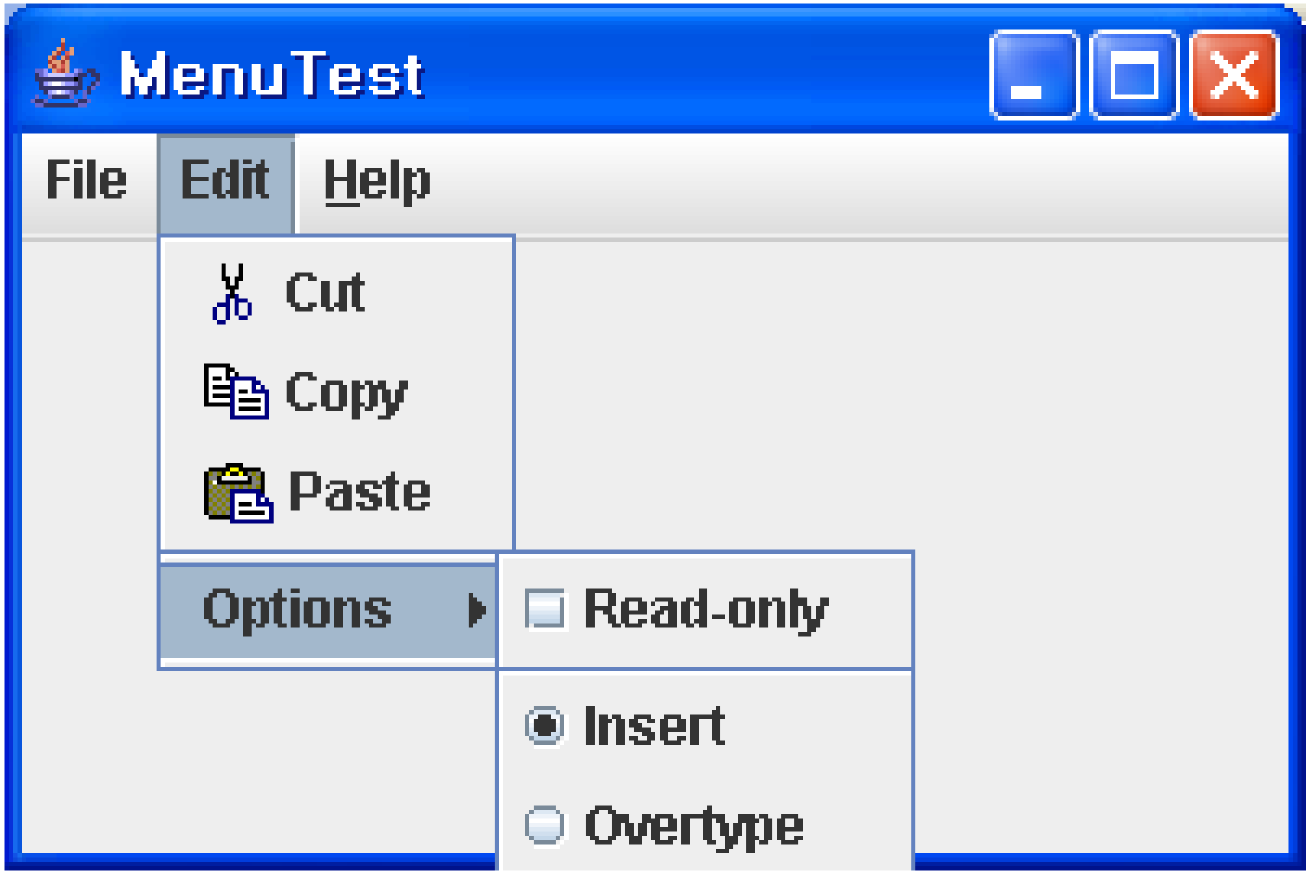
```
// add a plain slider
JSlider slider = new JSlider();
addSlider(slider, "Plain");

// add a slider with major and minor ticks
slider = new JSlider();
slider.setPaintTicks(true);
slider.setMajorTickSpacing(20);
slider.setMinorTickSpacing(5);
addSlider(slider, "Ticks");

// add a slider that snaps to ticks
slider = new JSlider();
slider.setPaintTicks(true);
slider.setSnapToTicks(true);
slider.setMajorTickSpacing(20);
slider.setMinorTickSpacing(5);
addSlider(slider, "Snap to ticks");

// add a slider with no track
slider = new JSlider();
slider.setPaintTicks(true);
slider.setMajorTickSpacing(20);
slider.setMinorTickSpacing(5);
slider.setPaintTrack(false);
addSlider(slider, "No track");
```

```
// add an inverted slider
slider = new JSlider();
slider.setPaintTicks(true);
slider.setMajorTickSpacing(20);
slider.setMinorTickSpacing(5);
slider.setInverted(true);
addSlider(slider, "Inverted");

// add a slider with numeric labels
slider = new JSlider();
slider.setPaintTicks(true);
slider.setPaintLabels(true);
slider.setMajorTickSpacing(20);
slider.setMinorTickSpacing(5);
addSlider(slider, "Labels");

// add a slider with alphabetic labels
slider = new JSlider();
slider.setPaintLabels(true);
slider.setPaintTicks(true);
slider.setMajorTickSpacing(20);
slider.setMinorTickSpacing(5);
```

```
Dictionary<Integer, Component> labelTable =
    new Hashtable<Integer, Component>();
labelTable.put(0, new JLabel("A"));
labelTable.put(20, new JLabel("B"));
labelTable.put(40, new JLabel("C"));
labelTable.put(60, new JLabel("D"));
labelTable.put(80, new JLabel("E"));
labelTable.put(100, new JLabel("F"));
slider.setLabelTable(labelTable); addSlider(slider, "Custom labels");

// add a slider with icon labels
slider = new JSlider();
slider.setPaintTicks(true); slider.setPaintLabels(true);
slider.setSnapToTicks(true);
slider.setMajorTickSpacing(20); slider.setMinorTickSpacing(20);
labelTable = new Hashtable<Integer, Component>();

// add card images
labelTable.put(0, new JLabel(new ImageIcon("nine.gif")));
labelTable.put(20, new JLabel(new ImageIcon("ten.gif")));
labelTable.put(40, new JLabel(new ImageIcon("jack.gif")));
labelTable.put(60, new JLabel(new ImageIcon("queen.gif")));
labelTable.put(80, new JLabel(new ImageIcon("king.gif")));
labelTable.put(100, new JLabel(new ImageIcon("ace.gif")));
slider.setLabelTable(labelTable);
addSlider(slider, "Icon labels");
```

```
    // add the text field that displays the slider value
    textField = new JTextField();
    add(sliderPanel, BorderLayout.CENTER);
    add(textField, BorderLayout.SOUTH);
  }
  public void addSlider(JSlider s, String description) {
    s.addChangeListener(listener);
    JPanel panel = new JPanel();
    panel.add(s);
    panel.add(new JLabel(description));
    sliderPanel.add(panel);
  }

  public static final int DEFAULT_WIDTH = 350;
  public static final int DEFAULT_HEIGHT = 450;

  private JPanel sliderPanel;
  private JTextField textField;
  private ChangeListener listener;
}
```

Menus


MenuTest
File
Edit
Help
Cut
Copy
Paste
Options
Read-only
Insert
Overtype

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
public class MenuTest {
   public static void main(String[] args) {
      MenuFrame frame = new MenuFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

class MenuFrame extends JFrame {
   public MenuFrame() {
      setTitle("MenuTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

      JMenu fileMenu = new JMenu("File");
      JMenuItem newItem = fileMenu.add(new TestAction("New"));

      JMenuItem openItem = fileMenu.add(new TestAction("Open"));
      openItem.setAccelerator(
          KeyStroke.getKeyStroke(KeyEvent.VK_O, InputEvent.CTRL_MASK));

      fileMenu.addSeparator();
```

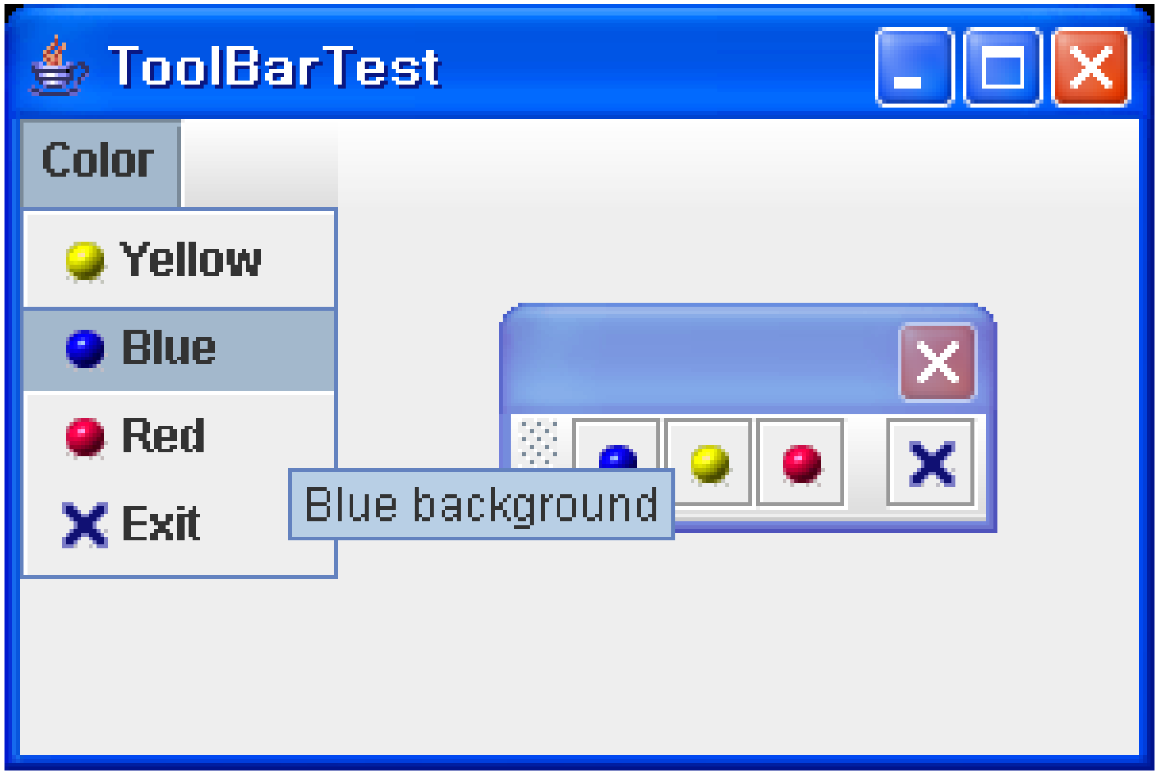
```
saveAction = new TestAction("Save");
JMenuItem saveItem = fileMenu.add(saveAction);
saveItem.setAccelerator(
    KeyStroke.getKeyStroke(KeyEvent.VK_S, InputEvent.CTRL_MASK));

saveAsAction = new TestAction("Save As");
JMenuItem saveAsItem = fileMenu.add(saveAsAction);
fileMenu.addSeparator();

fileMenu.add(new  AbstractAction("Exit") {
      public void actionPerformed(ActionEvent event) { System.exit(0); }
   });

// demonstrate check box and radio button menus
readonlyItem = new JCheckBoxMenuItem("Read-only");
readonlyItem.addActionListener(new
   ActionListener() {
      public void actionPerformed(ActionEvent event) {
         boolean saveOk = !readonlyItem.isSelected();
         saveAction.setEnabled(saveOk);
         saveAsAction.setEnabled(saveOk);
      }
   });
```

```
ButtonGroup group = new ButtonGroup();
JRadioButtonMenuItem insertItem = new JRadioButtonMenuItem("Insert");
insertItem.setSelected(true);
JRadioButtonMenuItem overtypeItem = new JRadioButtonMenuItem("Overtype");

group.add(insertItem);
group.add(overtypeItem);

// demonstrate icons
Action cutAction = new TestAction("Cut");
cutAction.putValue(Action.SMALL_ICON, new ImageIcon("cut.gif"));
Action copyAction = new TestAction("Copy");
copyAction.putValue(Action.SMALL_ICON, new ImageIcon("copy.gif"));
Action pasteAction = new TestAction("Paste");
pasteAction.putValue(Action.SMALL_ICON, new ImageIcon("paste.gif"));

JMenu editMenu = new JMenu("Edit");
editMenu.add(cutAction);
editMenu.add(copyAction);
editMenu.add(pasteAction);
```

```
// demonstrate nested menus
JMenu optionMenu = new JMenu("Options");
optionMenu.add(readonlyItem);
optionMenu.addSeparator();
optionMenu.add(insertItem);
optionMenu.add(overtypeItem);

editMenu.addSeparator();
editMenu.add(optionMenu);

// demonstrate mnemonics
JMenu helpMenu = new JMenu("Help");
helpMenu.setMnemonic('H');
JMenuItem indexItem = new JMenuItem("Index");
indexItem.setMnemonic('I');
helpMenu.add(indexItem);

// you can also add the mnemonic key to an action
Action aboutAction = new TestAction("About");
aboutAction.putValue(Action.MNEMONIC_KEY, new Integer('A'));
helpMenu.add(aboutAction);

// add all top-level menus to menu bar
JMenuBar menuBar = new JMenuBar();
setJMenuBar(menuBar);
menuBar.add(fileMenu); menuBar.add(editMenu); menuBar.add(helpMenu);
```

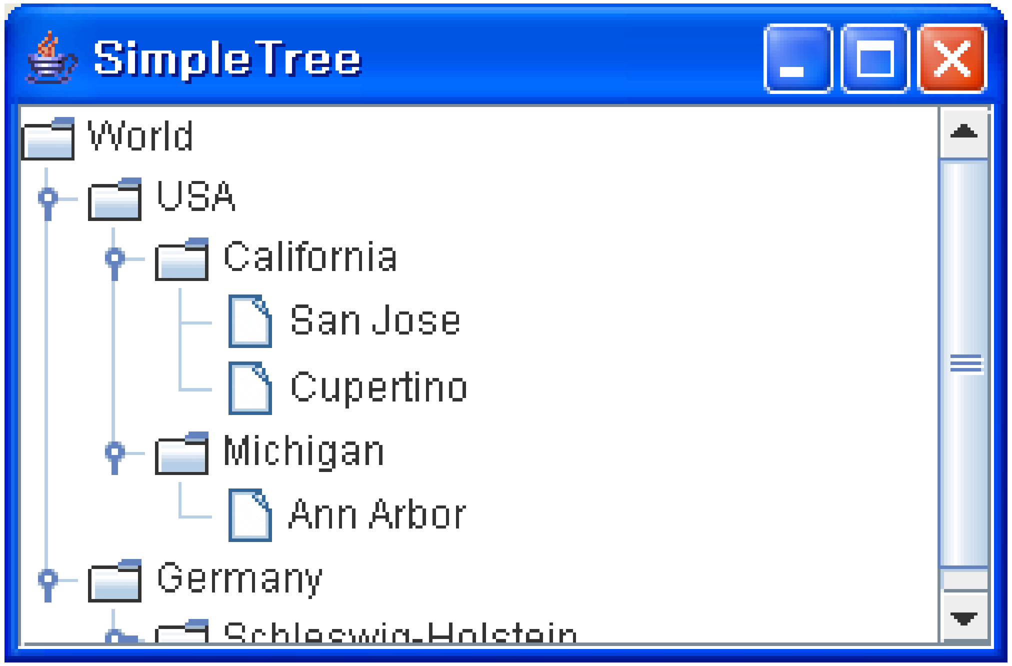
```
    // demonstrate pop-ups
    popup = new JPopupMenu();
    popup.add(cutAction); popup.add(copyAction); popup.add(pasteAction);

    JPanel panel = new JPanel();
    panel.setComponentPopupMenu(popup);
    add(panel);
  }
  public static final int DEFAULT_WIDTH = 300;
  public static final int DEFAULT_HEIGHT = 200;

  private Action saveAction;
  private Action saveAsAction;
  private JCheckBoxMenuItem readonlyItem;
  private JPopupMenu popup;
}

class TestAction extends AbstractAction {
  public TestAction(String name) { super(name); }

  public void actionPerformed(ActionEvent event) {
    System.out.println(getValue(Action.NAME) + " selected.");
  }
}
```

Toolbars



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

public class ToolBarTest {
  public static void main(String[] args) {
    ToolBarFrame frame = new ToolBarFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
class ToolBarFrame extends JFrame {
  public ToolBarFrame() {
    setTitle("ToolBarTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

    // add a panel for color change
    panel = new JPanel(); add(panel, BorderLayout.CENTER);

    // set up actions
    Action blueAction = new ColorAction("Blue",
       new ImageIcon("blue-ball.gif"), Color.BLUE);
    Action yellowAction = new ColorAction("Yellow",
       new ImageIcon("yellow-ball.gif"), Color.YELLOW);
    Action redAction = new ColorAction("Red",
       new ImageIcon("red-ball.gif"), Color.RED);
    Action exitAction = new
      AbstractAction("Exit", new ImageIcon("exit.gif")) {
        public void actionPerformed(ActionEvent event) { System.exit(0); }
      };
    exitAction.putValue(Action.SHORT_DESCRIPTION, "Exit");

    // populate tool bar
    JToolBar bar = new JToolBar();
    bar.add(blueAction);
    bar.add(yellowAction);
    bar.add(redAction);
    bar.addSeparator();
    bar.add(exitAction);
    add(bar, BorderLayout.NORTH);

    // populate menu
    JMenu menu = new JMenu("Color");
    menu.add(yellowAction);
    menu.add(blueAction);
    menu.add(redAction);
    menu.add(exitAction);
    JMenuBar menuBar = new JMenuBar();
    menuBar.add(menu);
    setJMenuBar(menuBar);
  }
  public static final int DEFAULT_WIDTH = 300;
  public static final int DEFAULT_HEIGHT = 200;

  private JPanel panel;

  /**
     The color action sets the background of the frame to a
     given color.
  */
  class ColorAction extends AbstractAction {
     public ColorAction(String name, Icon icon, Color c) {
        putValue(Action.NAME, name);
        putValue(Action.SMALL_ICON, icon);
        putValue(Action.SHORT_DESCRIPTION, name + " background");
        putValue("Color", c);
     }

     public void actionPerformed(ActionEvent event) {
        Color c = (Color) getValue("Color");
        panel.setBackground(c);
     }
  }
}
```

Trees

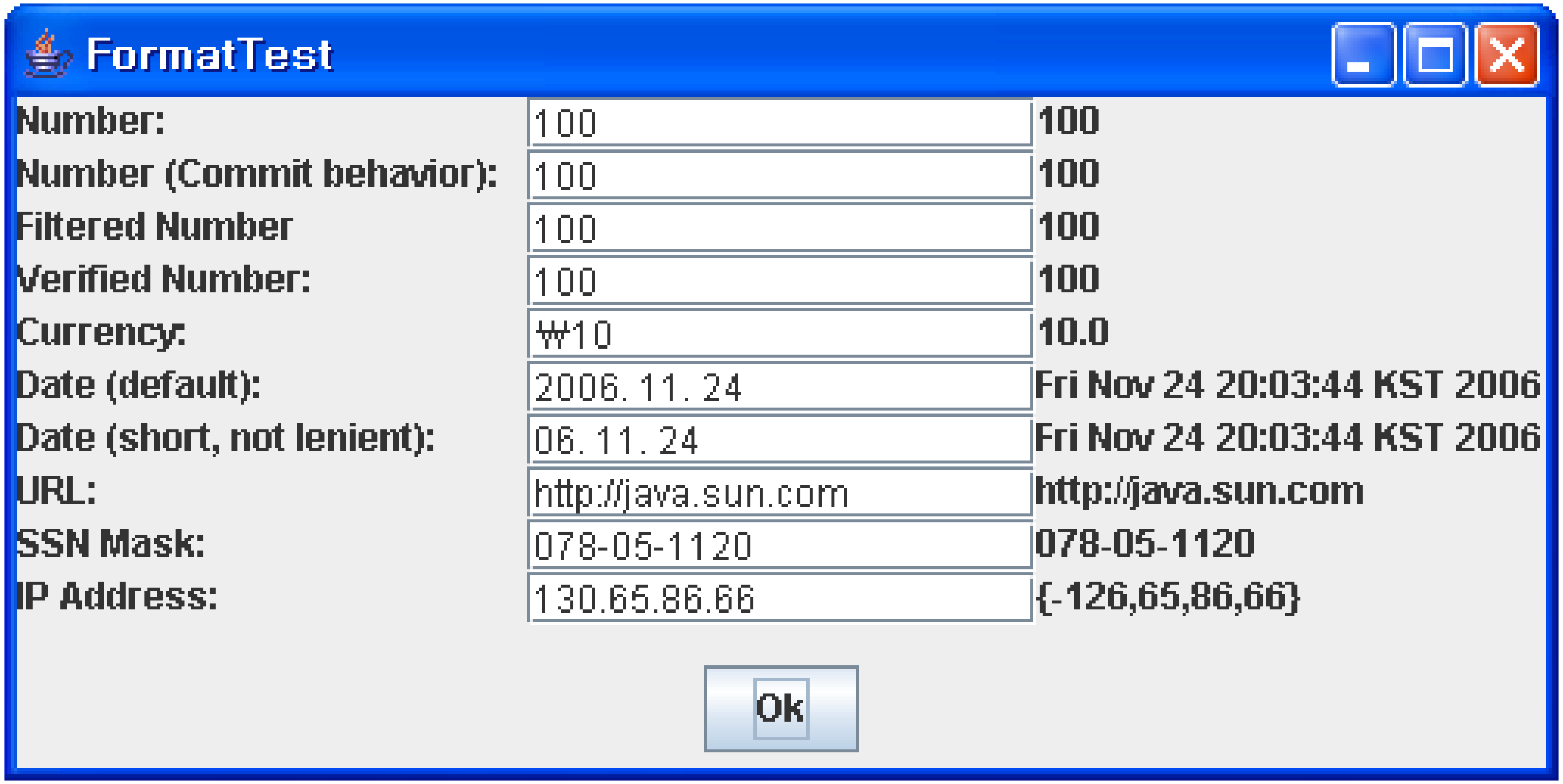


How to Use Trees in Java Tutorial
https://docs.oracle.com/javase/tutorial/uiswing/components/tree.html

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

public class SimpleTree {
   public static void main(String[] args) {
      JFrame frame = new SimpleTreeFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}

/**
   This frame contains a simple tree that displays a
   manually constructed tree model.
*/
class SimpleTreeFrame extends JFrame
{
   public SimpleTreeFrame()
   {
      setTitle("SimpleTree");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
```

```
    // set up tree model data
    DefaultMutableTreeNode root = new DefaultMutableTreeNode("World");
    DefaultMutableTreeNode country = new DefaultMutableTreeNode("USA");
    root.add(country);

    DefaultMutableTreeNode state = new DefaultMutableTreeNode("California");
    country.add(state);
    DefaultMutableTreeNode city = new DefaultMutableTreeNode("San Jose");
    state.add(city);
    city = new DefaultMutableTreeNode("Cupertino");
    state.add(city);

    state = new DefaultMutableTreeNode("Michigan");
    country.add(state);
    city = new DefaultMutableTreeNode("Ann Arbor");
    state.add(city);
    country = new DefaultMutableTreeNode("Germany");
    root.add(country);
    state = new DefaultMutableTreeNode("Schleswig-Holstein");
    country.add(state);
    city = new DefaultMutableTreeNode("Kiel");
    state.add(city);
    // construct tree and put it in a scroll pane
    JTree tree = new JTree(root);
    Container contentPane = getContentPane();
    contentPane.add(new JScrollPane(tree));
  }
  private static final int DEFAULT_WIDTH = 300, DEFAULT_HEIGHT = 200;
}
```

Formatted Input


FormatTest

Number:	100	100
Number (Commit behavior):	100	100
Filtered Number	100	100
Verified Number:	100	100
Currency:	₩10	10.0
Date (default):	2006. 11. 24	Fri Nov 24 20:03:44 KST 2006
Date (short, not lenient):	06. 11. 24	Fri Nov 24 20:03:44 KST 2006
URL:	http://java.sun.com	http://java.sun.com
SSN Mask:	078-05-1120	078-05-1120
IP Address:	130.65.86.66	{-126,65,86,66}

How to Use Formatted Text Fields
https://docs.oracle.com/javase/tutorial/uiswing/components/formattedtextfield.html

```java
import java.awt.*;
import java.awt.event.*;
import java.lang.reflect.*;
import java.net.*;
import java.text.*;
import java.util.*;
import javax.swing.*;
import javax.swing.text.*;

/**
   A program to test formatted text fields
*/
public class FormatTest
{
   public static void main(String[] args)
   {
      FormatTestFrame frame = new FormatTestFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

```
class FormatTestFrame extends JFrame {
  public FormatTestFrame() {
    setTitle("FormatTest");
    setSize(WIDTH, HEIGHT);

    JPanel buttonPanel = new JPanel();
    okButton = new JButton("Ok");
    buttonPanel.add(okButton);
    add(buttonPanel, BorderLayout.SOUTH);

    mainPanel = new JPanel();
    mainPanel.setLayout(new GridLayout(0, 3));
    add(mainPanel, BorderLayout.CENTER);

    JFormattedTextField intField =
       new JFormattedTextField(NumberFormat.getIntegerInstance());
    // java.text.NumberFormat
    // JFormattedTextField(Format format): Creates a JFormattedTextField.
    intField.setValue(new Integer(100));
    addRow("Number:", intField);

    JFormattedTextField intField2 =
       new JFormattedTextField(NumberFormat.getIntegerInstance());
    intField2.setValue(new Integer(100));
    intField2.setFocusLostBehavior(JFormattedTextField.COMMIT);
    addRow("Number (Commit behavior):", intField2);
```

```
JFormattedTextField intField3 = new JFormattedTextField(new
     InternationalFormatter (NumberFormat.getIntegerInstance()) {
    // javax.swing.text.InternationalFormatter for formatting string
        protected DocumentFilter getDocumentFilter() { return filter; }
        // javax.swing.text.DefaultFormatter.getDocumentFilter()
        private DocumentFilter filter = new IntFilter();
     });
intField3.setValue(new Integer(100));  addRow("Filtered Number", intField3);

JFormattedTextField intField4 =
    new JFormattedTextField(NumberFormat.getIntegerInstance());
intField4.setValue(new Integer(100));
intField4.setInputVerifier(new FormattedTextFieldVerifier());
addRow("Verified Number:", intField4);

JFormattedTextField currencyField
   = new JFormattedTextField(NumberFormat.getCurrencyInstance());
currencyField.setValue(new Double(10));
addRow("Currency:", currencyField);

JFormattedTextField dateField =
    new JFormattedTextField(DateFormat.getDateInstance());
dateField.setValue(new Date());
addRow("Date (default):", dateField);
```

```
DateFormat format = DateFormat.getDateInstance(DateFormat.SHORT);
format.setLenient(false);
JFormattedTextField dateField2 = new JFormattedTextField(format);
dateField2.setValue(new Date());
addRow("Date (short, not lenient):", dateField2);

try {
  DefaultFormatter formatter = new DefaultFormatter();
  formatter.setOverwriteMode(false);
  JFormattedTextField urlField = new JFormattedTextField(formatter);
  urlField.setValue(new URL("http://java.sun.com"));
  addRow("URL:", urlField);
}
catch (MalformedURLException e) { e.printStackTrace(); }

try {
  MaskFormatter formatter = new MaskFormatter("###-##-####");
  formatter.setPlaceholderCharacter('0');
  JFormattedTextField ssnField = new JFormattedTextField(formatter);
  ssnField.setValue("078-05-1120");
  addRow("SSN Mask:", ssnField);
}
catch (ParseException exception) { exception.printStackTrace(); }
```

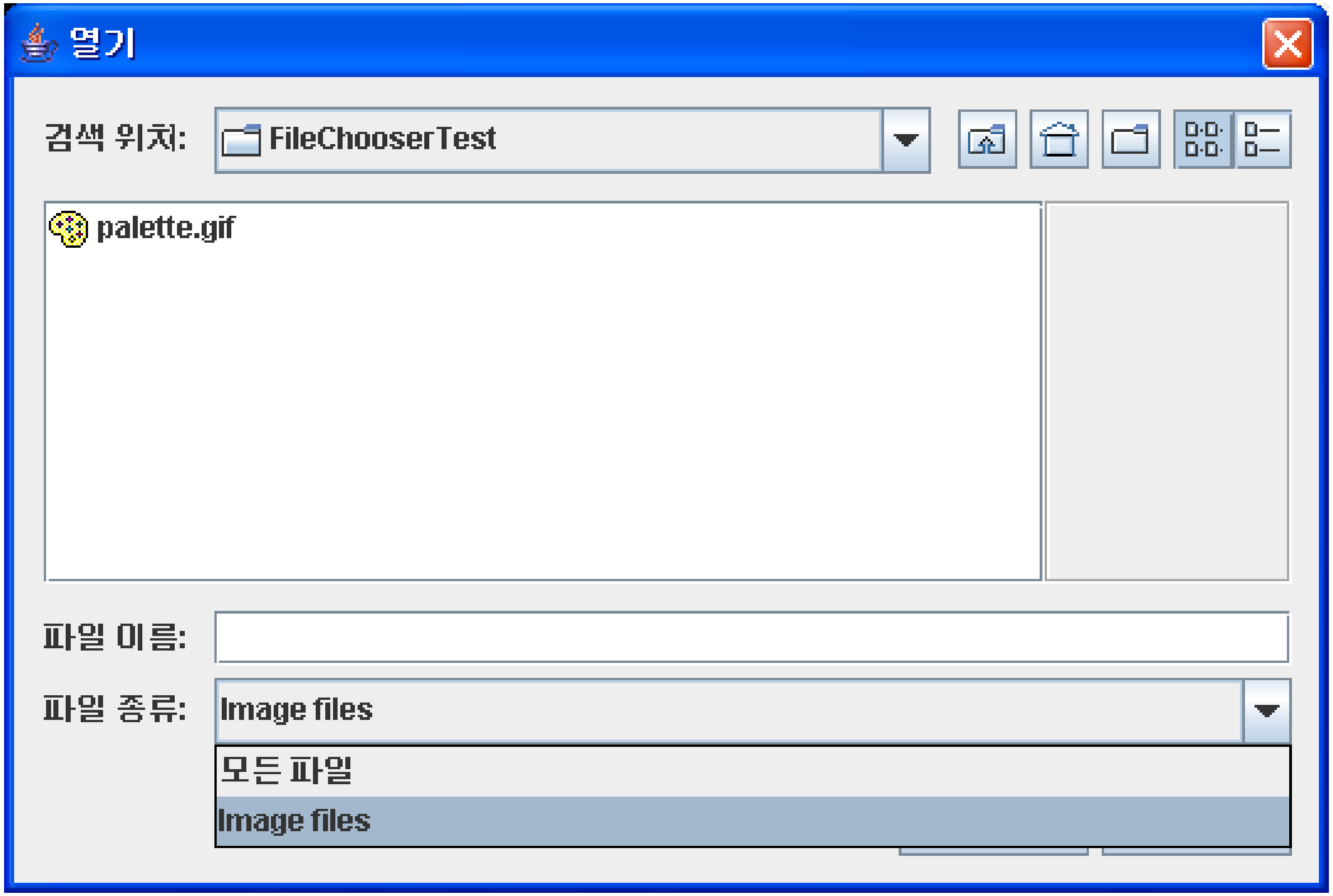
```
JFormattedTextField ipField = new JFormattedTextField(new IPAddressFormatter());
   ipField.setValue(new byte[] { (byte) 130, 65, 86, 66 });
   addRow("IP Address:", ipField);
}
public void addRow(String labelText, final JFormattedTextField field) {
   mainPanel.add(new JLabel(labelText)); mainPanel.add(field);
   final JLabel valueLabel = new JLabel();
   mainPanel.add(valueLabel);
   okButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) {
           Object value = field.getValue();
           if (value.getClass().isArray()) {
              StringBuilder builder = new StringBuilder(); builder.append('{');
              for (int i = 0; i < Array.getLength(value); i++) {
                 if (i > 0) builder.append(',');
                 builder.append(Array.get(value, i).toString());
              }
              builder.append('}');
              valueLabel.setText(builder.toString());
           }
           else valueLabel.setText(value.toString());
        }
     });
}
```

```
   public static final int WIDTH = 500, HEIGHT = 250;
   private JButton okButton;
   private JPanel mainPanel;
}
class IntFilter extends DocumentFilter {
  // javax.swing.text.DocumentFilter
  // insertString: Invoked prior to insertion of text into the specified Document.
   public void insertString (FilterBypass fb, int offset, String string, AttributeSet attr)
      throws BadLocationException  {
      // analyze string to be inserted and inserts only the chars that are digits or a - sign
      StringBuilder builder = new StringBuilder(string);
      for (int i = builder.length() - 1; i >= 0; i--) {
         int cp = builder.codePointAt(i);
         if (!Character.isDigit(cp) && cp != '-')  {
            builder.deleteCharAt(i);
            if (Character.isSupplementaryCodePoint(cp)) {
               i--;
               builder.deleteCharAt(i);
            }
         }
      }
      super.insertString(fb, offset, builder.toString(), attr);
   }
```

```java
public void replace (FilterBypass fb, int offset, int length, String string, AttributeSet attr)
    throws BadLocationException  {
    if (string != null)  {
      StringBuilder builder = new StringBuilder(string);
      for (int i = builder.length() - 1; i >= 0; i--) {
        int cp = builder.codePointAt(i);
        if (!Character.isDigit(cp) && cp != '-')  {
          builder.deleteCharAt(i);
          if (Character.isSupplementaryCodePoint(cp)) {
            i--;
            builder.deleteCharAt(i);
          }
        }
      }
      string = builder.toString();
    }
    super.replace(fb, offset, length, string, attr);
  }
}
```

```
class FormattedTextFieldVerifier extends InputVerifier  {
  public boolean verify(JComponent component)  {
    JFormattedTextField field = (JFormattedTextField) component;
    return field.isEditValid();
  }
}
class IPAddressFormatter extends DefaultFormatter {
  public String valueToString(Object value)  throws ParseException {
    if (!(value instanceof byte[]))
      throw new ParseException("Not a byte[]", 0);
    byte[] a = (byte[]) value;
    if (a.length != 4)
      throw new ParseException("Length != 4", 0);
    StringBuilder builder = new StringBuilder();
    for (int i = 0; i < 4; i++) {
      int b = a[i];
      if (b < 0) b += 256;
      builder.append(String.valueOf(b));
      if (i < 3) builder.append('.');
    }
    return builder.toString();
  }
  public Object stringToValue(String text) throws ParseException {
    StringTokenizer tokenizer = new StringTokenizer(text, ".");
    byte[] a = new byte[4];
    for (int i = 0; i < 4; i++) {
      int b = 0;
      if (!tokenizer.hasMoreTokens())
        throw new ParseException("Too few bytes", 0);
      try {
        b = Integer.parseInt(tokenizer.nextToken());
      }
      catch (NumberFormatException e) {
        throw new ParseException("Not an integer", 0);
      }
      if (b < 0 || b >= 256)
        throw new ParseException("Byte out of range", 0);
      a[i] = (byte) b;
    }
    if (tokenizer.hasMoreTokens())
      throw new ParseException("Too many bytes", 0);
    return a;
  }
}
```

File Dialogs



```
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import java.beans.*;
import java.util.*;
import java.io.*;
import javax.swing.*;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileView;

public class FileChooserTest {
   public static void main(String[] args) {
      ImageViewerFrame frame = new ImageViewerFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

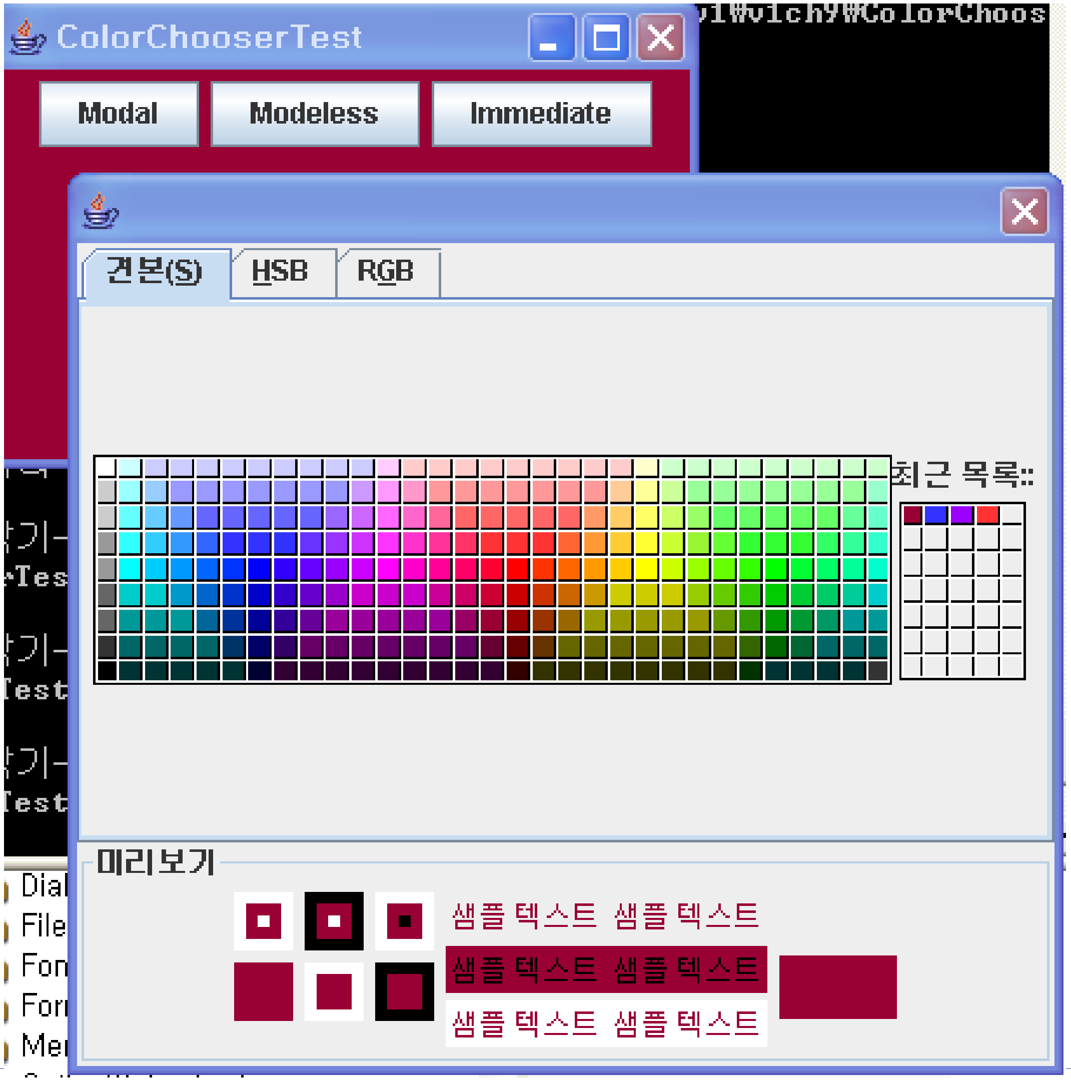
```
class ImageViewerFrame extends JFrame {
  public ImageViewerFrame() {
    setTitle("FileChooserTest");
    setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

    // set up menu bar
    JMenuBar menuBar = new JMenuBar();
    setJMenuBar(menuBar);

    JMenu menu = new JMenu("File");
    menuBar.add(menu);

    JMenuItem openItem = new JMenuItem("Open");
    menu.add(openItem);
    openItem.addActionListener(new FileOpenListener());

    JMenuItem exitItem = new JMenuItem("Exit");
    menu.add(exitItem);
    exitItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent event) { System.exit(0); }
      });

    // use a label to display the images
    label = new JLabel();
    add(label);
    // set up file chooser
    chooser = new JFileChooser();

    // accept all image files ending with .jpg, .jpeg, .gif
    final ExtensionFileFilter filter = new ExtensionFileFilter();
    filter.addExtension("jpg"); filter.addExtension("jpeg"); filter.addExtension("gif");
    filter.setDescription("Image files");
    chooser.setFileFilter(filter);

    chooser.setAccessory(new ImagePreviewer(chooser));
    chooser.setFileView(new FileIconView(filter, new ImageIcon("palette.gif")));
  }
  private class FileOpenListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
      chooser.setCurrentDirectory(new File("."));
      // show file chooser dialog
      int result = chooser.showOpenDialog(ImageViewerFrame.this);
      // if image file accepted, set it as icon of the label
      if(result == JFileChooser.APPROVE_OPTION) {
        String name = chooser.getSelectedFile().getPath();
        label.setIcon(new ImageIcon(name));
      }
    }
  }
  public static final int DEFAULT_WIDTH = 300, DEFAULT_HEIGHT = 400;
  private JLabel label;
  private JFileChooser chooser;
}
```

```
class ExtensionFileFilter extends FileFilter {
  public void addExtension(String extension) {
    if (!extension.startsWith("."))
        extension = "." + extension;
    extensions.add(extension.toLowerCase());
  }
  public void setDescription(String aDescription) { description = aDescription; }

  public String getDescription() { return description; }

  public boolean accept(File f) {
    if (f.isDirectory()) return true;
    String name = f.getName().toLowerCase();

    // check if the file name ends with any of the extensions
    for (String extension : extensions)
      if (name.endsWith(extension))
        return true;
    return false;
  }

  private String description = "";
  private ArrayList<String> extensions = new ArrayList<String>();
}
```

```
class FileIconView extends FileView {
  /**
     Constructs a FileIconView.
     @param aFilter a file filter--all files that this filter
     accepts will be shown with the icon.
     @param anIcon--the icon shown with all accepted files.
  */
  public FileIconView(FileFilter aFilter, Icon anIcon)
  {
    filter = aFilter;
    icon = anIcon;
  }

  public Icon getIcon(File f)
  {
    if (!f.isDirectory() && filter.accept(f))
      return icon;
    else return null;
  }

  private FileFilter filter;
  private Icon icon;
}
```

```
class ImagePreviewer extends JLabel  {
  public ImagePreviewer(JFileChooser chooser)  {
    setPreferredSize(new Dimension(100, 100));
    setBorder(BorderFactory.createEtchedBorder());

    chooser.addPropertyChangeListener(new  PropertyChangeListener() {
        public void propertyChange(PropertyChangeEvent event)  {
          if (event.getPropertyName() ==
              JFileChooser.SELECTED_FILE_CHANGED_PROPERTY) {
            // the user has selected a new file
            File f = (File) event.getNewValue();
            if (f == null) { setIcon(null); return; }

            // read the image into an icon
            ImageIcon icon = new ImageIcon(f.getPath());

            // if the icon is too large to fit, scale it
            if(icon.getIconWidth() > getWidth())
              icon = new ImageIcon(icon.getImage().getScaledInstance(
                getWidth(), -1, Image.SCALE_DEFAULT));

            setIcon(icon);
          }
        }
    });
  }
}
```

Color Choosers



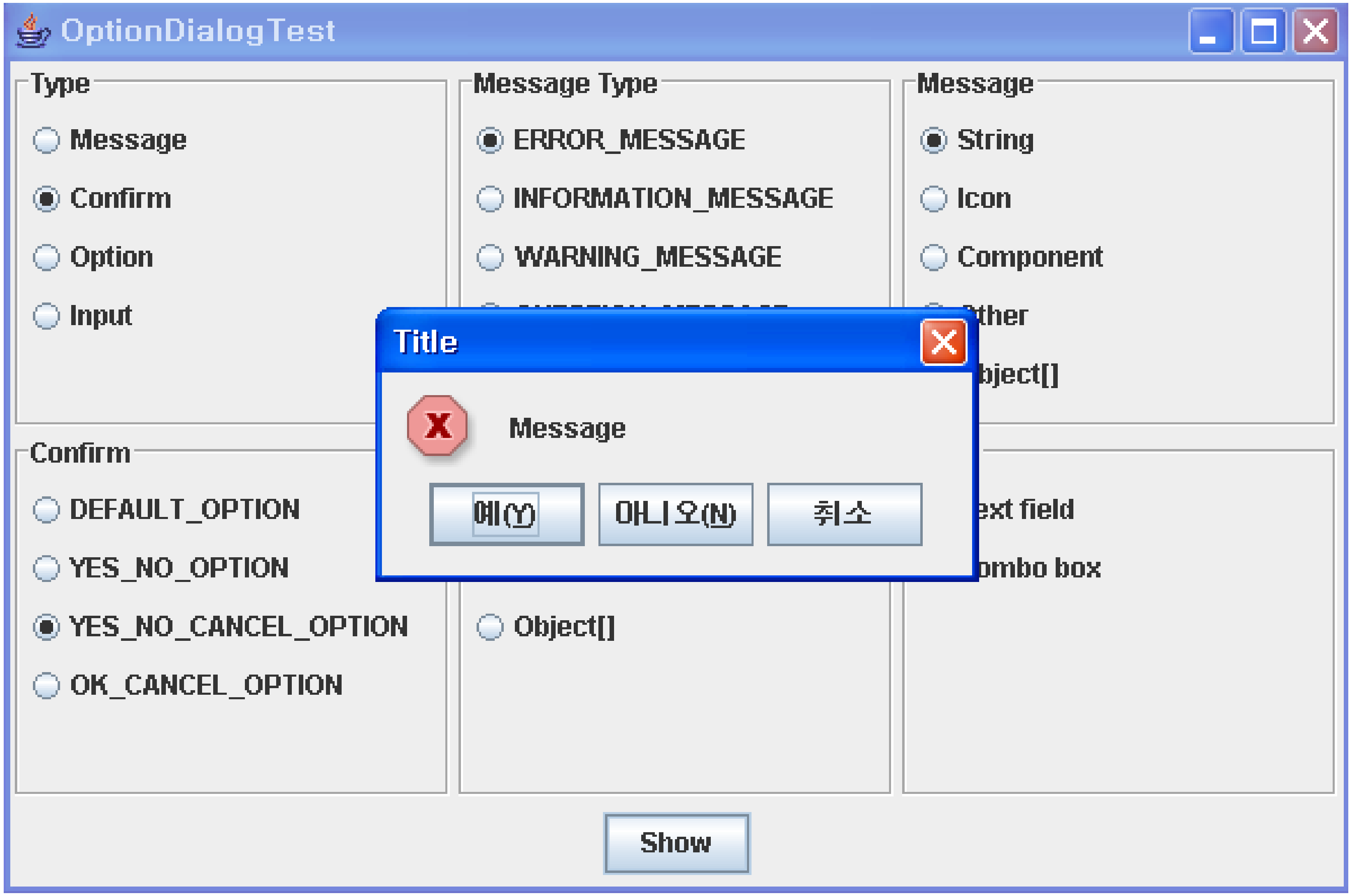
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class ColorChooserTest {
   public static void main(String[] args) {
      ColorChooserFrame frame = new ColorChooserFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
class ColorChooserFrame extends JFrame {
   public ColorChooserFrame() {
      setTitle("ColorChooserTest");
      setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);

      // add color chooser panel to frame

      ColorChooserPanel panel = new ColorChooserPanel();
      add(panel);
   }

   public static final int DEFAULT_WIDTH = 300;
   public static final int DEFAULT_HEIGHT = 200;
}
```

```
/**
  A panel with buttons to pop up three types of color choosers
*/
class ColorChooserPanel extends JPanel {
  public ColorChooserPanel() {
    JButton modalButton = new JButton("Modal");
    modalButton.addActionListener(new ModalListener());
    add(modalButton);

    JButton modelessButton = new JButton("Modeless");
    modelessButton.addActionListener(new ModelessListener());
    add(modelessButton);

    JButton immediateButton = new JButton("Immediate");
    immediateButton.addActionListener(new ImmediateListener());
    add(immediateButton);
  }
  /**
    This listener pops up a modal color chooser
  */
  private class ModalListener implements ActionListener {
    public void actionPerformed(ActionEvent event) {
      Color defaultColor = getBackground();
      Color selected = JColorChooser.showDialog(
        ColorChooserPanel.this,  "Set background",  defaultColor);
      if (selected != null) setBackground(selected);
    }
```

```java
/**
   This listener pops up a modeless color chooser.
   The panel color is changed when the user clicks the Ok
   button.
 */
 private class ModelessListener implements ActionListener {
   public ModelessListener() {
      chooser = new JColorChooser();
      dialog = JColorChooser.createDialog(
         ColorChooserPanel.this, "Background Color", false /* not modal */, chooser,
         new ActionListener() { // OK button listener
               public void actionPerformed(ActionEvent event) {
                  setBackground(chooser.getColor());
               }
            },
         null /* no Cancel button listener */);
   }
   public void actionPerformed(ActionEvent event) {
      chooser.setColor(getBackground());
      dialog.setVisible(true);
   }
   private JDialog dialog;
   private JColorChooser chooser;
 }
```

```
  /**
     This listener pops up a modeless color chooser.
     The panel color is changed immediately when the
     user picks a new color.
  */
  private class ImmediateListener implements ActionListener {
     public ImmediateListener() {
        chooser = new JColorChooser();
        chooser.getSelectionModel().addChangeListener(new ChangeListener() {
           public void stateChanged(ChangeEvent event) {
              // javax.swing.event.ChangeEvent
              setBackground(chooser.getColor());
             }
           });
        dialog = new JDialog( (Frame) null,  false /* not modal */);
        dialog.add(chooser);
        dialog.pack();
     }
     public void actionPerformed(ActionEvent event) {
        chooser.setColor(getBackground());
        dialog.setVisible(true);
     }
     private JDialog dialog;
     private JColorChooser chooser;
  }
}
```

Option Dialog



```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import java.util.*;
import javax.swing.*;
import javax.swing.border.*;
public class OptionDialogTest {
   public static void main(String[] args) {
      OptionDialogFrame frame = new OptionDialogFrame();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); frame.setVisible(true);
   }
}
class ButtonPanel extends Jpanel {
  public ButtonPanel(String title, String[] options) {
      setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder(), title));
      setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
      group = new ButtonGroup();
      // make one radio button for each option
      for (int i = 0; i < options.length; i++) {
         JRadioButton b = new JRadioButton(options[i]);
         b.setActionCommand(options[i]);
         add(b);
         group.add(b);
         b.setSelected(i == 0);
      }
   }
  public String getSelection() { return group.getSelection().getActionCommand(); }
  private ButtonGroup group;
}
class OptionDialogFrame extends JFrame  {
  public OptionDialogFrame() {
    setTitle("OptionDialogTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
    JPanel gridPanel = new JPanel(); gridPanel.setLayout(new GridLayout(2, 3));

    typePanel = new ButtonPanel("Type",
       new String[] { "Message",  "Confirm",  "Option",  "Input" });

    messageTypePanel = new ButtonPanel("Message Type",
      new String[] { "ERROR_MESSAGE",  "INFORMATION_MESSAGE",
        "WARNING_MESSAGE", "QUESTION_MESSAGE", "PLAIN_MESSAGE"
      });

    messagePanel = new ButtonPanel("Message",
      new String[] { "String", "Icon", "Component", "Other", "Object[]‟ });

    optionTypePanel = new ButtonPanel("Confirm",
      new String[]  { "DEFAULT_OPTION",  "YES_NO_OPTION",
        "YES_NO_CANCEL_OPTION",  "OK_CANCEL_OPTION"
      });

    optionsPanel = new ButtonPanel("Option",
       new String[] { "String[]", "Icon[]", "Object[]"});

    inputPanel = new ButtonPanel("Input", new String[] {   "Text field", "Combo box‟ });
    gridPanel.add(typePanel);
    gridPanel.add(messageTypePanel);
    gridPanel.add(messagePanel);
    gridPanel.add(optionTypePanel);
    gridPanel.add(optionsPanel);
    gridPanel.add(inputPanel);

    // add a panel with a Show button
    JPanel showPanel = new JPanel();
    JButton showButton = new JButton("Show");
    showButton.addActionListener(new ShowAction());
    showPanel.add(showButton);

    add(gridPanel, BorderLayout.CENTER);
    add(showPanel, BorderLayout.SOUTH);
  }
  public Object getMessage() {
    String s = messagePanel.getSelection();
    if (s.equals("String")) return messageString;
    else if (s.equals("Icon")) return messageIcon;
    else if (s.equals("Component")) return messageComponent;
    else if (s.equals("Object[]")) return new Object[] {
        messageString, messageIcon, messageComponent, messageObject
      };
    else if (s.equals("Other")) return messageObject;
    else return null;
  }
```

```
public Object[] getOptions() {
   String s = optionsPanel.getSelection();
   if (s.equals("String[]")) return new String[] { "Yellow", "Blue", "Red" };
   else if (s.equals("Icon[]"))
      return new Icon[] {   new ImageIcon("yellow-ball.gif"),
            new ImageIcon("blue-ball.gif"), new ImageIcon("red-ball.gif")
      };
   else if (s.equals("Object[]"))
      return new Object[] {
            messageString, messageIcon, messageComponent, messageObject
      };
   else  return null;
}
public int getType(ButtonPanel panel) {
   String s = panel.getSelection();
   try { return JOptionPane.class.getField(s).getInt(null); }
   catch(Exception e) {   return -1; }
}
```

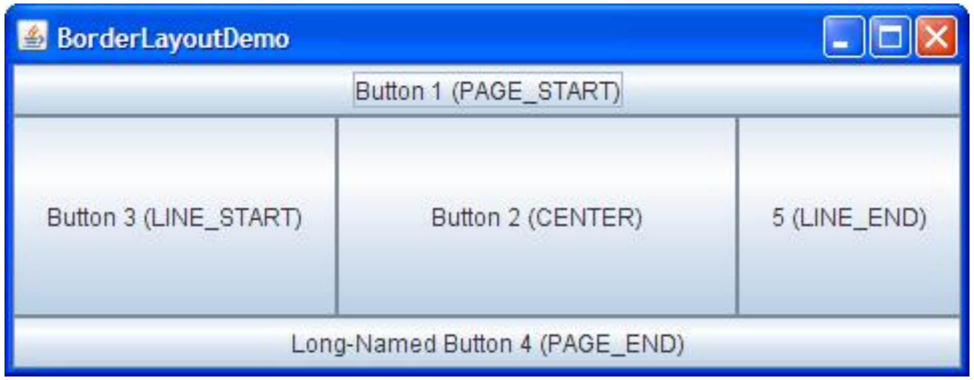
```
private class ShowAction implements ActionListener {
   public void actionPerformed(ActionEvent event) {
      if (typePanel.getSelection().equals("Confirm"))
         JOptionPane.showConfirmDialog(
            OptionDialogFrame.this, getMessage(),
            "Title", getType(optionTypePanel), getType(messageTypePanel));
      else if (typePanel.getSelection().equals("Input"))
      {
         if (inputPanel.getSelection().equals("Text field"))
            JOptionPane.showInputDialog(
               OptionDialogFrame.this, getMessage(),
               "Title", getType(messageTypePanel));
         else
            JOptionPane.showInputDialog(
               OptionDialogFrame.this, getMessage(),
               "Title", getType(messageTypePanel), null,
               new String[] { "Yellow", "Blue", "Red" }, "Blue");
      }
      else if (typePanel.getSelection().equals("Message"))
         JOptionPane.showMessageDialog(
            OptionDialogFrame.this, getMessage(),
            "Title", getType(messageTypePanel));
      else if (typePanel.getSelection().equals("Option"))
         JOptionPane.showOptionDialog(
            OptionDialogFrame.this, getMessage(),
            "Title", getType(optionTypePanel), getType(messageTypePanel),
            null, getOptions(), getOptions()[0]);
   }
   public static final int DEFAULT_WIDTH = 600;
   public static final int DEFAULT_HEIGHT = 400;

   private ButtonPanel typePanel;
   private ButtonPanel messagePanel;
   private ButtonPanel messageTypePanel;
   private ButtonPanel optionTypePanel;
   private ButtonPanel optionsPanel;
   private ButtonPanel inputPanel;

   private String messageString = "Message";
   private Icon messageIcon = new ImageIcon("blue-ball.gif");
   private Object messageObject = new Date();
   private Component messageComponent = new SamplePanel();
 }
class SamplePanel extends JPanel {
   public void paintComponent(Graphics g) {
      super.paintComponent(g);
      Graphics2D g2 = (Graphics2D) g;
      Rectangle2D rect = new Rectangle2D.Double(0, 0, getWidth() - 1, getHeight() - 1);
      g2.setPaint(Color.YELLOW);
      g2.fill(rect);
      g2.setPaint(Color.BLUE);
      g2.draw(rect);
   }
   public Dimension getMinimumSize() { return new Dimension(10, 10);}
}
```

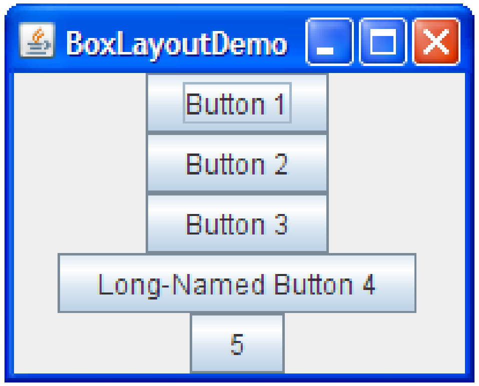
Layout Management

- A layout manager determines the size and position of the components within a container.
  - BorderLayout
  - BoxLayout
  - CardLayout
  - FlowLayout
  - GridBagLayout
  - GridLayout
  - GroupLayout
  - SpringLayout

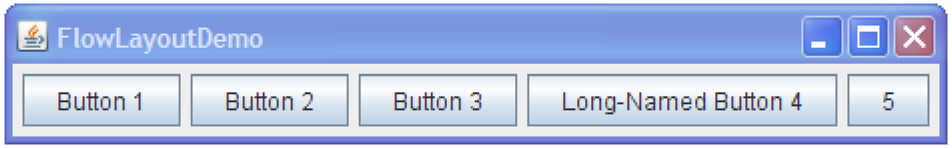
https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

Layout Management

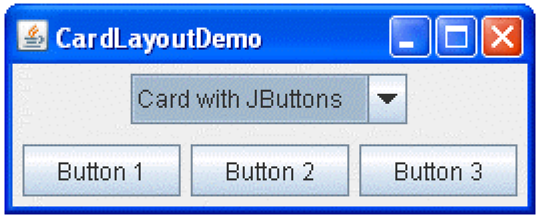


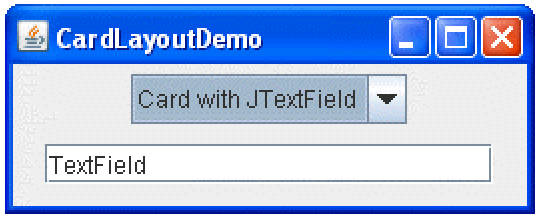


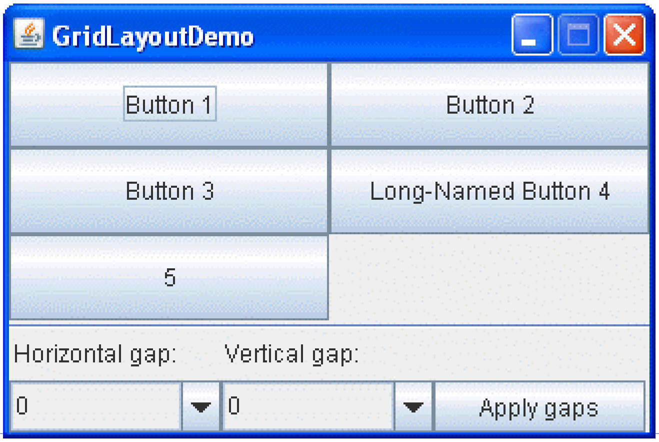
Every content pane is initialized to use a BorderLayout

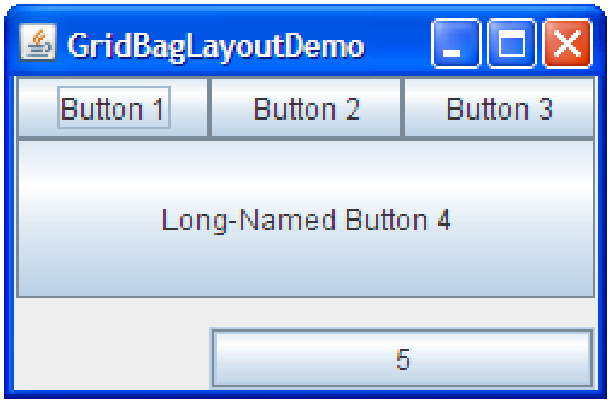


Every content pane is initialized to use a BorderLayout



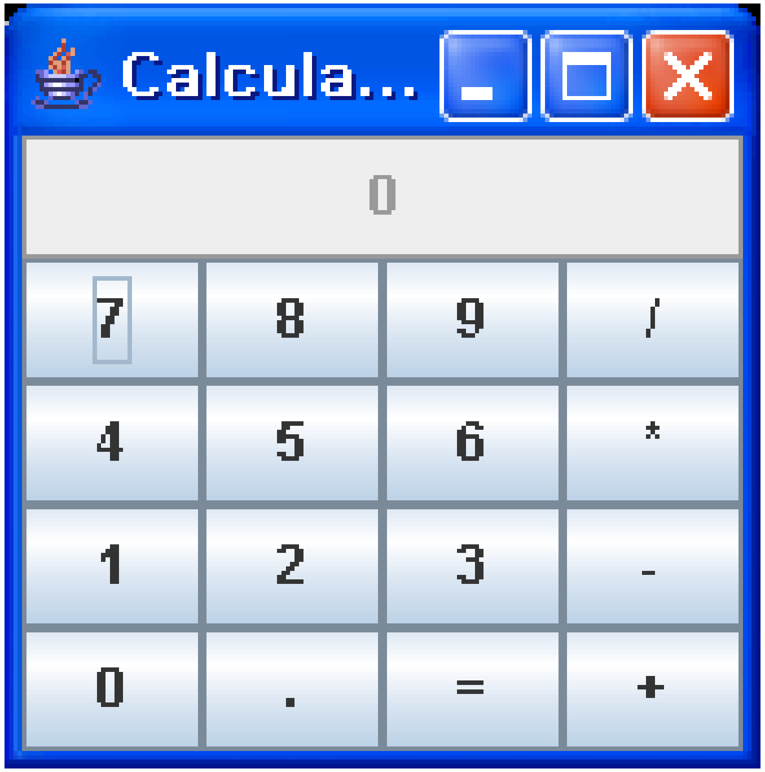






Layout Management: Example

- Calculator with BorderLayout Manger and GridLayout Manager



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

public class Calculator {
  public static void main(String[] args) {
    CalculatorFrame frame = new CalculatorFrame();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
class CalculatorFrame extends JFrame {
  public CalculatorFrame() {
    setTitle("Calculator");
    CalculatorPanel panel = new CalculatorPanel();
    add(panel);
    pack();
    // java.awt.Window.pack(): Causes this Window to be sized to fit
    // the preferred size and layouts of its subcomponents.
  }
}
```

```
class CalculatorPanel extends JPanel {
 public CalculatorPanel() {
   setLayout(new BorderLayout());
   // North, West, Center, East, South
   // the default layout manager for a panel: flow layout manager

   result = 0; lastCommand = "="; start = true;

   display = new JButton("0"); display.setEnabled(false);
   add(display, BorderLayout.NORTH);

   ActionListener insert = new InsertAction();
   ActionListener command = new CommandAction();
   // The grid layout arranges all components in rows and columns like a spreadsheet.
   commandPanel = new JPanel();
   commandPanel.setLayout(new GridLayout(4, 4));
   addButton("7", insert); addButton("8", insert);
   addButton("9", insert); addButton("/", command);
   addButton("4", insert); addButton("5", insert);
   addButton("6", insert); addButton("*", command);
   addButton("1", insert); addButton("2", insert);
   addButton("3", insert); addButton("-", command);
   addButton("0", insert); addButton(".", insert);
   addButton("=", command); addButton("+", command);

   add(commandPanel, BorderLayout.CENTER);
 }
 private void addButton(String label, ActionListener listener) {
    JButton button = new JButton(label);
    button.addActionListener(listener);
    commandPanel.add(button);
 }
 private class InsertAction implements ActionListener {
    public void actionPerformed(ActionEvent event) {
       String input = event.getActionCommand();
       if (start)  { display.setText(""); start = false; }
       display.setText(display.getText() + input);
    }
 }
 private class CommandAction implements ActionListener {
    public void actionPerformed(ActionEvent event) {
       String command = event.getActionCommand();
       if (start) {
          if (command.equals("-"))  {  display.setText(command);  start = false;  }
          else  lastCommand = command;
       } else {
          calculate(Double.parseDouble(display.getText()));
          lastCommand = command;
          start = true;
       }
    }
 }
```

```
/**
  * Carries out the pending calculation.
  * @param x the value to be accumulated with the prior result.
  */
  public void calculate(double x)
  {
     if (lastCommand.equals("+")) result += x;
     else if (lastCommand.equals("-")) result -= x;
     else if (lastCommand.equals("*")) result *= x;
     else if (lastCommand.equals("/")) result /= x;
     else if (lastCommand.equals("=")) result = x;
     display.setText("" + result);
  }
  private JButton display;
  private JPanel panel;
  private double result;
  private String lastCommand;
  private boolean start;
}
```