

GUI Programming with Swing

Core Java Volume I – Fundamentals

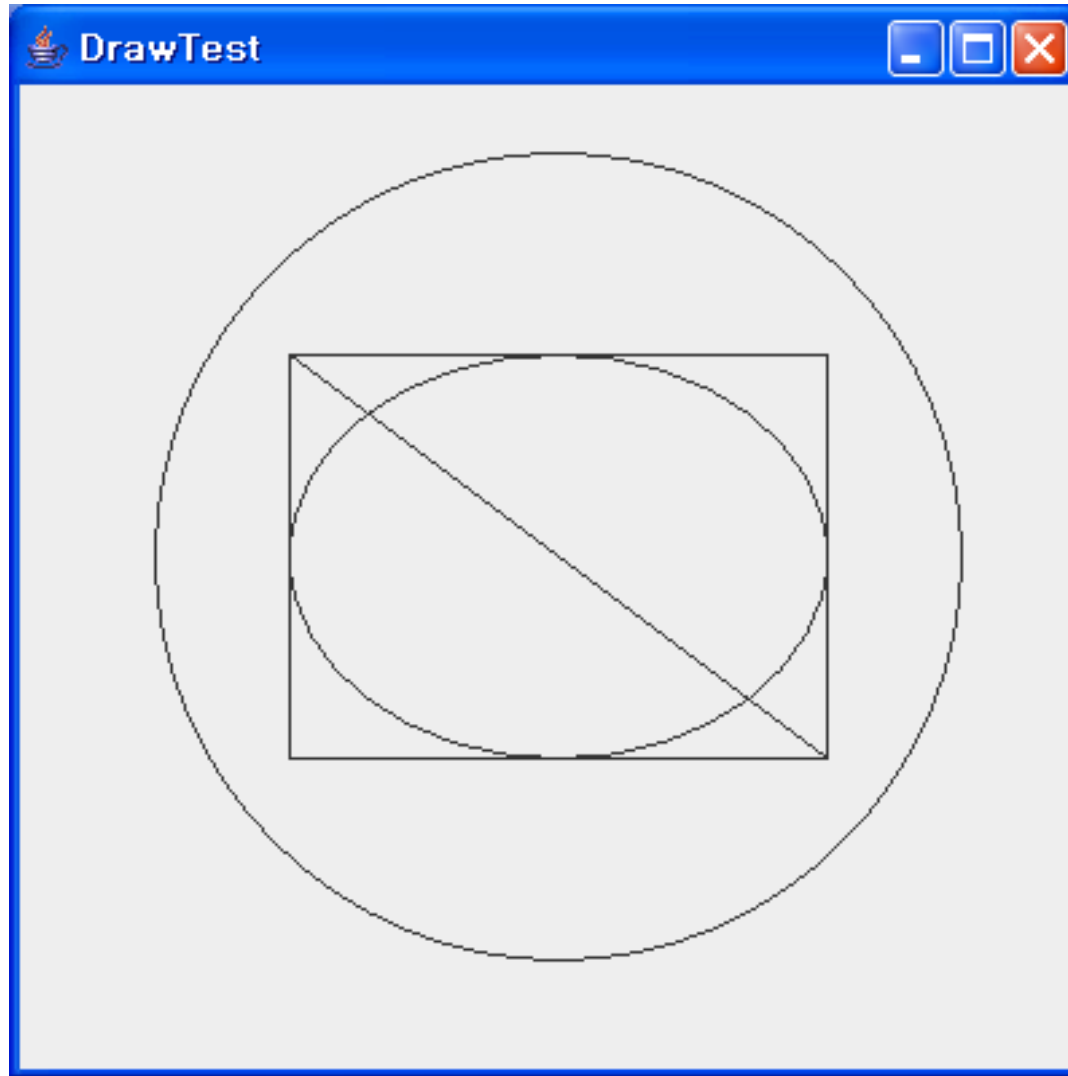
- Chapter 10. Graphical User Interface Programming
- Chapter 11. User Interface Components with Swing

Displaying Information in a Panel



```
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



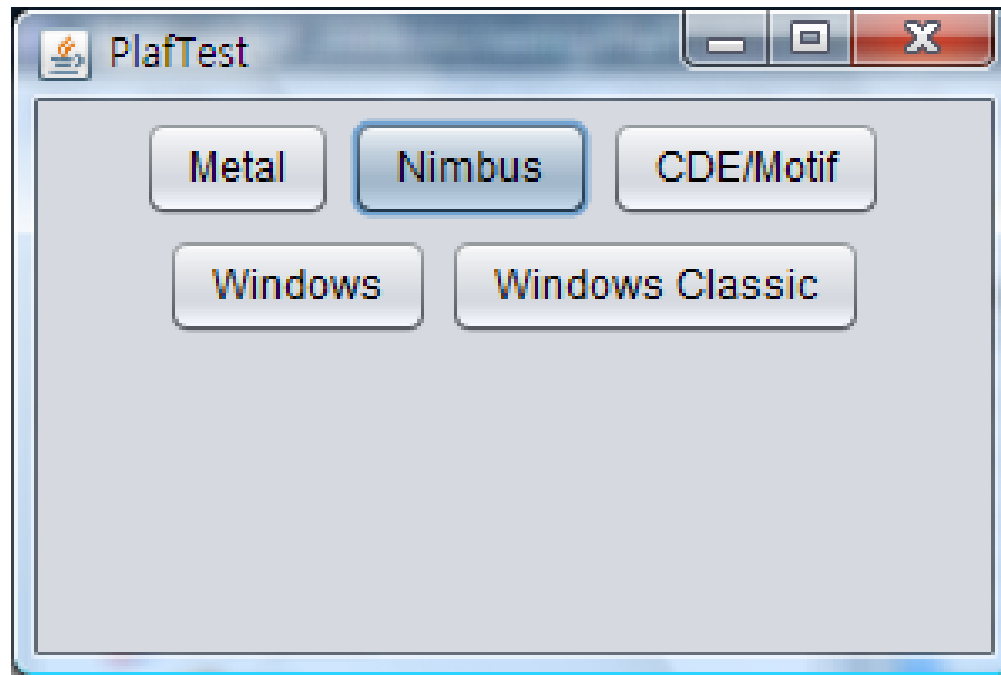
```
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;

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);  
    }  
}
```

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(); }
            }
        });
    }
}

```

Java Tutorial: Creating a GUI with JFC/Swing

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

Using Swing Components: Example

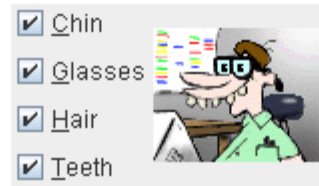
<https://docs.oracle.com/javase/tutorial/uiswing/examples/components/index.html>

Basic Controls

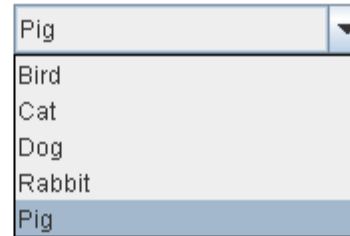
❖ Simple components get input from the user



[JButton](#)



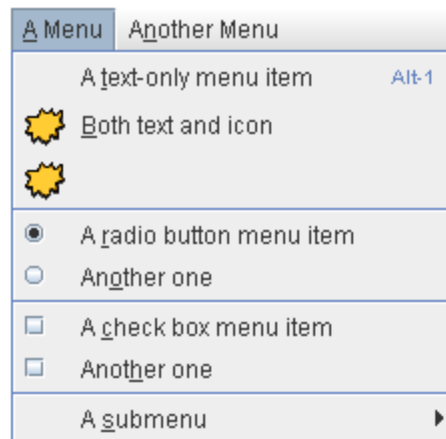
[JCheckBox](#)



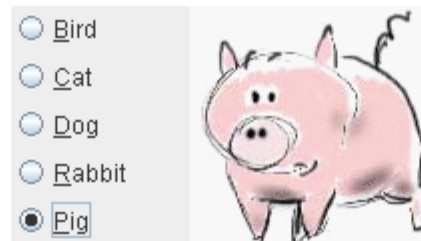
[JComboBox](#)



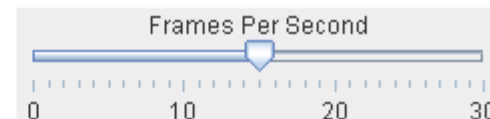
[JList](#)



[JMenu](#)



[JRadioButton](#)



[JSlider](#)



[JSpinner](#)

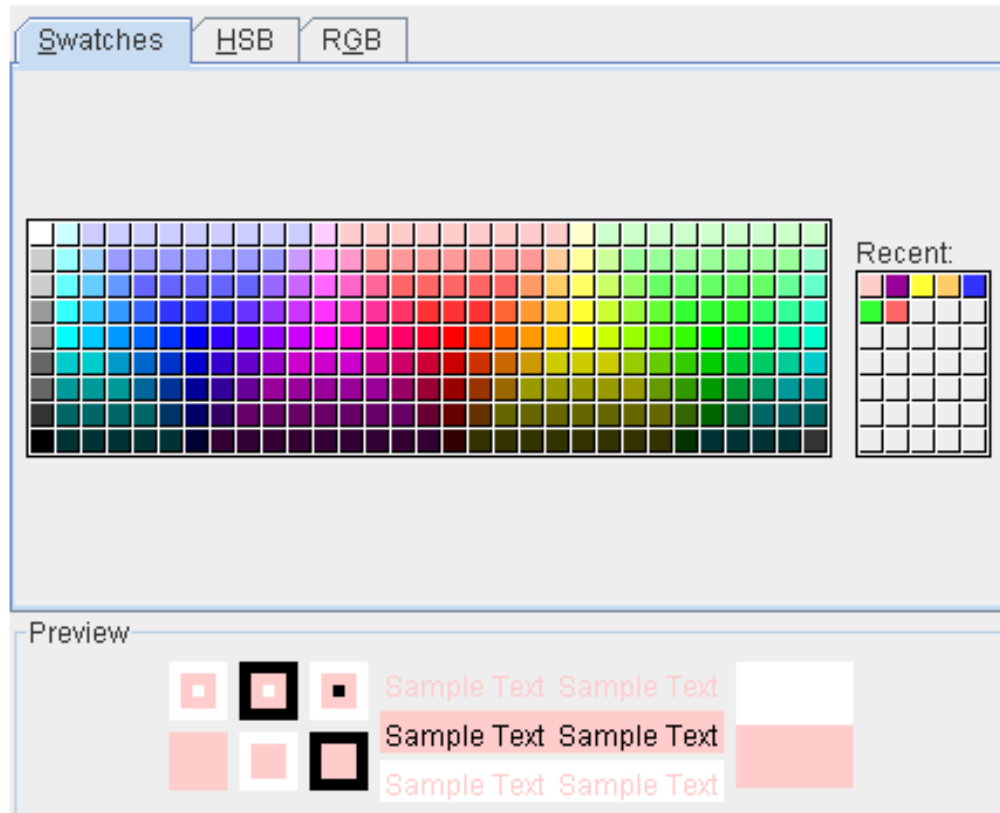


[JTextField](#)

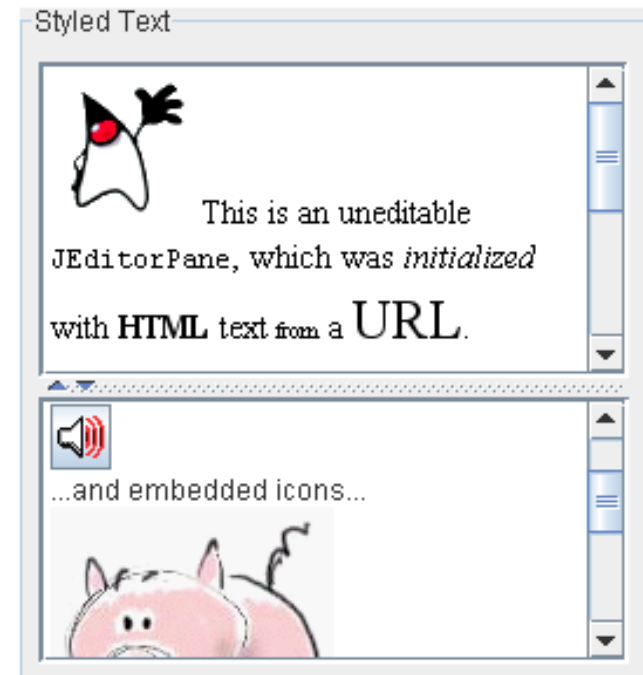


[JPasswordField](#)

Interactive Displays of Highly Formatted Information

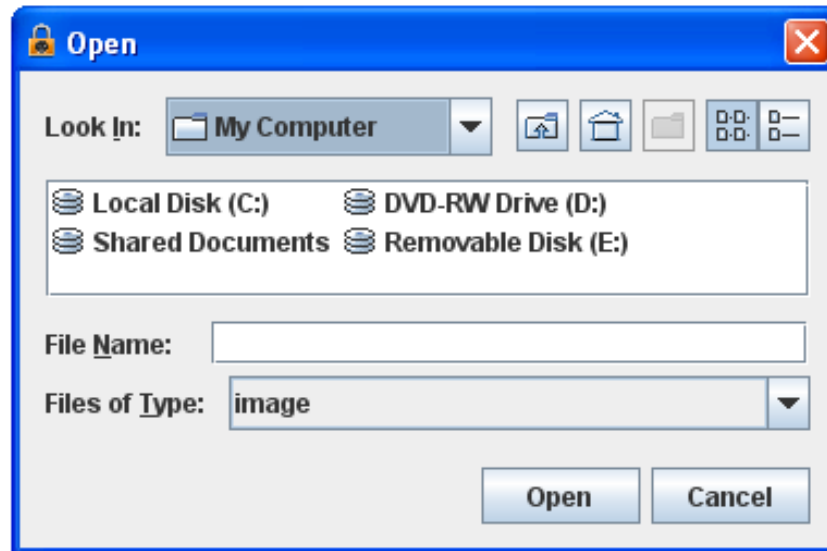


[JColorChooser](#)



[JEditorPane](#) and [JTextPane](#)

Interactive Displays of Highly Formatted Information



[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...	vbAf1 24%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](#)

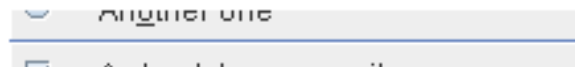
Uneditable Information Displays



[JLabel](#)



[JProgressBar](#)

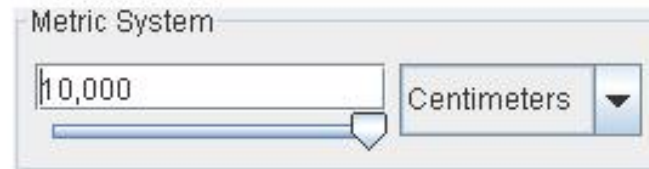


[JSeparator](#)



[JToolTip](#)

Containers



[JPanel](#)



[JScrollPane](#)



[JSplitPane](#)



[JTabbedPane](#)

[JToolBar](#)

Contents

❖ Text Input

- Text Fields, Formatted Text, Text Area

❖ Choice Components

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

❖ Menus

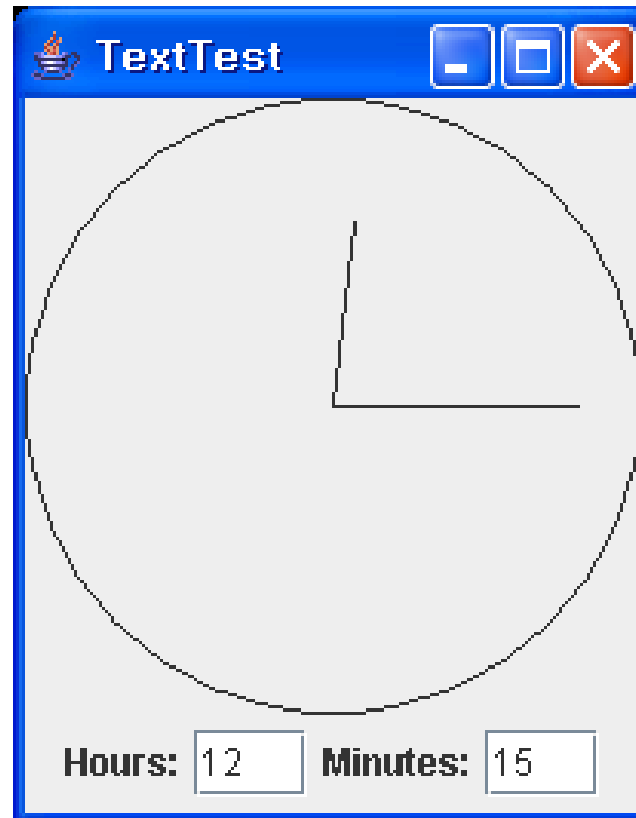
- Menu building, icons in menu items, keyboard mnemonics and accelerators, toolbars, tooltips

❖ Dialog Boxes

- Option dialogs, file dialogs, color choosers

❖ Layout Management

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 pane is BorderLayout  
  
        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

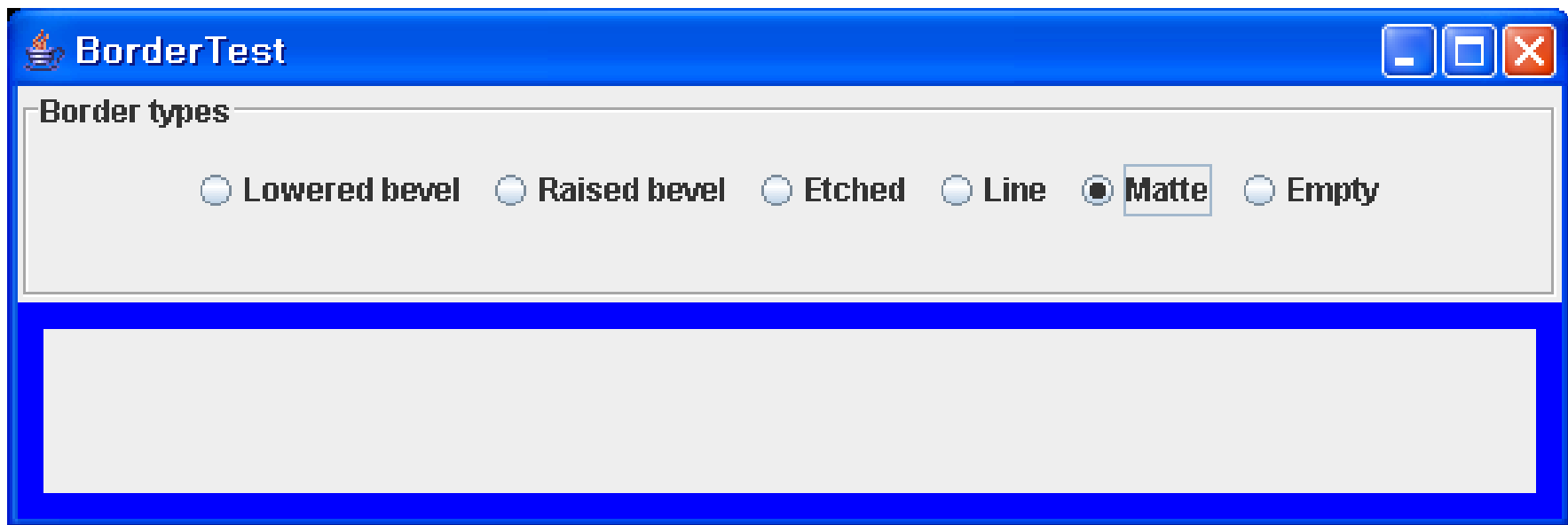
        double minuteAngle = Math.toRadians(90 - 360 * minutes / 60);
        drawHand(g2, minuteAngle, MINUTE_HAND_LENGTH);
    }
}
```

```
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;  
}
```

Borders



```
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);
    }
}

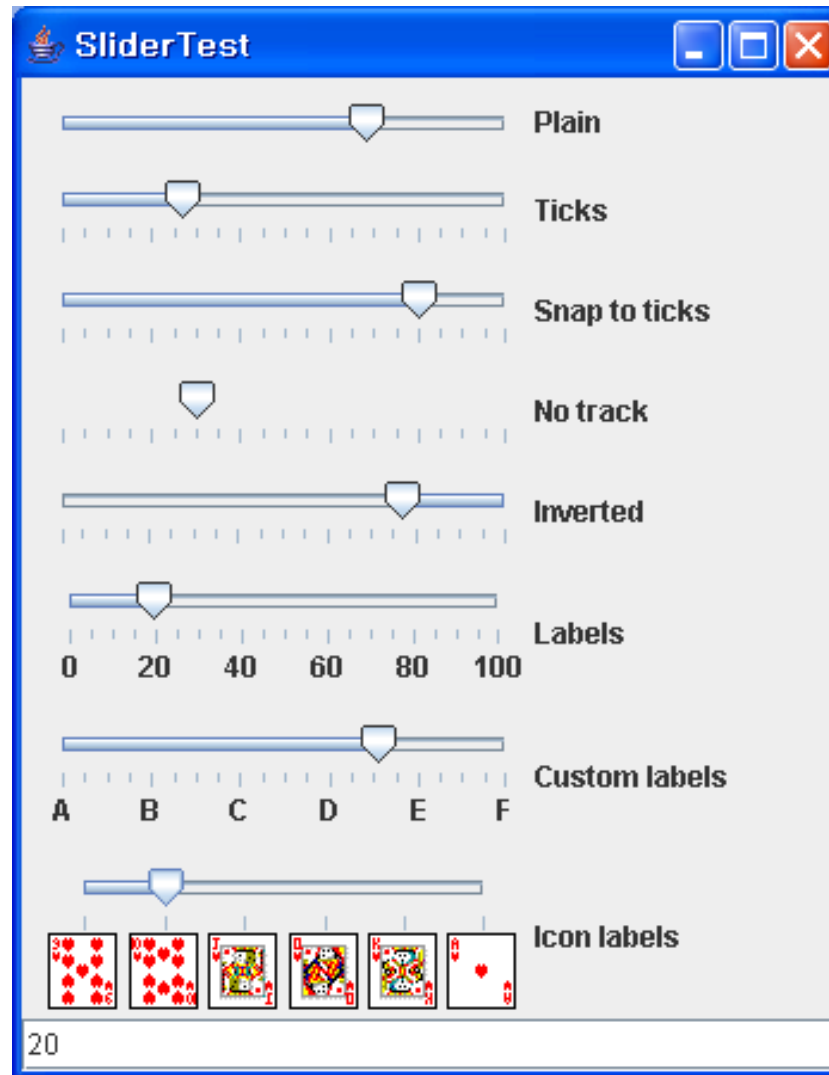
class BorderFrame extends JFrame {
    public BorderFrame() {
        setTitle("BorderTest"); setSize(DEFAULT_WIDTH, DEFAULT_HEIGHT);
        demoPanel = new JPanel();
        buttonPanel = new JPanel();
        group = new ButtonGroup();
        addRadioButton("Lowered bevel", BorderFactory.createLoweredBevelBorder());
        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;
}
```

Sliders



```
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>();
```

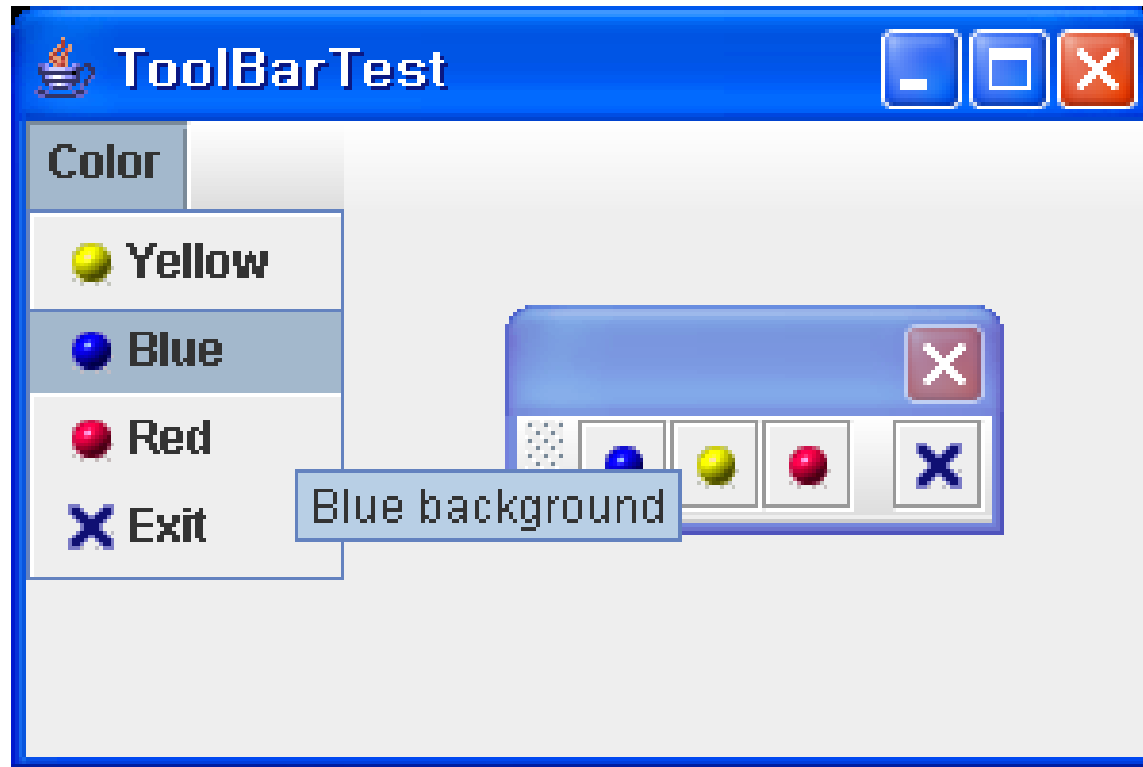
```
// add card images  
labelTable.put(0, new JLabel(new ImageIcon("nine.gif")));  
labelTable.put(20, new JLabel(new ImageIcon("ten.gif")));  
labelTable.put(40, new JLabel(new ImageIcon("jack.gif")));  
labelTable.put(60, new JLabel(new ImageIcon("queen.gif")));  
labelTable.put(80, new JLabel(new 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;
}
```

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);
```

```
}
```

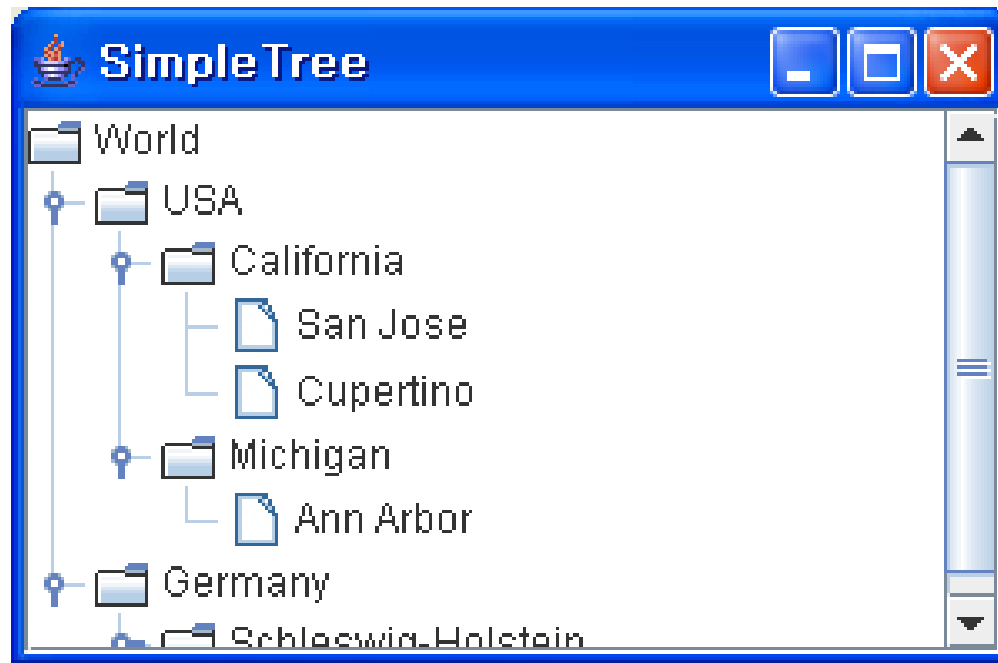
```
public static final int DEFAULT_WIDTH = 300;
public static final int DEFAULT_HEIGHT = 200;

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);
```

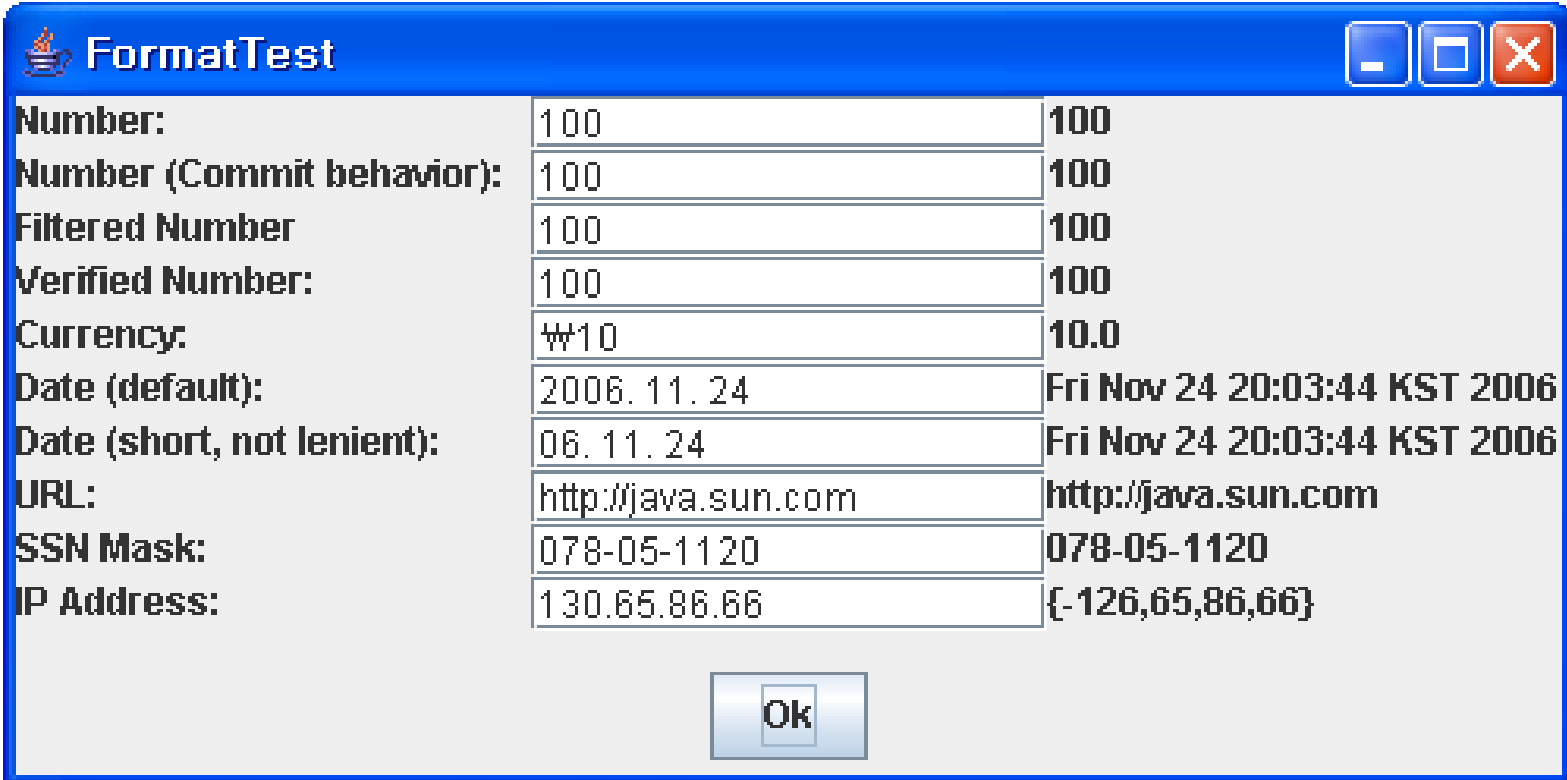
```
// construct tree and put it in a scroll pane
JTree tree = new JTree(root);
Container contentPane = getContentPane();
contentPane.add(new JScrollPane(tree));
```

```
}
```

```
private static final int DEFAULT_WIDTH = 300, DEFAULT_HEIGHT = 200;
```

```
}
```

Formatted Input



The screenshot shows a Java Swing window titled "FormatTest" with a standard Mac OS X-style title bar (blue with minimize, maximize, and close buttons). The window contains a table of formatted text fields. The fields are arranged in three columns: a label, an input field, and the formatted output. The labels are: "Number:", "Number (Commit behavior):", "Filtered Number", "Verified Number:", "Currency:", "Date (default):", "Date (short, not lenient):", "URL:", "SSN Mask:", and "IP Address:". The input fields contain: "100", "100", "100", "100", "₩10", "2006. 11. 24", "06. 11. 24", "http://java.sun.com", "078-05-1120", and "130.65.86.66". The formatted outputs are: "100", "100", "100", "100", "10.0", "Fri Nov 24 20:03:44 KST 2006", "Fri Nov 24 20:03:44 KST 2006", "http://java.sun.com", "078-05-1120", and "{-126,65,86,66}". An "Ok" button is located at the bottom center of the window.

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}

Ok

How to Use Formatted Text Fields

<https://docs.oracle.com/javase/tutorial/uiswing/components/formattedtextfield.html>

```
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);
    }
}

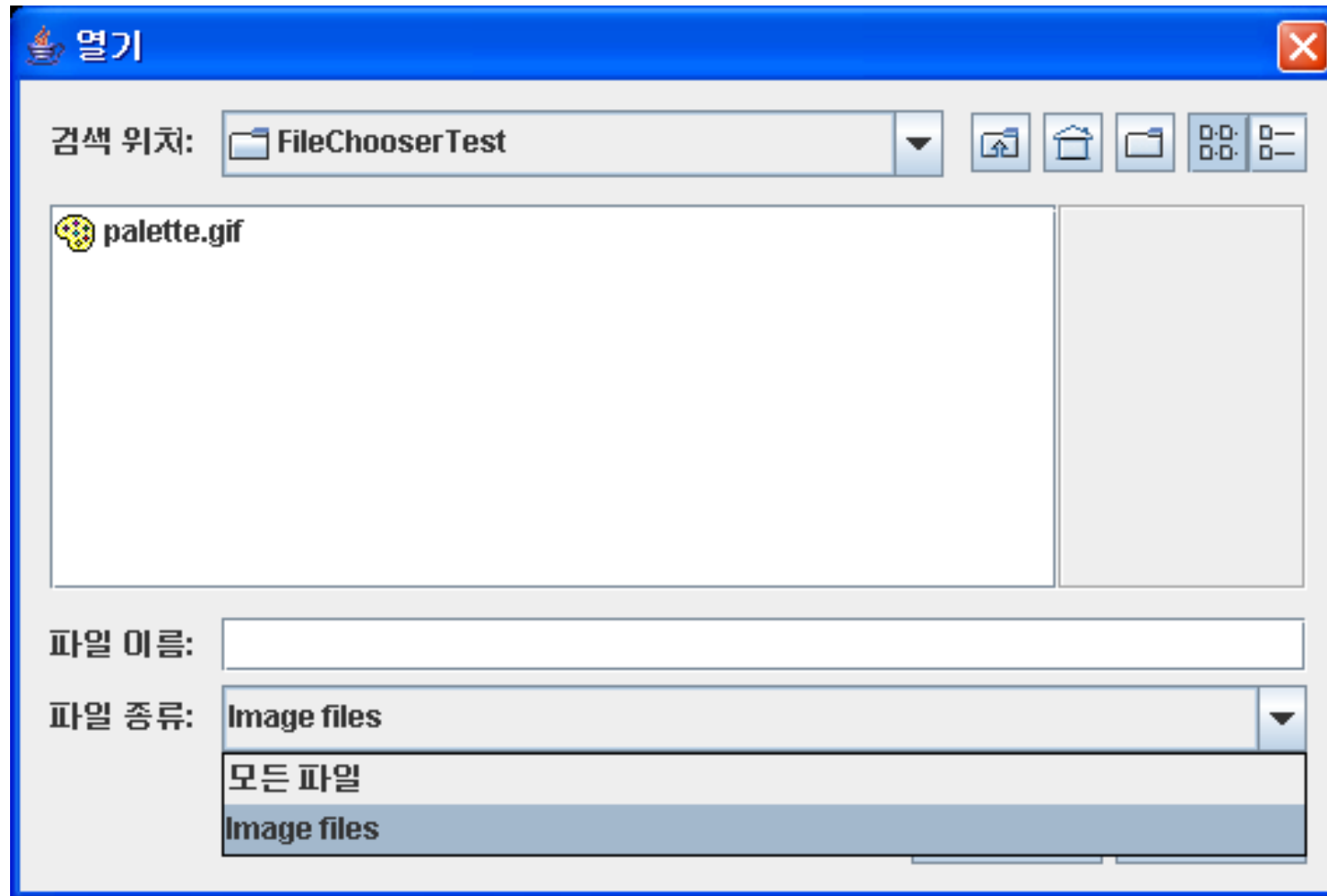
```

```
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());

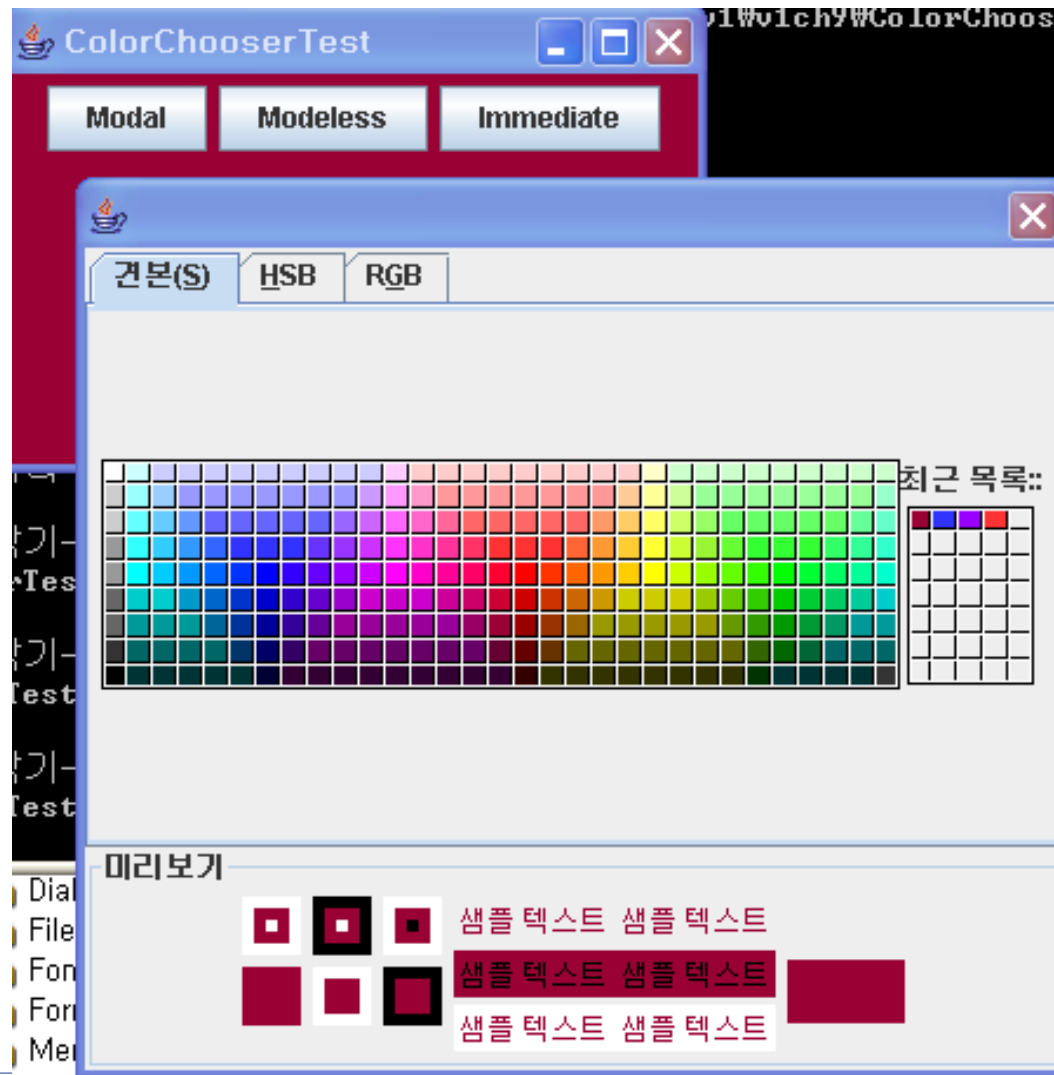
        chooser.addPropertyChangeListener(new PropertyChangeListener() {
            public void propertyChange(PropertyChangeEvent event) {
                if (event.getPropertyName() ==
                    JFileChooser.SELECTED_FILE_CHANGED_PROPERTY) {
                    // the user has selected a new file
                    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);  
    }  
}
```

```
}
```

```
/**
```

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;  
}
```

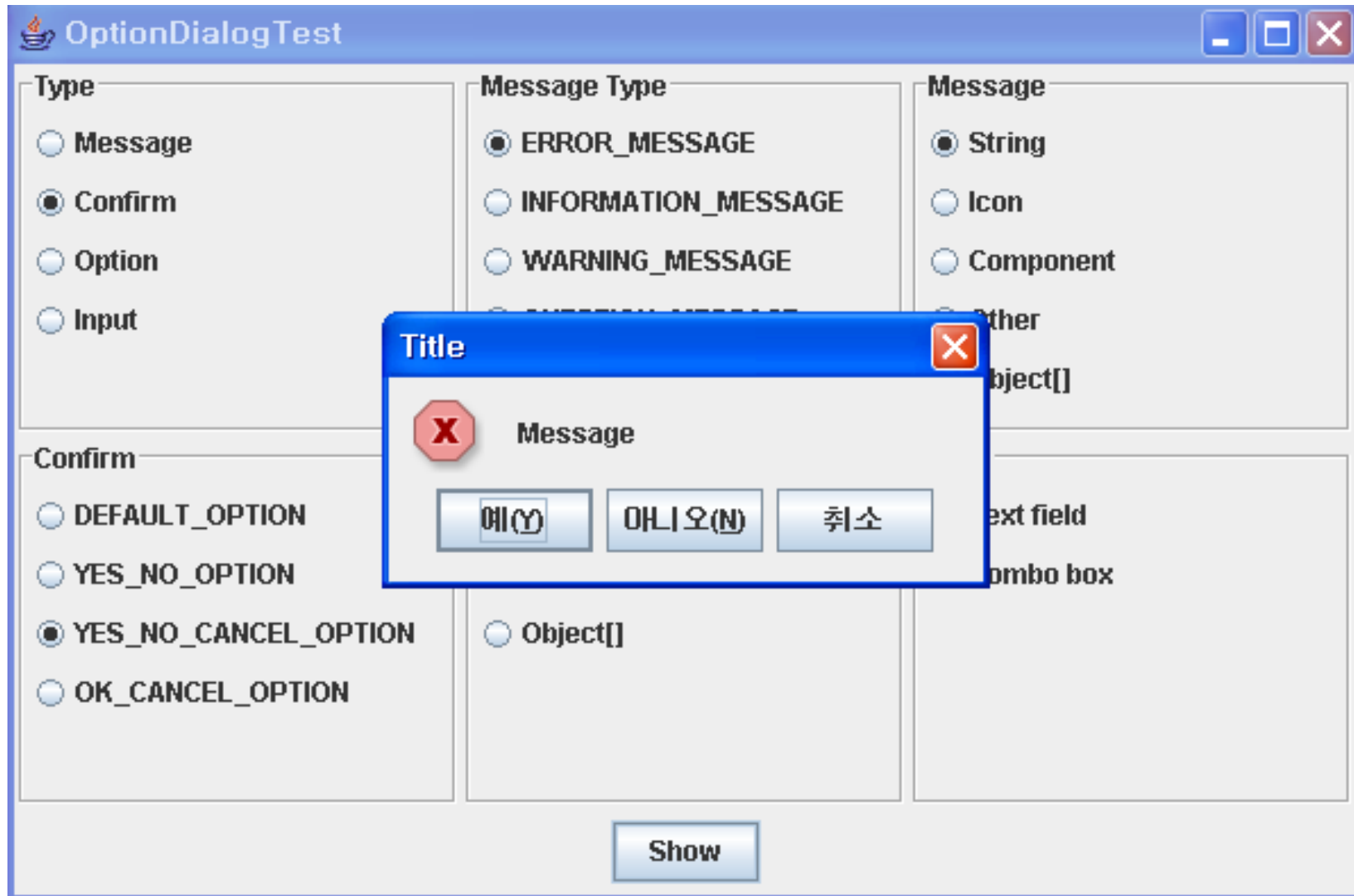
```
/**
```

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);}
}
```

Q&A
