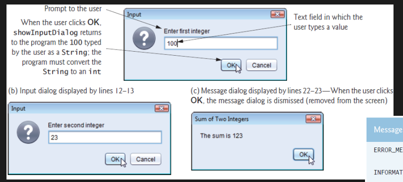
1

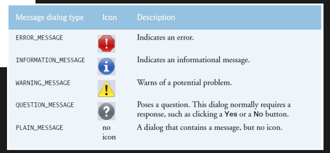
```java
import javax.swing.JOptionPane;

public class Addition
{
  public static void main(String[] args) {
    String firstNumber =
      JOptionPane.showInputDialog("Enter first integer");
    String secondNumber =
      JOptionPane.showInputDialog("Enter second integer");

    int number1 = Integer.parseInt(firstNumber);
    int number2 = Integer.parseInt(secondNumber);
    int sum = number1 + number2;

    JOptionPane.showMessageDialog(null, "The sum is " + sum,
     "Sum of Two Integers", JOptionPane.PLAIN_MESSAGE);
  }
}
```



When the user clicks OK, showInputDialog returns to the program the 100 typed by the user as a String; the program must convert the String to an int

(b) Input dialog displayed by lines 12–13

(c) Message dialog displayed by lines 22–23—When the user clicks OK, the message dialog is dismissed (removed from the screen)



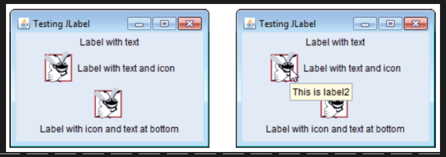
Message dialog type	Icon	Description
ERROR_MESSAGE	❗	Indicates an error.
INFORMATION_MESSAGE	ℹ	Indicates an informational message.
WARNING_MESSAGE	⚠	Warns of a potential problem.
QUESTION_MESSAGE	❓	Poses a question. This dialog normally requires a response, such as clicking a **Yes** or a **No** button.
PLAIN_MESSAGE	no icon	A dialog that contains a message, but no icon.

5

```java
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;+
import javax.swing.JTextField;
import javax.swing.JCheckBox;

public class CheckBoxFrame extends JFrame {
  private final JTextField textField;
  private final JCheckBox boldJCheckBox;
  private final JCheckBox italicJCheckBox;

  public CheckBoxFrame() {
    super("JCheckBox Test");
    setLayout(new FlowLayout());

    textField =
        new JTextField("Watch the font style change", 20);
    textField.setFont(new Font("Serif", Font.PLAIN, 14));
    add(textField);

    boldJCheckBox = new JCheckBox("Bold");
    italicJCheckBox = new JCheckBox("Italic");
    add(boldJCheckBox);
    add(italicJCheckBox);

    CheckBoxHandler handler = new CheckBoxHandler();
    boldJCheckBox.addItemListener(handler);
    italicJCheckBox.addItemListener(handler);
  }

  private class CheckBoxHandler implements ItemListener  {
    @Override public void itemStateChanged(ItemEvent event) {
      Font font = null;

      if (boldJCheckBox.isSelected() &&
            italicJCheckBox.isSelected())
        font = new Font("Serif", Font.BOLD + Font.ITALIC, 14);
      else if (boldJCheckBox.isSelected())
        font = new Font("Serif", Font.BOLD, 14);
      else if (italicJCheckBox.isSelected())
        font = new Font("Serif", Font.ITALIC, 14);
      else
        font = new Font("Serif", Font.PLAIN, 14);

      textField.setFont(font);
    }
  }
}
```

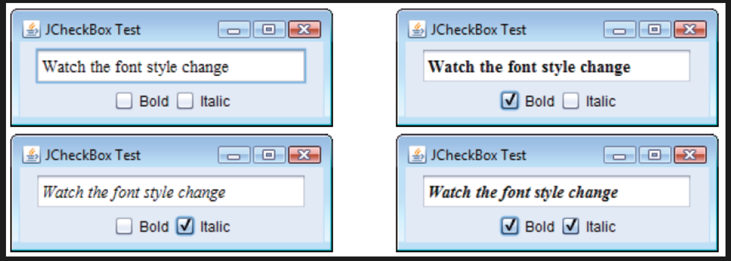
2

```java
import javax.swing.JFrame;

public class LabelTest
{
  public static void main(String[] args) {
    LabelFrame labelFrame = new LabelFrame();
    labelFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    labelFrame.setSize(260, 180);
    labelFrame.setVisible(true);
  }
}
```



```java
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.Icon;
import javax.swing.ImageIcon;

public class LabelFrame extends JFrame {
  private final JLabel label1;
  private final JLabel label2;
  private final JLabel label3;

  public LabelFrame() {
    super("Testing JLabel");
    setLayout(new FlowLayout());

    label1 = new JLabel("Label with text");
    label1.setToolTipText("This is label1");
    add(label1);

    Icon bug =
      new ImageIcon(getClass().getResource("bug1.png"));
    label2 = new JLabel("Label with text and icon",
                bug,  SwingConstants.LEFT);
    label2.setToolTipText("This is label2");
    add(label2);

    label3 = new JLabel();
    label3.setText("Label with icon and text at bottom");
    label3.setIcon(bug);
    label3.setHorizontalTextPosition(SwingConstants.CENTER);
    label3.setVerticalTextPosition(SwingConstants.BOTTOM);
    label3.setToolTipText("This is label3");
    add(label3);
  }
}
```

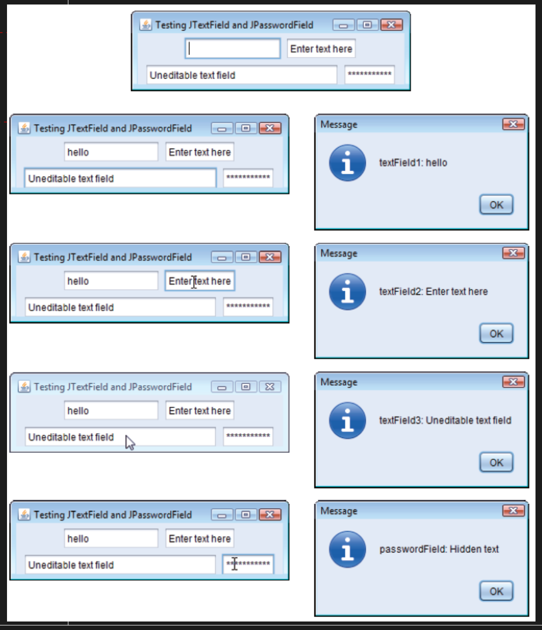
```java
import javax.swing.JFrame;

public class CheckBoxTest  {
  public static void main(String[] args) {
    CheckBoxFrame checkBoxFrame = new CheckBoxFrame();

    checkBoxFrame.setDefaultCloseOperation(
                    JFrame.EXIT_ON_CLOSE);
    checkBoxFrame.setSize(275, 100);
    checkBoxFrame.setVisible(true);
  }
}
```


JCheckBox Test
Watch the font style change
Bold
Italic
JCheckBox Test
Watch the font style change
Bold
Italic
JCheckBox Test
Watch the font style change
Bold
Italic
JCheckBox Test
Watch the font style change
Bold
Italic

3

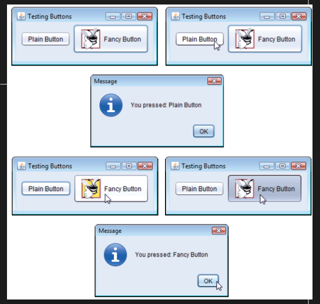
```java
import javax.swing.JFrame;

public class TextFieldTest {
  public static void main(String[] args) {
    TextFieldFrame textFieldFrame = new TextFieldFrame();
    textFieldFrame.setDefaultCloseOperation(
                    JFrame.EXIT_ON_CLOSE);
    textFieldFrame.setSize(350, 100);
    textFieldFrame.setVisible(true);
  }
}
```

```java
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JOptionPane;

public class TextFieldFrame extends JFrame {
  private final JTextField textField1;
  private final JTextField textField2;
  private final JTextField textField3;
  private final JPasswordField passwordField;

  public TextFieldFrame()  {
    super("Testing JTextField and JPasswordField");
    setLayout(new FlowLayout());

    textField1 = new JTextField(10);
    add(textField1);

    textField2 = new JTextField("Enter text here");
    add(textField2);

    textField3 = new JTextField("Uneditable text field", 21);
    textField3.setEditable(false);
    add(textField3);

    passwordField = new JPasswordField("Hidden text");
    add(passwordField);

    TextFieldHandler handler = new TextFieldHandler();
    textField1.addActionListener(handler);
    textField2.addActionListener(handler);
    textField3.addActionListener(handler);
    passwordField.addActionListener(handler);
  }

  private class TextFieldHandler implements ActionListener  {
    @Override  public void actionPerformed(ActionEvent event)  {
      String string = "";

      if (event.getSource() == textField1)
        string = String.format("textField1: %s",
          event.getActionCommand());

      else if (event.getSource() == textField2)
        string = String.format("textField2: %s",
          event.getActionCommand());

      else if (event.getSource() == textField3)
        string = String.format("textField3: %s",
          event.getActionCommand());

      else if (event.getSource() == passwordField)
        string = String.format("passwordField: %s",
          event.getActionCommand());

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



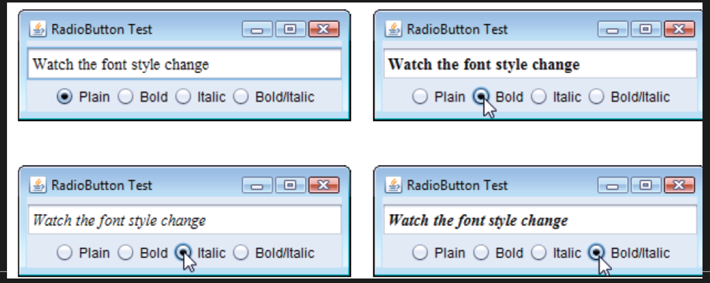
4

```java
import javax.swing.JFrame;

public class ButtonTest {
  public static void main(String[] args)  {
    ButtonFrame buttonFrame = new ButtonFrame();
    buttonFrame.setDefaultCloseOperation(
                    JFrame.EXIT_ON_CLOSE);
    buttonFrame.setSize(275, 110);
    buttonFrame.setVisible(true);
  }
}
```



```java
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

public class ButtonFrame extends JFrame {
  private final JButton plainJButton;
  private final JButton fancyJButton;

  public ButtonFrame() {
    super("Testing Buttons");
    setLayout(new FlowLayout());

    plainJButton = new JButton("Plain Button");
    add(plainJButton);

    Icon bug1 =
      new ImageIcon(getClass().getResource("bug1.gif"));
    Icon bug2 =
      new ImageIcon(getClass().getResource("bug2.gif"));
    fancyJButton = new JButton("Fancy Button", bug1);
    fancyJButton.setRolloverIcon(bug2);
    add(fancyJButton);

    ButtonHandler handler = new ButtonHandler();
    fancyJButton.addActionListener(handler);
    plainJButton.addActionListener(handler);
  }

  private class ButtonHandler implements ActionListener   {
    @Override public void actionPerformed(ActionEvent event) {
      JOptionPane.showMessageDialog(ButtonFrame.this,
        String.format( "You pressed: %s",
                  event.getActionCommand()));
    }
  }
}
```

```java
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;

public class RadioButtonFrame extends JFrame {
 private JTextField textField;
 private Font plainFont;
 private Font boldFont;
 private Font italicFont;
 private Font boldItalicFont;
 private JRadioButton plainJRadioButton;
 private JRadioButton boldJRadioButton;
 private JRadioButton italicJRadioButton;
 private JRadioButton boldItalicJRadioButton;
 private ButtonGroup radioGroup;

 public RadioButtonFrame()  {
  super("RadioButton Test");
  setLayout(new FlowLayout());

  textField = new JTextField("Watch the font style change", 25);
  add(textField);

  plainJRadioButton      = new JRadioButton("Plain", true);
  boldJRadioButton       = new JRadioButton("Bold", false);
  italicJRadioButton     = new JRadioButton("Italic", false);
  boldItalicJRadioButton = new JRadioButton("Bold/Italic", false);
  add(plainJRadioButton);
  add(boldJRadioButton);
  add(italicJRadioButton);
  add(boldItalicJRadioButton);

  radioGroup = new ButtonGroup();
  radioGroup.add(plainJRadioButton);
  radioGroup.add(boldJRadioButton);
  radioGroup.add(italicJRadioButton);
  radioGroup.add(boldItalicJRadioButton);

  plainFont      = new Font("Serif", Font.PLAIN, 14);
  boldFont       = new Font("Serif", Font.BOLD, 14);
  italicFont     = new Font("Serif", Font.ITALIC, 14);
  boldItalicFont = new Font("Serif", Font.BOLD + Font.ITALIC, 14);
  textField.setFont(plainFont);

  plainJRadioButton.addItemListener(
   new RadioButtonHandler(plainFont));
  boldJRadioButton.addItemListener(
   new RadioButtonHandler(boldFont));
  italicJRadioButton.addItemListener(
   new RadioButtonHandler(italicFont));
  boldItalicJRadioButton.addItemListener(
   new RadioButtonHandler(boldItalicFont));
 }

 private class RadioButtonHandler implements ItemListener  {
  private Font font;

  public RadioButtonHandler(Font f)  { font = f;  }

  @Override public void itemStateChanged(ItemEvent event)  {
   textField.setFont(font);
  }
 }
}
```


RadioButton Test — Watch the font style change — [● Plain] ○ Bold ○ Italic ○ Bold/Italic
RadioButton Test — Watch the font style change — ○ Plain [● Bold] ○ Italic ○ Bold/Italic
RadioButton Test — Watch the font style change — ○ Plain ○ Bold [● Italic] ○ Bold/Italic
RadioButton Test — Watch the font style change — ○ Plain ○ Bold ○ Italic [● Bold/Italic]

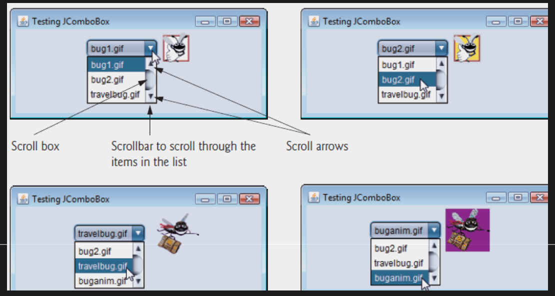
6

```java
import javax.swing.JFrame;

public class RadioButtonTest {
 public static void main(String[] args) {
  RadioButtonFrame radioButtonFrame =
     new RadioButtonFrame();
  radioButtonFrame.setDefaultCloseOperation(
                 JFrame.EXIT_ON_CLOSE);
  radioButtonFrame.setSize(300, 100);
  radioButtonFrame.setVisible(true);
 }
}
```

7

```java
import javax.swing.JFrame;

public class ComboBoxTest {
 public static void main(String[] args) {
  ComboBoxFrame comboBoxFrame = new ComboBoxFrame();

  comboBoxFrame.setDefaultCloseOperation(
                  JFrame.EXIT_ON_CLOSE);
  comboBoxFrame.setSize(350, 150);
  comboBoxFrame.setVisible(true);
 }
}
```

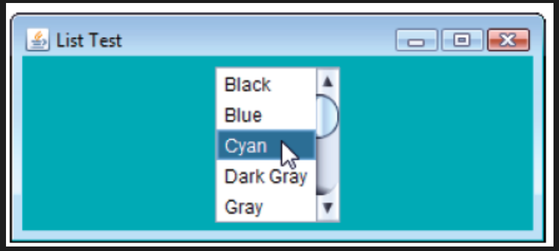
```java
import java.awt.FlowLayout;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import javax.swing.Icon;
import javax.swing.ImageIcon;

public class ComboBoxFrame extends JFrame {
 private final JComboBox<String> imagesJComboBox;
 private final JLabel label;

 private static final String[] names =
  {"bug1.gif", "bug2.gif",  "travelbug.gif", "buganim.gif"};
 private final Icon[] icons = {
  new ImageIcon(getClass().getResource(names[0])),
  new ImageIcon(getClass().getResource(names[1])),
  new ImageIcon(getClass().getResource(names[2])),
  new ImageIcon(getClass().getResource(names[3]))
 };

 public ComboBoxFrame()  {
  super("Testing JComboBox");
  setLayout(new FlowLayout());

  imagesJComboBox = new JComboBox<String>(names);
  imagesJComboBox.setMaximumRowCount(3);

  imagesJComboBox.addItemListener(
   new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent event) {
     if (event.getStateChange() == ItemEvent.SELECTED)
      label.setIcon(icons[
       imagesJComboBox.getSelectedIndex()]);
    }
   }
  );

  add(imagesJComboBox);
  label = new JLabel(icons[0]);
  add(label);
 }
}
```

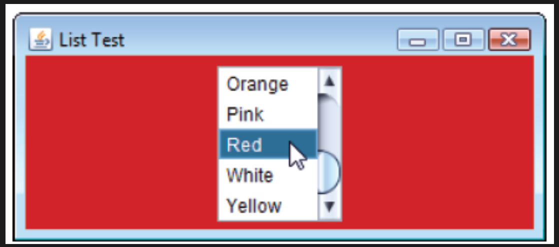


8

```java
import javax.swing.JFrame;

public class ListTest
{
 public static void main(String[] args)
 {
  ListFrame listFrame = new ListFrame();

listFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  listFrame.setSize(350, 150);
  listFrame.setVisible(true);
 }
}
```

```java
import java.awt.FlowLayout;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JScrollPane;
import javax.swing.event.ListSelectionListener;
import javax.swing.event.ListSelectionEvent;
import javax.swing.ListSelectionModel;

public class ListFrame extends JFrame {
 private final JList<String> colorJList;
 private static final String[] colorNames = {
    "Black",  "Blue",  "Cyan",         "Dark Gray",
    "Gray",   "Green", "Light Gray",   "Magenta",
    "Orange", "Pink",  "Red", "White", "Yellow"};

 private static final Color[] colors = {
   Color.BLACK,     Color.BLUE,    Color.CYAN,
   Color.DARK_GRAY,  Color.GRAY,    Color.GREEN,
   Color.LIGHT_GRAY, Color.MAGENTA, Color.ORANGE,
   Color.PINK,       Color.RED, Color.WHITE,
   Color.YELLOW};

 public ListFrame()  {
  super("List Test");
  setLayout(new FlowLayout());

  colorJList = new JList<String>(colorNames);
  colorJList.setVisibleRowCount(5);
  colorJList.setSelectionMode(
        ListSelectionModel.SINGLE_SELECTION);
  add(new JScrollPane(colorJList));

  colorJList.addListSelectionListener(
   new ListSelectionListener() {
    @Override
    public void valueChanged(ListSelectionEvent event) {
      getContentPane().setBackground(
              colors[colorJList.getSelectedIndex()]);
    }
   }
  );
 }
}
```

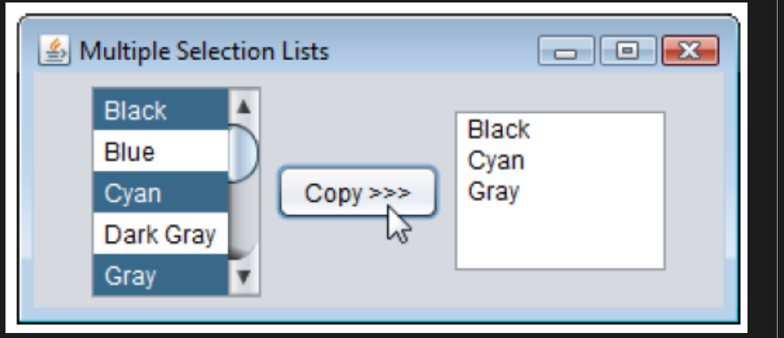




9

```java
import javax.swing.JFrame;

public class MultipleSelectionTest {
 public static void main(String[] args) {
  MultipleSelectionFrame multipleSelectionFrame =
     new MultipleSelectionFrame();
  multipleSelectionFrame.setDefaultCloseOperation(
                JFrame.EXIT_ON_CLOSE);
  multipleSelectionFrame.setSize(350, 150);
  multipleSelectionFrame.setVisible(true);
 }
}
```

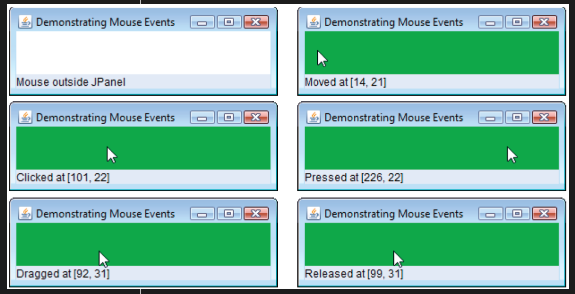
```java
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JList;
import javax.swing.JButton;
import javax.swing.JScrollPane;
import javax.swing.ListSelectionModel;

public class MultipleSelectionFrame extends JFrame {
 private final JList<String> colorJList;
 private final JList<String> copyJList;
 private JButton copyJButton;
 private static final String[] colorNames ={
    "Black",  "Blue",  "Cyan",         "Dark Gray",
    "Gray",   "Green", "Light Gray",   "Magenta",
    "Orange", "Pink",  "Red", "White", "Yellow"};

 public MultipleSelectionFrame()  {
  super("Multiple Selection Lists");
  setLayout(new FlowLayout());

  colorJList = new JList<String>(colorNames);
  colorJList.setVisibleRowCount(5);
  colorJList.setSelectionMode(
   ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
  add(new JScrollPane(colorJList));

  copyJButton = new JButton("Copy >>>");
  copyJButton.addActionListener(
   new ActionListener() {
    @Override public void actionPerformed(ActionEvent event) {
      copyJList.setListData(
        colorJList.getSelectedValuesList().toArray(
          new String[0])
       );
    }
   }
  );

  add(copyJButton);

  copyJList = new JList<String>();
  copyJList.setVisibleRowCount(5);
  copyJList.setFixedCellWidth(100);
  copyJList.setFixedCellHeight(15);
  copyJList.setSelectionMode(
   ListSelectionModel.SINGLE_INTERVAL_SELECTION);
  add(new JScrollPane(copyJList));
 }
}
```



9

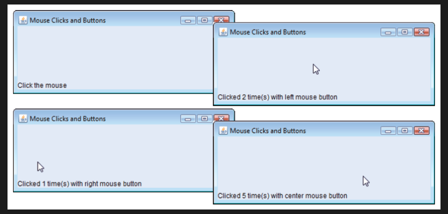
```java
import javax.swing.JFrame;

public class MouseTracker {
  public static void main(String[] args) {
    MouseTrackerFrame mouseTrackerFrame = new MouseTrackerFrame();
    mouseTrackerFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    mouseTrackerFrame.setSize(300, 100);
    mouseTrackerFrame.setVisible(true);
  }
}
```

```java
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
import java.awt.event.MouseEvent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class MouseTrackerFrame extends JFrame {
  private final JPanel mousePanel;
  private final JLabel statusBar;

    public MouseTrackerFrame() {
    super("Demonstrating Mouse Events");

    mousePanel = new JPanel();
    mousePanel.setBackground(Color.WHITE);
    add(mousePanel, BorderLayout.CENTER);

    statusBar = new JLabel("Mouse outside JPanel");
    add(statusBar, BorderLayout.SOUTH);

    MouseHandler handler = new MouseHandler();
    mousePanel.addMouseListener(handler);
    mousePanel.addMouseMotionListener(handler);
  }

  private class MouseHandler
        implements MouseListener, MouseMotionListener  {

    @Override public void mouseClicked(MouseEvent event) {
      statusBar.setText(String.format("Clicked at [%d, %d]",
                    event.getX(), event.getY()));
    }

    @Override  public void mousePressed(MouseEvent event)  {
      statusBar.setText(String.format("Pressed at [%d, %d]",
                    event.getX(), event.getY()));
    }

    @Override public void mouseReleased(MouseEvent event)  {
      statusBar.setText(String.format("Released at [%d, %d]",
                     event.getX(), event.getY()));
    }

    @Override public void mouseEntered(MouseEvent event)  {
      statusBar.setText(String.format("Mouse entered at [%d, %d]",
               event.getX(), event.getY()));
      mousePanel.setBackground(Color.GREEN);
    }

    @Override public void mouseExited(MouseEvent event)  {
      statusBar.setText("Mouse outside JPanel");
      mousePanel.setBackground(Color.WHITE);
    }

    @Override public void mouseDragged(MouseEvent event)  {
      statusBar.setText(String.format("Dragged at [%d, %d]",
                    event.getX(), event.getY()));
    }

    @Override public void mouseMoved(MouseEvent event)  {
      statusBar.setText(String.format("Moved at [%d, %d]",
        event.getX(), event.getY()));
    }
  }
}
```


Demonstrating Mouse Events — Mouse outside JPanel
Demonstrating Mouse Events — Moved at [14, 21]
Demonstrating Mouse Events — Clicked at [101, 22]
Demonstrating Mouse Events — Pressed at [226, 22]
Demonstrating Mouse Events — Dragged at [92, 31]
Demonstrating Mouse Events — Released at [99, 31]

10

```java
import javax.swing.JFrame;

public class MouseDetails {
  public static void main(String[] args)  {
    MouseDetailsFrame mouseDetailsFrame =
            new MouseDetailsFrame();
    mouseDetailsFrame.setDefaultCloseOperation(
                  JFrame.EXIT_ON_CLOSE);
    mouseDetailsFrame.setSize(400, 150);
    mouseDetailsFrame.setVisible(true);
  }
}
```

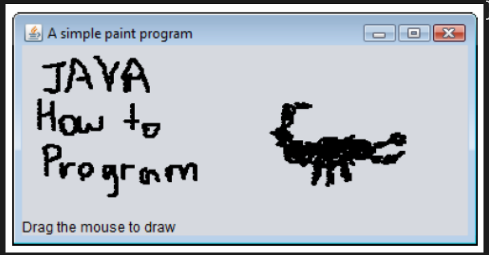

Mouse Clicks and Buttons — Click the mouse
Mouse Clicks and Buttons — Clicked 2 time(s) with left mouse button
Mouse Clicks and Buttons — Clicked 1 time(s) with right mouse button
Mouse Clicks and Buttons — Clicked 5 time(s) with center mouse button

```java
import java.awt.BorderLayout;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class MouseDetailsFrame extends JFrame {
  private String details;
  private final JLabel statusBar;

  public MouseDetailsFrame()  {
    super("Mouse Clicks and Buttons");

    statusBar = new JLabel("Click the mouse");
    add(statusBar, BorderLayout.SOUTH);
    addMouseListener(new MouseClickHandler());
  }

  private class MouseClickHandler extends MouseAdapter {

    @Override
    public void mouseClicked(MouseEvent event) {
      int xPos = event.getX();
      int yPos = event.getY();

      details = String.format("Clicked %d time(s)",
                  event.getClickCount());

      if (event.isMetaDown())
        details += " with right mouse button";
      else if (event.isAltDown())
        details += " with center mouse button";
      else
        details += " with left mouse button";

      statusBar.setText(details);
    }
  }
}
```

11

```java
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class Painter {
  public static void main(String[] args)  {

    JFrame application = new JFrame("A simple paint program");

    PaintPanel paintPanel = new PaintPanel();
    application.add(paintPanel, BorderLayout.CENTER);

    application.add(new JLabel("Drag the mouse to draw"),
                    BorderLayout.SOUTH);

    application.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    application.setSize(400, 200);
    application.setVisible(true);
  }
}
```

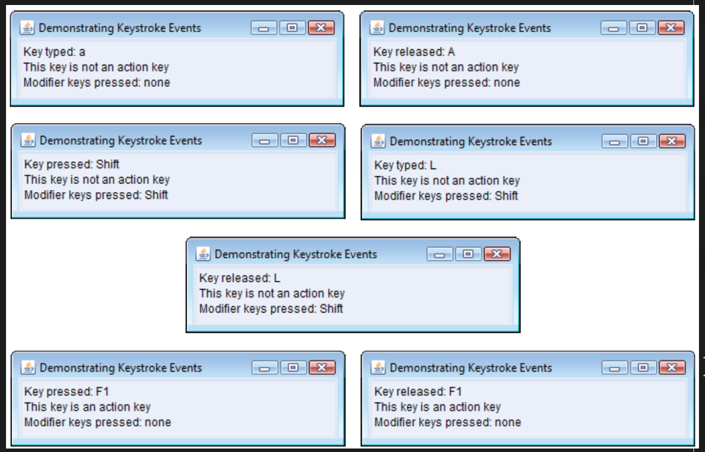
```java
import java.awt.Point;
import java.awt.Graphics;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;
import java.util.ArrayList;
import javax.swing.JPanel;

public class PaintPanel extends JPanel {

  private final ArrayList<Point> points = new ArrayList<>();

  public PaintPanel() {
    addMouseMotionListener(
        new MouseMotionAdapter()  {
          @Override public void mouseDragged(MouseEvent event) {
            points.add(event.getPoint());
            repaint();
          }
        }
    );
  }

  @Override public void paintComponent(Graphics g)  {
    super.paintComponent(g);

    for (Point point : points)
      g.fillOval(point.x, point.y, 4, 4);
  }
}
```



12

```java
import javax.swing.JFrame;

public class KeyDemo {
  public static void main(String[] args) {
    KeyDemoFrame keyDemoFrame = new KeyDemoFrame();
    keyDemoFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    keyDemoFrame.setSize(350, 100);
    keyDemoFrame.setVisible(true);
  }
}
```

```java
import java.awt.Color;
import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;
import javax.swing.JFrame;
import javax.swing.JTextArea;

public class KeyDemoFrame extends JFrame
                   implements KeyListener {
  private String line1 = "";
  private String line2 = "";
  private String line3 = "";
  private JTextArea textArea;

  public KeyDemoFrame()  {
    super("Demonstrating Keystroke Events");

    textArea = new JTextArea(10, 15);
    textArea.setText("Press any key on the keyboard...");
    textArea.setEnabled(false);
    textArea.setDisabledTextColor(Color.BLACK);
    add(textArea);

    addKeyListener(this);
  }

  @Override public void keyPressed(KeyEvent event) {
    line1 = String.format("Key pressed: %s",
     KeyEvent.getKeyText(event.getKeyCode()));
                 setLines2and3(event);
  }

  @Override  public void keyReleased(KeyEvent event)  {
    line1 = String.format("Key released: %s",
               KeyEvent.getKeyText(event.getKeyCode()));
    setLines2and3(event);
  }

  @Override public void keyTyped(KeyEvent event) {
    line1 = String.format("Key typed: %s", event.getKeyChar());
    setLines2and3(event);
  }

  private void setLines2and3(KeyEvent event) {
    line2 = String.format("This key is %san action key",
              (event.isActionKey() ? "" : "not "));

    String temp =
          KeyEvent.getKeyModifiersText(event.getModifiers());

    line3 = String.format("Modifier keys pressed: %s",
                (temp.equals("") ? "none" : temp));

    textArea.setText(String.format("%s\n%s\n%s\n",
               line1, line2, line3));
  }
}
```


Demonstrating Keystroke Events
Key typed: a
This key is not an action key
Modifier keys pressed: none

Demonstrating Keystroke Events
Key released: A
This key is not an action key
Modifier keys pressed: none

Demonstrating Keystroke Events
Key pressed: Shift
This key is not an action key
Modifier keys pressed: Shift

Demonstrating Keystroke Events
Key typed: L
This key is not an action key
Modifier keys pressed: Shift

Demonstrating Keystroke Events
Key released: L
This key is not an action key
Modifier keys pressed: Shift

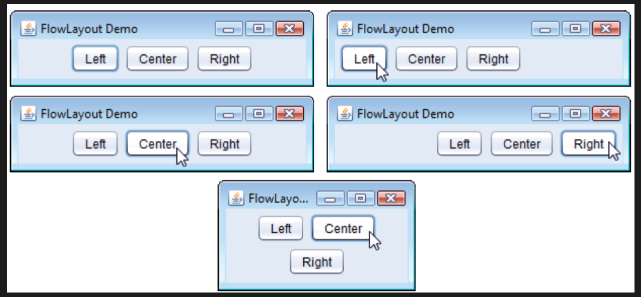
Demonstrating Keystroke Events
Key pressed: F1
This key is an action key
Modifier keys pressed: none

Demonstrating Keystroke Events
Key released: F1
This key is an action key
Modifier keys pressed: none

13

```java
import javax.swing.JFrame;

public class FlowLayoutDemo {
 public static void main(String[] args)
 {
   FlowLayoutFrame flowLayoutFrame = new FlowLayoutFrame();
   flowLayoutFrame.setDefaultCloseOperation(
                            JFrame.EXIT_ON_CLOSE);
   flowLayoutFrame.setSize(300, 75);
   flowLayoutFrame.setVisible(true);
 }
}
```

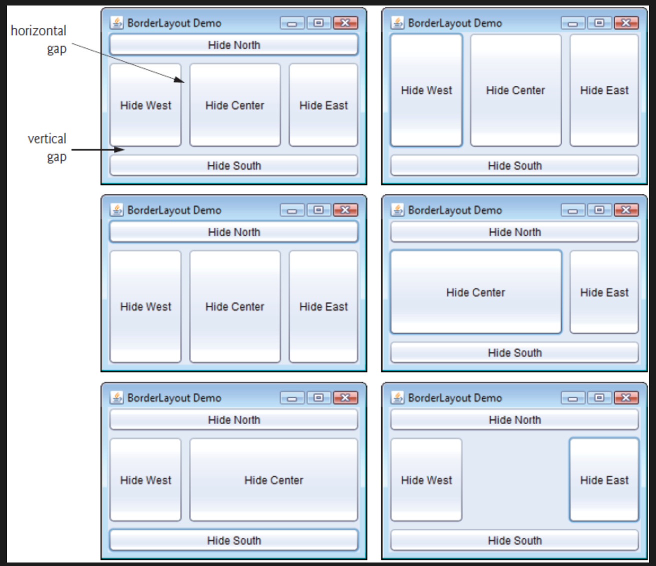
```java
import java.awt.FlowLayout;
import java.awt.Container;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;

public class FlowLayoutFrame extends JFrame
{
 private final JButton leftJButton;
 private final JButton centerJButton;
 private final JButton rightJButton;
 private final FlowLayout layout;
 private final Container container;

 public FlowLayoutFrame() {
  super("FlowLayout Demo");

   layout = new FlowLayout();
   container = getContentPane();
   setLayout(layout);

   leftJButton = new JButton("Left");
   add(leftJButton);
   leftJButton.addActionListener(
    new ActionListener() {
     @Override public void actionPerformed(ActionEvent event) {
      layout.setAlignment(FlowLayout.LEFT);
      layout.layoutContainer(container);
     }
    }
   );

   centerJButton = new JButton("Center");
   add(centerJButton);
   centerJButton.addActionListener(
    new ActionListener() {
     @Override public void actionPerformed(ActionEvent event) {
      layout.setAlignment(FlowLayout.CENTER);
      layout.layoutContainer(container);
     }
    }
   );

   rightJButton = new JButton("Right");
   add(rightJButton);
   rightJButton.addActionListener(
    new ActionListener() {
     @Override public void actionPerformed(ActionEvent event) {
      layout.setAlignment(FlowLayout.RIGHT);
      layout.layoutContainer(container);
     }
    }
   );
 }
}
```



14

```java
import javax.swing.JFrame;

public class BorderLayoutDemo
{
 public static void main(String[] args) {
  BorderLayoutFrame borderLayoutFrame =
           new BorderLayoutFrame();
  borderLayoutFrame.setDefaultCloseOperation(
                   JFrame.EXIT_ON_CLOSE);
  borderLayoutFrame.setSize(300, 200);
  borderLayoutFrame.setVisible(true);
 }
}
```

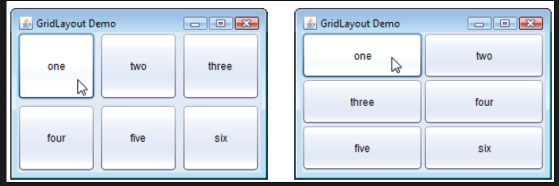
```java
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;

public class BorderLayoutFrame extends JFrame
                       implements ActionListener {
 private final JButton[] buttons;
 private static final String[] names = {
      "Hide North", "Hide South", "Hide East",
      "Hide West", "Hide Center"};
 private final BorderLayout layout;

 public BorderLayoutFrame() {
  super("BorderLayout Demo");

  layout = new BorderLayout(5, 5);
  setLayout(layout);
  buttons = new JButton[names.length];

  for (int count = 0; count < names.length; count++) {
   buttons[count] = new JButton(names[count]);
   buttons[count].addActionListener(this);
  }

  add(buttons[0], BorderLayout.NORTH);
  add(buttons[1], BorderLayout.SOUTH);
  add(buttons[2], BorderLayout.EAST);
  add(buttons[3], BorderLayout.WEST);
  add(buttons[4], BorderLayout.CENTER);
 }

 @Override public void actionPerformed(ActionEvent event) {

  for (JButton button : buttons) {
   if (event.getSource() == button)
    button.setVisible(false);
   else
    button.setVisible(true);
  }

  layout.layoutContainer(getContentPane());
 }
}
```



15

```java
import javax.swing.JFrame;

public class GridLayoutDemo {
 public static void main(String[] args) {
  GridLayoutFrame gridLayoutFrame = new GridLayoutFrame();
  gridLayoutFrame.setDefaultCloseOperation(
                      JFrame.EXIT_ON_CLOSE);
  gridLayoutFrame.setSize(300, 200);
  gridLayoutFrame.setVisible(true);
 }
}
```

```java
import java.awt.GridLayout;
import java.awt.Container;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;

public class GridLayoutFrame extends JFrame
                     implements ActionListener {
 private final JButton[] buttons;
 private static final String[] names =
      {"one", "two", "three", "four", "five", "six"};
 private boolean toggle = true;
 private final Container container;
 private final GridLayout gridLayout1;
 private final GridLayout gridLayout2;

 public GridLayoutFrame() {
  super("GridLayout Demo");

  gridLayout1 = new GridLayout(2, 3, 5, 5);
  gridLayout2 = new GridLayout(3, 2);

  container = getContentPane();
  setLayout(gridLayout1);
  buttons = new JButton[names.length];

  for (int count = 0; count < names.length; count++) {
   buttons[count] = new JButton(names[count]);
   buttons[count].addActionListener(this);
   add(buttons[count]);
  }
 }

 @Override public void actionPerformed(ActionEvent event) {
  if (toggle)
   container.setLayout(gridLayout2);
  else
   container.setLayout(gridLayout1);

  toggle = !toggle;
  container.validate();
 }
}
```

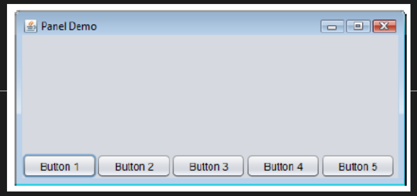


17

```java
import javax.swing.JFrame;

public class TextAreaDemo
{
 public static void main(String[] args)
 {
  TextAreaFrame textAreaFrame = new TextAreaFrame();

textAreaFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  textAreaFrame.setSize(425, 200);
  textAreaFrame.setVisible(true);
 }
}
```

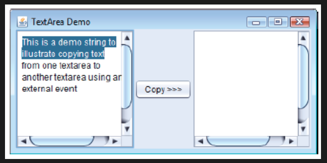
16

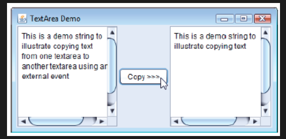
```java
import javax.swing.JFrame;

public class PanelDemo extends JFrame
{
 public static void main(String[] args)
 {
  PanelFrame panelFrame = new PanelFrame();
  panelFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  panelFrame.setSize(450, 200);
  panelFrame.setVisible(true);
 }
}
```



```java
import java.awt.GridLayout;
import java.awt.BorderLayout;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;

public class PanelFrame extends JFrame {
 private final JPanel buttonJPanel;
 private final JButton[] buttons;

 public PanelFrame() {
  super("Panel Demo");

  buttons = new JButton[5];
  buttonJPanel = new JPanel();
  buttonJPanel.setLayout(new GridLayout(1, buttons.length));

  for (int count = 0; count < buttons.length; count++) {
   buttons[count] = new JButton("Button " + (count + 1));
   buttonJPanel.add(buttons[count]);
  }

  add(buttonJPanel, BorderLayout.SOUTH);
 }
}
```

```java
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.Box;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JButton;
import javax.swing.JScrollPane;

public class TextAreaFrame extends JFrame {
 private final JTextArea textArea1;
 private final JTextArea textArea2;
 private final JButton copyJButton;

 public TextAreaFrame() {
  super("TextArea Demo");

  Box box = Box.createHorizontalBox();
  String demo = "This is a demo string to\n" +
   "illustrate copying text\nfrom one textarea to \n" +
   "another textarea using an\nexternal event\n";

  textArea1 = new JTextArea(demo, 10, 15);
  box.add(new JScrollPane(textArea1));

  copyJButton = new JButton("Copy >>>");
  box.add(copyJButton);
  copyJButton.addActionListener(
   new ActionListener() {
    @Override public void actionPerformed(ActionEvent event) {
     textArea2.setText(textArea1.getSelectedText());
    }
   }
  );

  textArea2 = new JTextArea(10, 15);
  textArea2.setEditable(false);
  box.add(new JScrollPane(textArea2));

  add(box);
 }
}
```



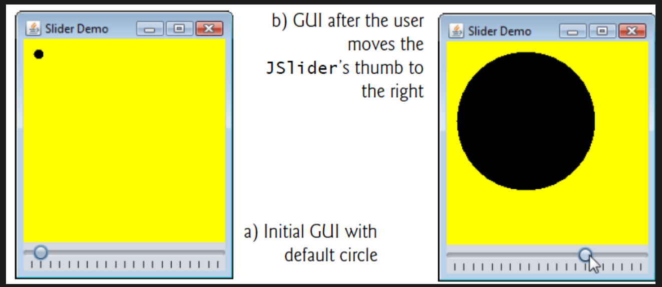


18

```java
import javax.swing.JFrame;

public class SliderDemo {
  public static void main(String[] args) {
    SliderFrame sliderFrame = new SliderFrame();
    sliderFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    sliderFrame.setSize(220, 270);
    sliderFrame.setVisible(true);
  }
}
```

```java
import java.awt.Graphics;
import java.awt.Dimension;
import javax.swing.JPanel;

public class OvalPanel extends JPanel {
  private int diameter = 10;

  @Override  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.fillOval(10, 10, diameter, diameter);
  }

  public void setDiameter(int newDiameter) {
    diameter = (newDiameter >= 0 ? newDiameter : 10);
    repaint();
  }

  public Dimension getPreferredSize() {
    return new Dimension(200, 200);
  }

  public Dimension getMinimumSize() {
    return getPreferredSize();
  }
}
```

```java
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.SwingConstants;
import javax.swing.JSlider;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;

public class SliderFrame extends JFrame {
  private final JSlider diameterJSlider;
  private final OvalPanel myPanel;

  public SliderFrame() {
    super("Slider Demo");
    myPanel = new OvalPanel();
    myPanel.setBackground(Color.YELLOW);

    diameterJSlider =
         new JSlider(SwingConstants.HORIZONTAL, 0, 200, 10);
    diameterJSlider.setMajorTickSpacing(10);
    diameterJSlider.setPaintTicks(true);

    diameterJSlider.addChangeListener(
      new ChangeListener() {
        @Override public void stateChanged(ChangeEvent e) {
          myPanel.setDiameter(diameterJSlider.getValue());
        }
      }
    );

    add(diameterJSlider, BorderLayout.SOUTH);
    add(myPanel, BorderLayout.CENTER);
  }
}
```

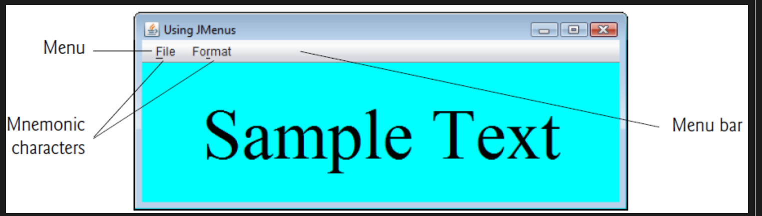


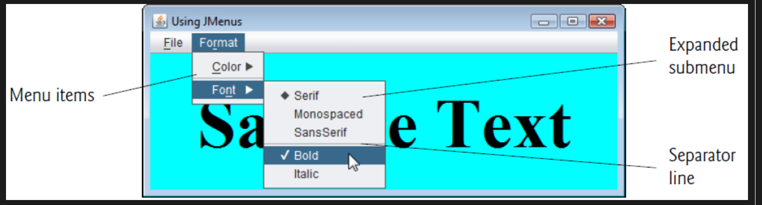
b) GUI after the user moves the `JSlider`'s thumb to the right

a) Initial GUI with default circle

```java
import javax.swing.JFrame;

public class MenuTest {
  public static void main(String[] args) {
    MenuFrame menuFrame = new MenuFrame();
    menuFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    menuFrame.setSize(500, 200);
    menuFrame.setVisible(true);
  }
}
```

```java
import java.awt.Color;
import java.awt.Font;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.ButtonGroup;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JMenuBar;


public class MenuFrame extends JFrame {

  private final Color[] colorValues =
    {Color.BLACK, Color.BLUE, Color.RED, Color.GREEN};

  private final JRadioButtonMenuItem[] colorItems;
  private final JRadioButtonMenuItem[] fonts;
  private final JCheckBoxMenuItem[] styleItems;
  private final JLabel displayJLabel;
  private final ButtonGroup fontButtonGroup;
  private final ButtonGroup colorButtonGroup;
  private int style;


  public MenuFrame() {
    super("Using JMenus");

     // CREAR EL MENÚ DE "File"
    JMenu fileMenu = new JMenu("File");
    fileMenu.setMnemonic('F');

    JMenuItem aboutItem = new JMenuItem("About...");
    aboutItem.setMnemonic('A');
    fileMenu.add(aboutItem);
    aboutItem.addActionListener(
      new ActionListener() {
        @Override public void actionPerformed(ActionEvent event){
          JOptionPane.showMessageDialog(MenuFrame.this,
          "This is an example\nof using menus",
          "About", JOptionPane.PLAIN_MESSAGE);
        }
      }
    );

    JMenuItem exitItem = new JMenuItem("Exit");
    exitItem.setMnemonic('x');
    fileMenu.add(exitItem);
    exitItem.addActionListener(
      new ActionListener() {
        @Override public void actionPerformed(ActionEvent event){
          System.exit(0);
        }
      }
    );

    // CREAR BARRA DE MENÚ y AÑADIR AL JFrame
    JMenuBar bar = new JMenuBar();
    setJMenuBar(bar);

     // AÑADIR MENÚ FILE A BARRA DE MENÚ
    bar.add(fileMenu);

     // CREAR EL MENÚ DE "Format"
    JMenu formatMenu = new JMenu("Format");
    formatMenu.setMnemonic('r');

     // CREAR EL MENÚ TIEM "Format->Color"
    JMenu colorMenu = new JMenu("Color");
    colorMenu.setMnemonic('C');

     String[] colors = { "Black", "Blue", "Red", "Green" };
    colorItems = new JRadioButtonMenuItem[colors.length];
    colorButtonGroup = new ButtonGroup();
    ItemHandler itemHandler = new ItemHandler();

    for (int count = 0; count < colors.length; count++) {
      colorItems[count] =
           new JRadioButtonMenuItem(colors[count]);
      colorMenu.add(colorItems[count]);
      colorButtonGroup.add(colorItems[count]);
      colorItems[count].addActionListener(itemHandler);
    }

    colorItems[0].setSelected(true);

    formatMenu.add(colorMenu);
    formatMenu.addSeparator();

     // CREAR EL MENÚ TIEM "Format->Font"
    JMenu fontMenu = new JMenu("Font");
    fontMenu.setMnemonic('n');

     // CREAR Expanded submenu "Serif, Monospaced, SansSerif"
    String[] fontNames =
       { "Serif", "Monospaced", "SansSerif" };
    fonts = new JRadioButtonMenuItem[fontNames.length];
    fontButtonGroup = new ButtonGroup();

    for (int count = 0; count < fonts.length; count++)
    {
      fonts[count] =
         new JRadioButtonMenuItem(fontNames[count]);
      fontMenu.add(fonts[count]);
      fontButtonGroup.add(fonts[count]);
      fonts[count].addActionListener(itemHandler);
    }

    fonts[0].setSelected(true);
    fontMenu.addSeparator();

     // CREAR Expanded submenu "Bold, Italic"
    String[] styleNames = { "Bold", "Italic" };
    styleItems = new JCheckBoxMenuItem[styleNames.length];
    StyleHandler styleHandler = new StyleHandler();

    for (int count = 0; count < styleNames.length; count++) {
      styleItems[count] =
          new JCheckBoxMenuItem(styleNames[count]);
      fontMenu.add(styleItems[count]);
      styleItems[count].addItemListener(styleHandler);
    }

    formatMenu.add(fontMenu);
    bar.add(formatMenu);

     // CREAR EL contenido principal: texto y fondo
    displayJLabel =
         new JLabel("Sample Text", SwingConstants.CENTER);
    displayJLabel.setForeground(colorValues[0]);
    displayJLabel.setFont(new Font("Serif", Font.PLAIN, 72));

    getContentPane().setBackground(Color.CYAN);
    add(displayJLabel, BorderLayout.CENTER);

  } // END  MenuFrame


  private class ItemHandler implements ActionListener  {
    @Override public void actionPerformed(ActionEvent event) {
      for (int count = 0; count < colorItems.length; count++) {
        if (colorItems[count].isSelected()) {
          displayJLabel.setForeground(colorValues[count]);
          break;
        }
      }


      for (int count = 0; count < fonts.length; count++) {
        if (event.getSource() == fonts[count]) {
          displayJLabel.setFont(
                new Font(fonts[count].getText(), style, 72));
        }
      }

      repaint();
    }
  }



  private class StyleHandler implements ItemListener {
    @Override public void itemStateChanged(ItemEvent e) {
      String name = displayJLabel.getFont().getName();
      Font font;

      if (styleItems[0].isSelected() &&
          styleItems[1].isSelected() )
         font = new Font(name, Font.BOLD + Font.ITALIC, 72);
      else if (styleItems[0].isSelected())
         font = new Font(name, Font.BOLD, 72);
      else if (styleItems[1].isSelected())
         font = new Font(name, Font.ITALIC, 72);
      else
         font = new Font(name, Font.PLAIN, 72);

      displayJLabel.setFont(font);

      repaint();
    }
  }


}
```

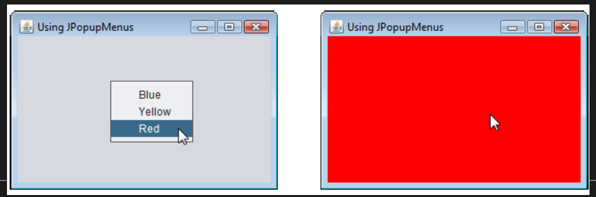




20

```
import javax.swing.JFrame;

public class PopupTest {
  public static void main(String[] args)  {
    PopupFrame popupFrame = new PopupFrame();
    popupFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    popupFrame.setSize(300, 200);
    popupFrame.setVisible(true);
  }
}
```

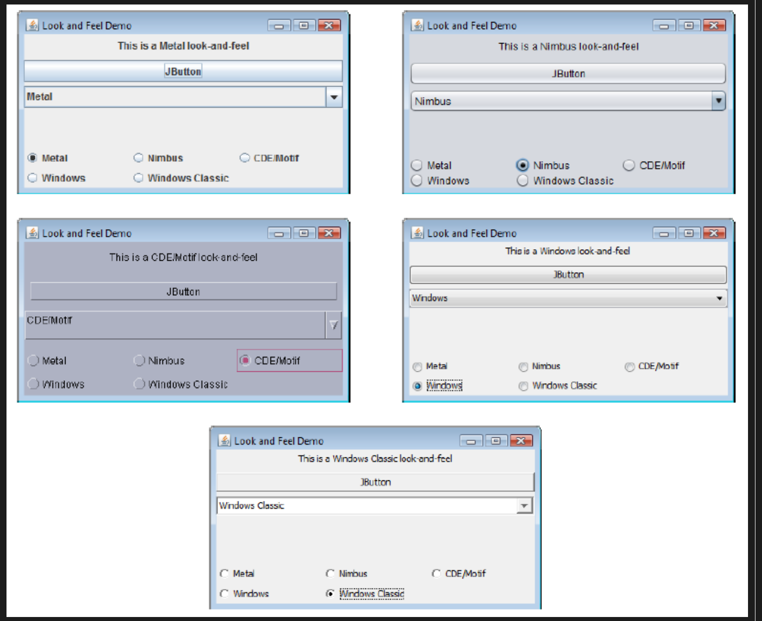
```
import java.awt.Color;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.JPopupMenu;
import javax.swing.ButtonGroup;

public class PopupFrame extends JFrame {
  private final JRadioButtonMenuItem[] items;
  private final Color[] colorValues =
    { Color.BLUE, Color.YELLOW, Color.RED };
  private final JPopupMenu popupMenu;

  public PopupFrame() {
    super("Using JPopupMenus");

    ItemHandler handler     = new ItemHandler();
    String[] colors         = { "Blue", "Yellow", "Red" };
    ButtonGroup colorGroup = new ButtonGroup();
    popupMenu               = new JPopupMenu();
    items         = new JRadioButtonMenuItem[colors.length];

    for (int count = 0; count < items.length; count++) {
      items[count] = new JRadioButtonMenuItem(colors[count]);
      popupMenu.add(items[count]);
      colorGroup.add(items[count]);
      items[count].addActionListener(handler);
    }

    setBackground(Color.WHITE);

    addMouseListener(
      new MouseAdapter() {
        @Override public void mousePressed(MouseEvent event) {
          checkForTriggerEvent(event);
        }

        @Override public void mouseReleased(MouseEvent event) {
          checkForTriggerEvent(event);
        }

        private void checkForTriggerEvent(MouseEvent event) {
          if (event.isPopupTrigger())
            popupMenu.show(event.getComponent(),
                           event.getX(), event.getY());
        }
      }
    );
  }

  private class ItemHandler implements ActionListener {
    @Override public void actionPerformed(ActionEvent event) {
      for (int i = 0; i < items.length; i++) {
        if (event.getSource() == items[i]) {
          getContentPane().setBackground(colorValues[i]);
          return;
        }
      }
    }
  }

}
```



21

```
import javax.swing.JFrame;

public class LookAndFeelDemo {
  public static void main(String[] args) {
    LookAndFeelFrame lookAndFeelFrame =
         new LookAndFeelFrame();

    lookAndFeelFrame.setDefaultCloseOperation(
            JFrame.EXIT_ON_CLOSE);
    lookAndFeelFrame.setSize(400, 220);
    lookAndFeelFrame.setVisible(true);
  }
}
```

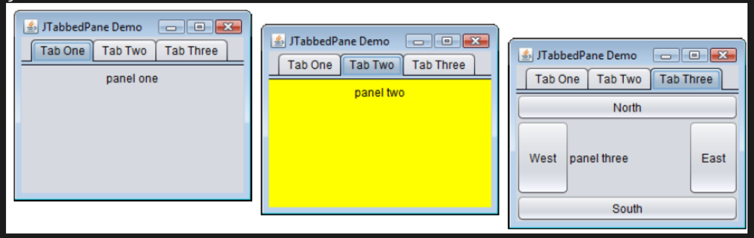
```
import java.awt.GridLayout;
import java.awt.BorderLayout;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
import javax.swing.JFrame;
import javax.swing.UIManager;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JComboBox;
import javax.swing.JPanel;
import javax.swing.SwingConstants;
import javax.swing.SwingUtilities;

public class LookAndFeelFrame extends JFrame  {
  private final UIManager.LookAndFeelInfo[] looks;
  private final String[] lookNames;
  private final JRadioButton[] radio;
  private final ButtonGroup group;
  private final JButton button;
  private final JLabel label;
  private final JComboBox<String> comboBox;

  public LookAndFeelFrame() {
    super("Look and Feel Demo");

    looks = UIManager.getInstalledLookAndFeels();
    lookNames = new String[looks.length];

    for (int i = 0; i < looks.length; i++)
    lookNames[i] = looks[i].getName();
    JPanel northPanel = new JPanel();
    northPanel.setLayout(new GridLayout(3, 1, 0, 5));
    label = new JLabel("This is a " +
                       lookNames[0] +
                      " look-and-feel",
                      SwingConstants.CENTER);
    northPanel.add(label);

    button = new JButton("JButton");
    northPanel.add(button);

    comboBox = new JComboBox<String>(lookNames);
    northPanel.add(comboBox);

    radio = new JRadioButton[looks.length];
    JPanel southPanel = new JPanel();

    int rows = (int) Math.ceil(radio.length / 3.0);
    southPanel.setLayout(new GridLayout(rows, 3));

    group = new ButtonGroup();
    ItemHandler handler = new ItemHandler();

    for (int count = 0; count < radio.length; count++) {
      radio[count] = new JRadioButton(lookNames[count]);
      radio[count].addItemListener(handler);
      group.add(radio[count]);
      southPanel.add(radio[count]);
    }

    add(northPanel, BorderLayout.NORTH);
    add(southPanel, BorderLayout.SOUTH);

    radio[0].setSelected(true);
 }

  private void changeTheLookAndFeel(int value) {
    try {
      UIManager.setLookAndFeel(looks[value].getClassName());
      SwingUtilities.updateComponentTreeUI(this);
    } catch (Exception exception) {
      exception.printStackTrace();
    }
  }

  private class ItemHandler implements ItemListener  {
    @Override public void itemStateChanged(ItemEvent event) {
      for (int count = 0; count < radio.length; count++) {
        if (radio[count].isSelected()) {
          label.setText(String.format(
           "This is a %s look-and-feel", lookNames[count]));
          comboBox.setSelectedIndex(count);
          changeTheLookAndFeel(count);
        }
      }
    }
  }

}
```


Look and Feel Demo
This is a Metal look-and-feel
JButton
Metal
Metal
Nimbus
CDE/Motif
Windows
Windows Classic
This is a Nimbus look-and-feel
This is a CDE/Motif look-and-feel
This is a Windows look-and-feel
This is a Windows Classic look-and-feel

23

```
import javax.swing.JFrame;

public class JTabbedPaneDemo {
  public static void main(String[] args)  {
    JTabbedPaneFrame tabbedPaneFrame = new JTabbedPaneFrame();

tabbedPaneFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    tabbedPaneFrame.setSize(250, 200);
    tabbedPaneFrame.setVisible(true);
  }
}
```

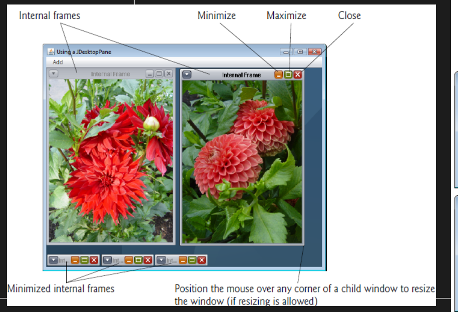
```
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JFrame;
import javax.swing.JTabbedPane;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.SwingConstants;

public class JTabbedPaneFrame extends JFrame  {

  public JTabbedPaneFrame()  {
    super("JTabbedPane Demo ");

    JTabbedPane tabbedPane = new JTabbedPane();

    JLabel label1 =
        new JLabel("panel one", SwingConstants.CENTER);
    JPanel panel1 = new JPanel();
    panel1.add(label1);
    tabbedPane.addTab("Tab One", null, panel1, "First Panel");

    JLabel label2 =
        new JLabel("panel two", SwingConstants.CENTER);
    JPanel panel2 = new JPanel();
    panel2.setBackground(Color.YELLOW);
    panel2.add(label2);
    tabbedPane.addTab("Tab Two", null, panel2, "Second Panel");

    JLabel label3 = new JLabel("panel three");
    JPanel panel3 = new JPanel();
    panel3.setLayout(new BorderLayout());
    panel3.add(new JButton("North"), BorderLayout.NORTH);
    panel3.add(new JButton("West"), BorderLayout.WEST);
    panel3.add(new JButton("East"), BorderLayout.EAST);
    panel3.add(new JButton("South"), BorderLayout.SOUTH);
    panel3.add(label3, BorderLayout.CENTER);
    tabbedPane.addTab("Tab Three",
                 null, panel3, "Third Panel");
    add(tabbedPane);
  }
}
```



22

```java
import javax.swing.JFrame;

public class DesktopTest {
  public static void main(String[] args) {
    DesktopFrame desktopFrame = new DesktopFrame();

    desktopFrame.setDefaultCloseOperation(
          JFrame.EXIT_ON_CLOSE);
    desktopFrame.setSize(600, 480);
    desktopFrame.setVisible(true);
  }
}
```

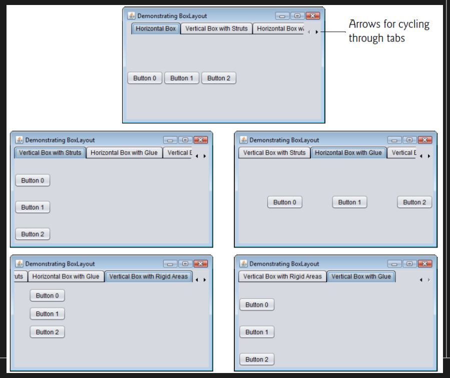
```java
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.security.SecureRandom;
import javax.swing.JFrame;
import javax.swing.JDesktopPane;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JInternalFrame;
import javax.swing.JPanel;
import javax.swing.ImageIcon;

public class DesktopFrame extends JFrame {
  private final JDesktopPane theDesktop;

  public DesktopFrame() {
    super("Using a JDesktopPane");

    JMenuBar bar = new JMenuBar();
    JMenu addMenu = new JMenu("Add");

    JMenuItem newFrame = new JMenuItem("Internal Frame");
    addMenu.add(newFrame);
    bar.add(addMenu);
    setJMenuBar(bar);

    theDesktop = new JDesktopPane();
    add(theDesktop);

    newFrame.addActionListener(
      new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent event)  {
          JInternalFrame frame = new JInternalFrame(
           "Internal Frame", true, true, true, true);
          MyJPanel panel = new MyJPanel();
         frame.add(panel, BorderLayout.CENTER);
         frame.pack();
         theDesktop.add(frame);
        frame.setVisible(true);
      } } );
  }
}

class MyJPanel extends JPanel {
  private static final SecureRandom generator =
      new SecureRandom();
  private final ImageIcon picture;

  private final static String[] images = {"flor1.png",
      "flor2.png", "flor3.png", "flor4.png", "flor5.png" };

  public MyJPanel() {
    int randomNumber = generator.nextInt(images.length);
    picture = new ImageIcon(images[randomNumber]);
  }

  @Override public void paintComponent(Graphics g) {
    super.paintComponent(g);
    picture.paintIcon(this, g, 0, 0);
  }

  public Dimension getPreferredSize() {
    return new Dimension(picture.getIconWidth(),
    picture.getIconHeight());  }
}
```


Internal frames
Minimize
Maximize
Close
Using a JDesktopPane
Add
Internal Frame
Internal Frame
Minimized internal frames
Position the mouse over any corner of a child window to resize the window (if resizing is allowed)

24

```java
import javax.swing.JFrame;
public class BoxLayoutDemo {
  public static void main(String[] args)  {
    BoxLayoutFrame boxLayoutFrame = new BoxLayoutFrame();

   boxLayoutFrame.setDefaultCloseOperation(
              JFrame.EXIT_ON_CLOSE);
    boxLayoutFrame.setSize(400, 220);
    boxLayoutFrame.setVisible(true);
  }
}
```

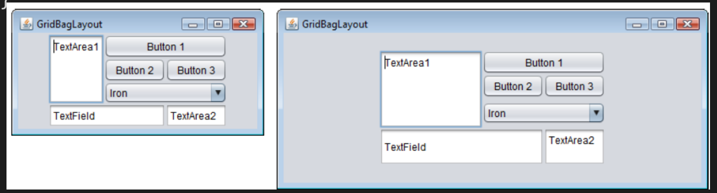
```java
import java.awt.Dimension;
import javax.swing.JFrame;
import javax.swing.Box;
import javax.swing.JButton;
import javax.swing.BoxLayout;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;

public class BoxLayoutFrame extends JFrame {

  public BoxLayoutFrame() {
    super("Demonstrating BoxLayout");

    Box horizontal1 = Box.createHorizontalBox();
    Box vertical1 = Box.createVerticalBox();
    Box horizontal2 = Box.createHorizontalBox();
    Box vertical2 = Box.createVerticalBox();

    final int SIZE = 3;

    for (int count = 0; count < SIZE; count++)
      horizontal1.add(new JButton("Button " + count));

    for (int count = 0; count < SIZE; count++) {
      vertical1.add(Box.createVerticalStrut(25));
      vertical1.add(new JButton("Button " + count));
    }

    for (int count = 0; count < SIZE; count++) {
      horizontal2.add(Box.createHorizontalGlue());
      horizontal2.add(new JButton("Button " + count));
    }

    for (int count = 0; count < SIZE; count++) {
      vertical2.add(Box.createRigidArea(new Dimension(12, 8)));
      vertical2.add(new JButton("Button " + count));
    }

    JPanel panel = new JPanel();
    panel.setLayout (new BoxLayout(panel, BoxLayout.Y_AXIS));

    for (int count = 0; count < SIZE; count++) {
      panel.add(Box.createGlue());
      panel.add(new JButton("Button " + count));
    }

    JTabbedPane tabs = new JTabbedPane(
      JTabbedPane.TOP, JTabbedPane.SCROLL_TAB_LAYOUT);

    tabs.addTab("Horizontal Box", horizontal1);
    tabs.addTab("Vertical Box with Struts", vertical1);
    tabs.addTab("Horizontal Box with Glue", horizontal2);
    tabs.addTab("Vertical Box with Rigid Areas", vertical2);
    tabs.addTab("Vertical Box with Glue", panel);
    add(tabs);
  }
}
```



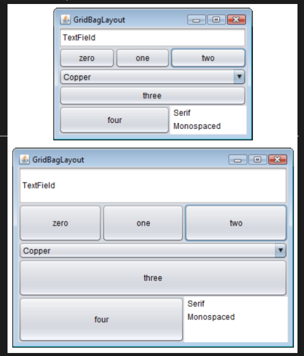
25

```java
import javax.swing.JFrame;

public class GridBagDemo {
  public static void main(String[] args)  {
    GridBagFrame gridBagFrame = new GridBagFrame();
    gridBagFrame.setDefaultCloseOperation(
          JFrame.EXIT_ON_CLOSE);
    gridBagFrame.setSize(300, 150);
    gridBagFrame.setVisible(true);
  }
}
```

```java
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.Component;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JComboBox;

public class GridBagFrame extends JFrame {
  private final GridBagLayout layout;
  private final GridBagConstraints constraints;

  public GridBagFrame() {
    super("GridBagLayout");

    layout = new GridBagLayout();
    setLayout(layout);

    constraints = new GridBagConstraints();
    JTextArea textArea1 = new JTextArea("TextArea1", 5, 10);
    JTextArea textArea2 = new JTextArea("TextArea2", 2, 2);

    String[] names = { "Iron", "Steel", "Brass" };
    JComboBox<String> comboBox = new JComboBox<String>(names);

    JTextField textField = new JTextField("TextField");
    JButton button1 = new JButton("Button 1");
    JButton button2 = new JButton("Button 2");
    JButton button3 = new JButton("Button 3");

    constraints.fill = GridBagConstraints.BOTH;
    addComponent(textArea1, 0, 0, 1, 3);

    constraints.fill = GridBagConstraints.HORIZONTAL;
    addComponent(button1, 0, 1, 2, 1);
    addComponent(comboBox, 2, 1, 2, 1);

    constraints.weightx = 1000;
    constraints.weighty = 1;
    constraints.fill = GridBagConstraints.BOTH;
    addComponent(button2, 1, 1, 1, 1);

    constraints.weightx = 0;
    constraints.weighty = 0;
    addComponent(button3, 1, 2, 1, 1);

    addComponent(textField, 3, 0, 2, 1);
    addComponent(textArea2, 3, 2, 1, 1);
  }

  private void addComponent(
      Component component,  int row, int column, int width,
      int height)  {
    constraints.gridx      = column;
    constraints.gridy      = row;
    constraints.gridwidth  = width;
    constraints.gridheight = height;

    layout.setConstraints(component, constraints);
    add(component);
  }
}
```



```java
import javax.swing.JFrame;

public class GridBagDemo2 {
  public static void main(String[] args) {
    GridBagFrame2 gridBagFrame = new GridBagFrame2();

    gridBagFrame.setDefaultCloseOperation(
          JFrame.EXIT_ON_CLOSE);
    gridBagFrame.setSize(300, 200);
    gridBagFrame.setVisible(true);
  }
}
```



```java
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.Component;
import javax.swing.JFrame;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import javax.swing.JList;
import javax.swing.JButton;

public class GridBagFrame2 extends JFrame {
  private final GridBagLayout layout;
  private final GridBagConstraints constraints;

  public GridBagFrame2() {
    super("GridBagLayout");
    layout = new GridBagLayout();
    setLayout(layout);
    constraints = new GridBagConstraints();

    String[] metals = { "Copper", "Aluminum", "Silver" };
    JComboBox comboBox = new JComboBox(metals);

    JTextField textField = new JTextField("TextField");

    String[] fonts = { "Serif", "Monospaced" };
    JList list = new JList(fonts);

    String[] names = { "zero", "one", "two", "three", "four" };
    JButton[] buttons = new JButton[names.length];

    for (int count = 0; count < buttons.length; count++)
      buttons[count] = new JButton(names[count]);

    constraints.weightx = 1;
    constraints.weighty = 1;
    constraints.fill = GridBagConstraints.BOTH;
    constraints.gridwidth = GridBagConstraints.REMAINDER;
    addComponent(textField);

    constraints.gridwidth = 1;
    addComponent(buttons[0]);

    constraints.gridwidth = GridBagConstraints.RELATIVE;
    addComponent(buttons[1]);

    constraints.gridwidth = GridBagConstraints.REMAINDER;
    addComponent(buttons[2]);

    constraints.weighty = 0;
    constraints.gridwidth = GridBagConstraints.REMAINDER;
    addComponent(comboBox);

    constraints.weighty = 1;
    constraints.gridwidth = GridBagConstraints.REMAINDER;
    addComponent(buttons[3]);

    constraints.gridwidth = GridBagConstraints.RELATIVE;
    addComponent(buttons[4]);

    constraints.gridwidth = GridBagConstraints.REMAINDER;
    addComponent(list);
  }

  private void addComponent(Component component)
  {
    layout.setConstraints(component, constraints);
    add(component);
  }
}
```