MyFrameWithComponents.java

import javax.swing.\*;

public class MyFrameWithComponents {

public static void main(String[] args) {

JFrame frame = new JFrame("MyFrameWithComponents");

// Add a button into the frame

JButton jbtOK = new JButton("OK");

frame.add(jbtOK);

frame.setSize(400, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLocationRelativeTo(null); // Center the frame

frame.setVisible(true);

}

}

FlowLayout :

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JFrame;

import java.awt.FlowLayout;

public class ShowFlowLayout extends JFrame {

public ShowFlowLayout() {

// Set FlowLayout, aligned left with horizontal gap 10

// and vertical gap 20 between components

setLayout(**new** FlowLayout(FlowLayout.LEFT, **10**, **20**) );

// Add labels and text fields to the frame

add(new JLabel("First Name"));

add(new JTextField(8));

add(new JLabel("MI"));

add(new JTextField(1));

add(new JLabel("Last Name"));

add(new JTextField(8));

}

/\*\* Main method \*/

public static void main(String[] args) {

ShowFlowLayout frame = new ShowFlowLayout();

frame.setTitle("ShowFlowLayout");

frame.setSize(200, 200);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

GridLayout :

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JFrame;

import java.awt.GridLayout;

public class ShowGridLayout extends JFrame {

public ShowGridLayout() {

// Set GridLayout, 3 rows, 2 columns, and gaps 5 between

// components horizontally and vertically

setLayout(new GridLayout(3, 2, 5, 5));

// Add labels and text fields to the frame

add(new JLabel("First Name"));

add(new JTextField(8));

add(new JLabel("MI"));

add(new JTextField(1));

add(new JLabel("Last Name"));

add(new JTextField(8));

}

/\*\* Main method \*/

public static void main(String[] args) {

ShowGridLayout frame = new ShowGridLayout();

frame.setTitle("ShowGridLayout");

frame.setSize(200, 125);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

BorderLayout :

import javax.swing.JButton;

import javax.swing.JFrame;

import java.awt.BorderLayout;

public class ShowBorderLayout extends JFrame{

public ShowBorderLayout() {

// Set BorderLayout with horizontal gap 5 and vertical gap 10

setLayout(new BorderLayout(5, 10) );

// Add buttons to the frame

add(new JButton("East"), BorderLayout.EAST);

add(new JButton("South"), BorderLayout.SOUTH);

add(new JButton("West"), BorderLayout.WEST);

add(new JButton("North"), BorderLayout.NORTH);

add(new JButton("Center"), BorderLayout.CENTER);

}

/\*\* Main method \*/

public static void main(String[] args) {

ShowBorderLayout frame = new ShowBorderLayout();

frame.setTitle("ShowBorderLayout");

frame.setSize(300, 200);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

OVEN :

import java.awt.\*;

import javax.swing.\*;

public class TestPanels extends JFrame {

public TestPanels() {

// Create panel p1 for the buttons and set GridLayout

JPanel p1 = new JPanel();

p1.setLayout(new GridLayout(4, 3));

// Add buttons to the panel

for (int i = 1; i <= 9; i++) {

p1.add (new JButton("" + i));

}

p1.add(new JButton("" + 0));

p1.add(new JButton("Start"));

p1.add(new JButton("Stop"));

// Create panel p2 to hold a text field and p1

JPanel p2 = new JPanel(new BorderLayout());

p2.add (new JTextField("Time to be displayed here"),

BorderLayout.NORTH);

p2.add (p1, BorderLayout.CENTER);

// add contents into the frame

add(p2, BorderLayout.EAST);

add(new JButton("Food to be placed here"),

BorderLayout.CENTER);

}

/\*\* Main method \*/

public static void main(String[] args) {

TestPanels frame = new TestPanels();

frame.setTitle("The Front View of a Microwave Oven");

frame.setSize(400, 250);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

The font class :

public Font(String name, int style, int size);

Font font1 = new Font("SansSerif", Font.BOLD, 16);

Font font2 = new Font("Serif", Font.BOLD + Font.ITALIC, 12);

JButton jbtOK = new JButton("OK");

jbtOK.setFont(font1);

TestSwingCommonFeatures :

import java.awt.\*;

import javax.swing.\*;

import javax.swing.border.\*;

public class TestSwingCommonFeatures extends JFrame {

public TestSwingCommonFeatures() {

// Create a panel to group three buttons

JPanel p1 = new JPanel(new FlowLayout(FlowLayout.LEFT, 2, 2));

JButton jbtLeft = new JButton("Left");

JButton jbtCenter = new JButton("Center");

JButton jbtRight = new JButton("Right");

jbtLeft.setBackground(Color.WHITE);

jbtRight.setToolTipText("This is the Right button");

jbtCenter.setForeground(Color.GREEN);

p1.add(jbtLeft);

p1.add(jbtCenter);

p1.add(jbtRight);

p1.setBorder(new TitledBorder("Three Buttons"));

// Create a font and a line border

Font largeFont = new Font("TimesRoman", Font.BOLD, 20);

Border lineBorder = new LineBorder(Color.BLACK, 2);

// Create a panel to group two labels

JPanel p2 = new JPanel(new GridLayout(1, 2, 5, 5));

JLabel jlblRed = new JLabel("Red");

JLabel jlblOrange = new JLabel("Orange");

jlblRed.setForeground(Color.RED);

jlblOrange.setForeground(Color.ORANGE);

jlblRed.setFont(largeFont);

jlblOrange.setFont(largeFont);

jlblRed.setBorder(lineBorder);

jlblOrange.setBorder(lineBorder);

p2.add(jlblRed);

p2.add(jlblOrange);

p2.setBorder(new TitledBorder("Two Labels"));

// Add two panels to the frame

setLayout(new GridLayout(2, 1, 5, 5));

add(p1);

add(p2);

}

public static void main(String[] args) {

// Create a frame and set its properties

JFrame frame = new TestSwingCommonFeatures();

frame.setTitle("TestSwingCommonFeatures");

frame.setSize(300, 150);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

TestImageIcon :

import javax.swing.\*;

import java.awt.\*;

public class TestImageIcon extends JFrame {

private ImageIcon usIcon = new ImageIcon("image/us.gif");

private ImageIcon myIcon = new ImageIcon("image/my.jpg");

private ImageIcon frIcon = new ImageIcon("image/fr.gif");

private ImageIcon ukIcon = new ImageIcon("image/uk.gif");

public TestImageIcon() {

setLayout(new GridLayout(1, 4, 5, 5));

add(new JLabel(usIcon));

add(new JLabel(myIcon));

add(new JButton(frIcon));

add(new JButton(ukIcon));

}

/\*\* Main method \*/

public static void main(String[] args) {

TestImageIcon frame = new TestImageIcon();

frame.setTitle("TestImageIcon");

frame.setSize(200, 200);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

TestFigurePanel :

import java.awt.\*;

import javax.swing.\*;

public class TestFigurePanel extends JFrame {

public TestFigurePanel() {

setLayout(new GridLayout(2, 3, 5, 5));

add(new FigurePanel(FigurePanel.LINE));

add(new FigurePanel(FigurePanel.RECTANGLE));

add(new FigurePanel(FigurePanel.ROUND\_RECTANGLE));

add(new FigurePanel(FigurePanel.OVAL));

add(new FigurePanel(FigurePanel.RECTANGLE, true));

add(new FigurePanel(FigurePanel.ROUND\_RECTANGLE, true));

}

public static void main(String[] args) {

TestFigurePanel frame = new TestFigurePanel();

frame.setSize(400, 200);

frame.setTitle("TestFigurePanel");

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

DetectSourceDemo :

import javax.swing.\*;

import java.awt.event.\*;

public class DetectSourceDemo extends JFrame {

// Create four buttons

private JButton jbtNew = new JButton("New");

private JButton jbtOpen = new JButton("Open");

private JButton jbtSave = new JButton("Save");

private JButton jbtPrint = new JButton("Print");

public DetectSourceDemo() {

// Create a panel to hold buttons

JPanel panel = new JPanel();

panel.add(jbtNew);

panel.add(jbtOpen);

panel.add(jbtSave);

panel.add(jbtPrint);

add(panel);

// Create a listener

ButtonListener listener = new ButtonListener();

// Register listener with buttons

jbtNew.addActionListener(listener);

jbtOpen.addActionListener(listener);

jbtSave.addActionListener(listener);

jbtPrint.addActionListener(listener);

}

class ButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

if (e.getSource() == jbtNew)

System.out.println("Process New");

else if (e.getSource() == jbtOpen)

System.out.println("Process Open");

else if (e.getSource() == jbtSave)

System.out.println("Process Save");

else if (e.getSource() == jbtPrint)

System.out.println("Process Print");

}

}

/\*\* Main method \*/

public static void main(String[] args) {

JFrame frame = new DetectSourceDemo();

frame.setTitle("DetectSourceDemo");

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

LoanCalculator :

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import javax.swing.border.TitledBorder;

public class LoanCalculator extends JFrame {

// Create text fields for interest rate,

// year, loan amount, monthly payment, and total payment

private JTextField jtfAnnualInterestRate = new JTextField();

private JTextField jtfNumberOfYears = new JTextField();

private JTextField jtfLoanAmount = new JTextField();

private JTextField jtfMonthlyPayment = new JTextField();

private JTextField jtfTotalPayment = new JTextField();

// Create a Compute Payment button

private JButton jbtComputeLoan = new JButton("Compute Payment");

public LoanCalculator() {

// Panel p1 to hold labels and text fields

JPanel p1 = new JPanel(new GridLayout(5, 2));

p1.add(new JLabel("Annual Interest Rate"));

p1.add(jtfAnnualInterestRate);

p1.add(new JLabel("Number of Years"));

p1.add(jtfNumberOfYears);

p1.add(new JLabel("Loan Amount"));

p1.add(jtfLoanAmount);

p1.add(new JLabel("Monthly Payment"));

p1.add(jtfMonthlyPayment);

p1.add(new JLabel("Total Payment"));

p1.add(jtfTotalPayment);

p1.setBorder(new

TitledBorder("Enter loan amount, interest rate, and year"));

// Panel p2 to hold the button

JPanel p2 = new JPanel(new FlowLayout(FlowLayout.RIGHT));

p2.add(jbtComputeLoan);

// Add the panels to the frame

add(p1, BorderLayout.CENTER);

add(p2, BorderLayout.SOUTH);

// Register listener

jbtComputeLoan.addActionListener(new ButtonListener());

}

/\*\* Handle the Compute Payment button \*/

private class ButtonListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

// Get values from text fields

double interest =

Double.parseDouble(jtfAnnualInterestRate.getText() );

int year =

Integer.parseInt(jtfNumberOfYears.getText());

double loanAmount =

Double.parseDouble(jtfLoanAmount.getText());

// Create a loan object

Loan loan = new Loan(interest, year, loanAmount);

// Display monthly payment and total payment

jtfMonthlyPayment.setText(String.format("%.2f",

loan.getMonthlyPayment()));

jtfTotalPayment.setText(String.format("%.2f",

loan.getTotalPayment()));

}

}

public static void main(String[] args) {

LoanCalculator frame = new LoanCalculator();

frame.pack();

frame.setTitle("LoanCalculator");

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

TestWindowEvent :

import java.awt.event.\*;

import javax.swing.JFrame;

public class TestWindowEvent extends JFrame {

public static void main(String[] args) {

TestWindowEvent frame = new TestWindowEvent();

frame.setSize(220, 80);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setTitle("TestWindowEvent");

frame.setVisible(true);

}

public TestWindowEvent() {

addWindowListener(new WindowListener() {

/\*\*

\* Handler for window-deiconified event

Invoked when a window is changed from a minimized

\* to a normal state.

\*/

public void windowDeiconified(WindowEvent event) {

System.out.println("Window deiconified");

}

/\*\*

\* Handler for window-iconified event

\* Invoked when a window is changed from a normal to a

\* minimized state. For many platforms, a minimized window

\* is displayed as the icon specified in the window's

\* iconImage property.

\*/

public void windowIconified(WindowEvent event) {

System.out.println("Window iconified");

}

/\*\*

\* Handler for window-activated event

\* Invoked when the window is set to be the user's

\* active window, which means the window (or one of its

\* subcomponents) will receive keyboard events.

\*/

public void windowActivated(WindowEvent event) {

System.out.println("Window activated");

}

/\*\*

\* Handler for window-deactivated event

\* Invoked when a window is no longer the user's active

\* window, which means that keyboard events will no longer

\* be delivered to the window or its subcomponents.

\*/

public void windowDeactivated(WindowEvent event) {

System.out.println("Window deactivated");

}

/\*\*

\* Handler for window-opened event

\* Invoked the first time a window is made visible.

\*/

public void windowOpened(WindowEvent event) {

System.out.println("Window opened");

}

/\*\*

\* Handler for window-closing event

\* Invoked when the user attempts to close the window

\* from the window's system menu. If the program does not

\* explicitly hide or dispose the window while processing

\* this event, the window-closing operation will be cancelled.

\*/

public void windowClosing(WindowEvent event) {

System.out.println("Window closing");

}

/\*\*

\* Handler for window-closed event

\* Invoked when a window has been closed as the result

\* of calling dispose on the window.

\*/

public void windowClosed(WindowEvent event) {

System.out.println("Window closed");

}

);

}

}

AdapterDemo :

import java.awt.event.\*;

import javax.swing.JFrame;

public class AdapterDemo extends JFrame {

public static void main(String[] args) {

AdapterDemo frame = new AdapterDemo();

frame.setSize(220, 80);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setTitle("AdapterDemo");

frame.setVisible(true);

}

public AdapterDemo() {

addWindowListener(new WindowAdapter() {

public void windowActivated(WindowEvent event) {

System.out.println("Window activated");

}

});

}

}

MoveMessageDemo :

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class MoveMessageDemo extends JFrame {

public MoveMessageDemo() {

// Create a MovableMessagePanel instance for moving a message

MovableMessagePanel p = new MovableMessagePanel

("Welcome to Java");

// Place the message panel in the frame

setLayout(new BorderLayout());

add(p);

}

/\*\* Main method \*/

public static void main(String[] args) {

MoveMessageDemo frame = new MoveMessageDemo();

frame.setTitle("MoveMessageDemo");

frame.setSize(200, 100);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

// Inner class: MovableMessagePanel draws a message

static class MovableMessagePanel extends JPanel {

private String message = "Welcome to Java";

private int x = 20;

private int y = 20;

/\*\* Construct a panel to draw string s \*/

public MovableMessagePanel(String s) {

message = s;

addMouseMotionListener(new MouseMotionAdapter() {

/\*\* Handle mouse-dragged event \*/

public void mouseDragged(MouseEvent e) {

// Get the new location and repaint the screen

x = e.getX();

y = e.getY();

repaint();

}

});

}

/\*\* Paint the component \*/

protected void paintComponent(Graphics g) {

super.paintComponent(g);

g.drawString(message, x, y);

}

}

}

KeyEventDemo :

1 import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class KeyEventDemo extends JFrame {

private KeyboardPanel keyboardPanel = new KeyboardPanel();

/\*\* Initialize UI \*/

public KeyEventDemo() {

// Add the keyboard panel to accept and display user input

add(keyboardPanel);

// Set focus

keyboardPanel.setFocusable(true);

}

/\*\* Main method \*/

public static void main(String[] args) {

KeyEventDemo frame = new KeyEventDemo();

frame.setTitle("KeyEventDemo");

frame.setSize(300, 300);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

// Inner class: KeyboardPanel for receiving key input

static class KeyboardPanel extends JPanel {

private int x = 100;

private int y = 100;

private char keyChar = 'A'; // Default key

public KeyboardPanel() {

addKeyListener(new KeyAdapter() {

public void keyPressed(KeyEvent e) {

switch (e.getKeyCode()) {

case KeyEvent.VK\_DOWN: y += 10; break;

case KeyEvent.VK\_UP: y -= 10; break;

case KeyEvent.VK\_LEFT: x -= 10; break;

case KeyEvent.VK\_RIGHT: x += 10; break;

default: keyChar = e.getKeyChar();

}

repaint();

}

});

}

/\*\* Draw the character \*/

protected void paintComponent(Graphics g) {

super.paintComponent(g);

g.setFont(new Font("TimesRoman", Font.PLAIN, 24));

g.drawString(String.valueOf(keyChar), x, y);

}

}

}

TestButtonIcons :

import javax.swing.\*;

public class TestButtonIcons extends JFrame {

public static void main(String[] args) {

// Create a frame and set its properties

JFrame frame = new TestButtonIcons();

frame.setTitle("ButtonIcons");

frame.setSize(200, 100);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

public TestButtonIcons() {

ImageIcon usIcon = new ImageIcon("image/usIcon.gif");

ImageIcon caIcon = new ImageIcon("image/caIcon.gif");

ImageIcon ukIcon = new ImageIcon("image/ukIcon.gif");

JButton jbt = new JButton("Click it", usIcon);

jbt.setPressedIcon(caIcon);

jbt.setRolloverIcon(ukIcon);

add(jbt);

}

}

ButtonDemo :

1 import java.awt.\*;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.\*;

5

public class ButtonDemo extends JFrame {

// Create a panel for displaying message

protected MessagePanel messagePanel

= new MessagePanel("Welcome to Java");

10

// Declare two buttons to move the message left and right

private JButton jbtLeft = new JButton("<=");

private JButton jbtRight = new JButton("=>");

14

public static void main(String[] args) {

ButtonDemo frame = new ButtonDemo();

frame.setTitle("ButtonDemo");

frame.setSize(250, 100);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

23

public ButtonDemo() {

// Set the background color of messagePanel

messagePanel.setBackground(Color.white);

27

// Create Panel jpButtons to hold two Buttons "<=” and “right =>"

JPanel jpButtons = new JPanel();

jpButtons.add(jbtLeft);

jpButtons.add(jbtRight);4 32

// Set keyboard mnemonics

jbtLeft.setMnemonic('L');

jbtRight.setMnemonic('R');

36

// Set icons and remove text

// jbtLeft.setIcon(new ImageIcon("image/left.gif"));

// jbtRight.setIcon(new ImageIcon("image/right.gif"));

// jbtLeft.setText(null);

// jbtRight.setText(null);

42

43 // Set tool tip text on the buttons

44 jbtLeft.setToolTipText("Move message to left");

45 jbtRight.setToolTipText("Move message to right");

46

47 // Place panels in the frame

48 setLayout(new BorderLayout());

49 add(messagePanel, BorderLayout.CENTER);

50 add(jpButtons, BorderLayout.SOUTH);

51

52 // Register listeners with the buttons

53 jbtLeft.addActionListener(new ActionListener() {

54 public void actionPerformed(ActionEvent e) {

55 messagePanel.moveLeft();

56 }

57 });

58 jbtRight.addActionListener(new ActionListener() {

59 public void actionPerformed(ActionEvent e) {

60 messagePanel.moveRight();

61 }

62 });

63 }

64 }

CheckBoxDemo :

1 import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

4

public class CheckBoxDemo extends ButtonDemo {

// Create three check boxes to control the display of message

private JCheckBox jchkCentered = new JCheckBox("Centered");

private JCheckBox jchkBold = new JCheckBox("Bold");

private JCheckBox jchkItalic = new JCheckBox("Italic");

10

public static void main(String[] args) {

CheckBoxDemo frame = new CheckBoxDemo();

frame.setTitle("CheckBoxDemo");

frame.setSize(500, 200);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

19

public CheckBoxDemo() {

// Set mnemonic keys

jchkCentered.setMnemonic('C');

jchkBold.setMnemonic('B');

jchkItalic.setMnemonic('I');

25

// Create a new panel to hold check boxes

JPanel jpCheckBoxes = new JPanel();

jpCheckBoxes.setLayout(new GridLayout(3, 1));

jpCheckBoxes.add(jchkCentered);

jpCheckBoxes.add(jchkBold);

jpCheckBoxes.add(jchkItalic);

add(jpCheckBoxes, BorderLayout.EAST);

33

// Register listeners with the check boxes

jchkCentered.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

messagePanel.setCentered(jchkCentered.isSelected());

}

});

jchkBold.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

setNewFont();

}

});

jchkItalic.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

setNewFont();

}

});

}

51

private void setNewFont() {

// Determine a font style

int fontStyle = Font.PLAIN;

fontStyle += (jchkBold.isSelected() ? Font.BOLD : Font.PLAIN);

fontStyle += (jchkItalic.isSelected() ? Font.ITALIC : Font.PLAIN);

57

// Set font for the message

Font font = messagePanel.getFont();

messagePanel.setFont(

new Font(font.getName(), fontStyle, font.getSize()));

}

}

RadioButtonDemo :

import java.awt.\*;

2 import java.awt.event.\*;

import javax.swing.\*;

4

public class RadioButtonDemo extends CheckBoxDemo {

// Declare radio buttons

private JRadioButton jrbRed, jrbGreen, jrbBlue;

8

public static void main(String[] args) {

RadioButtonDemo frame = new RadioButtonDemo();

frame.setSize(500, 200);

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setTitle("RadioButtonDemo");

frame.setVisible(true);

}

17

public RadioButtonDemo() {

// Create a new panel to hold check boxes

JPanel jpRadioButtons = new JPanel();

jpRadioButtons.setLayout(new GridLayout(3, 1));

jpRadioButtons.add(jrbRed = new JRadioButton("Red") );

jpRadioButtons.add(jrbGreen = new JRadioButton("Green"));

jpRadioButtons.add(jrbBlue = new JRadioButton("Blue"));

add(jpRadioButtons, BorderLayout.WEST);

26

// Create a radio-button group to group three buttons

ButtonGroup group = new ButtonGroup();

group.add(jrbRed);

group.add(jrbGreen);

group.add(jrbBlue);

32

// Set keyboard mnemonics

jrbRed.setMnemonic('E');

jrbGreen.setMnemonic('G');

jrbBlue.setMnemonic('U');

37

// Register listeners for radio buttons

jrbRed.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

messagePanel.setForeground(Color.red);

}

});

jrbGreen.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

messagePanel.setForeground(Color.green);

}

});

jrbBlue.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

messagePanel.setForeground(Color.blue);

}

});

54

// Set initial message color to blue

jrbBlue.setSelected(true);

messagePanel.setForeground(Color.blue);

}

}

Labels :

// Create an image icon from an image file

ImageIcon icon = new ImageIcon("image/grapes.gif");

// Create a label with a text, an icon,

// with centered horizontal alignment

JLabel jlbl = new JLabel("Grapes", icon, SwingConstants.CENTER);

//Set label's text alignment and gap between text and icon

jlbl.setHorizontalTextPosition(SwingConstants.CENTER);

jlbl.setVerticalTextPosition(SwingConstants.BOTTOM);

jlbl.setIconTextGap(5);

TextFieldDemo :

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

4

public class TextFieldDemo extends RadioButtonDemo {

private JTextField jtfMessage = new JTextField(10);

7

/\*\* Main method \*/

public static void main(String[] args) {

TextFieldDemo frame = new TextFieldDemo();

frame.pack();

frame.setTitle("TextFieldDemo");

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

} 22 jpTextField.add(

23 new JLabel("Enter a new message"), BorderLayout.WEST);

24 jpTextField.add(jtfMessage, BorderLayout.CENTER);

25 add(jpTextField, BorderLayout.NORTH);

26

27 jtfMessage.setHorizontalAlignment(JTextField.RIGHT);

28

29 // Register listener

30 jtfMessage.addActionListener(new ActionListener() {

31 /\*\* Handle ActionEvent \*/

32 public void actionPerformed(ActionEvent e) {

33 messagePanel.setMessage(jtfMessage.getText());

34 jtfMessage.requestFocusInWindow();

35 }

36 });

37 }

38 }

DescriptionPanel :

1 import javax.swing.\*;

import java.awt.\*;

3

public class DescriptionPanel extends JPanel {

/\*\* Label for displaying an image icon and a text \*/

private JLabel jlblImageTitle = new JLabel();

7

/\*\* Text area for displaying text \*/

private JTextArea jtaDescription = new JTextArea();

10

public DescriptionPanel() {

// Center the icon and text and place the text under the icon

jlblImageTitle.setHorizontalAlignment(JLabel.CENTER);

jlblImageTitle.setHorizontalTextPosition(JLabel.CENTER);

jlblImageTitle.setVerticalTextPosition(JLabel.BOTTOM);

16

// Set the font in the label and the text field

jlblImageTitle.setFont(new Font("SansSerif", Font.BOLD, 16));

jtaDescription.setFont(new Font("Serif", Font.PLAIN, 14));

20

// Set lineWrap and wrapStyleWord true for the text area

jtaDescription.setLineWrap(true);

jtaDescription.setWrapStyleWord(true);

jtaDescription.setEditable(false);

25

// Create a scroll pane to hold the text area

JScrollPane scrollPane = new JScrollPane(jtaDescription);

28

// Set BorderLayout for the panel, add label and scrollpane

setLayout(new BorderLayout(5, 5));

add(scrollPane, BorderLayout.CENTER);

add(jlblImageTitle, BorderLayout.WEST);

}

34

/\*\* Set the title \*/

public void setTitle(String title) {

jlblImageTitle.setText(title);

}

39

/\*\* Set the image icon \*/

public void setImageIcon(ImageIcon icon) {

jlblImageTitle.setIcon(icon);

}

44

/\*\* Set the text description \*/

public void setDescription(String text) {

jtaDescription.setText(text);

}

}

TextAreaDemo :

1 import java.awt.\*;

import javax.swing.\*;

3

public class TextAreaDemo extends JFrame {

// Declare and create a description panel

private DescriptionPanel descriptionPanel = new DescriptionPanel();

7

public static void main(String[] args) {

TextAreaDemo frame = new TextAreaDemo();

frame.pack();

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setTitle("TextAreaDemo");

frame.setVisible(true);

}

16

public TextAreaDemo() {

// Set title, text and image in the description panel

descriptionPanel.setTitle("Canada");

String description = "The Maple Leaf flag \n\n" +

"The Canadian National Flag was adopted by the Canadian " +

"Parliament on October 22, 1964 and was proclaimed into law " +

"by Her Majesty Queen Elizabeth II (the Queen of Canada) on " +

"February 15, 1965. The Canadian Flag (colloquially known " +

"as The Maple Leaf Flag) is a red flag of the proportions " +

"two by length and one by width, containing in its center a " +

"white square, with a single red stylized eleven-point " +

"maple leaf centered in the white square.";

descriptionPanel.setDescription(description);

descriptionPanel.setImageIcon(new ImageIcon("image/ca.gif"));

31

// Add the description panel to the frame

setLayout(new BorderLayout());

add(descriptionPanel, BorderLayout.CENTER);

}

}

ComboBoxDemo :

1 import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

4

public class ComboBoxDemo extends JFrame {

// Declare an array of Strings for flag titles

private String[] flagTitles = {"Canada", "China", "Denmark",

"France", "Germany", "India", "Norway", "United Kingdom",

"United States of America"};

10

// Declare an ImageIcon array for the national flags of 9 countries

private ImageIcon[] flagImage = {

new ImageIcon("image/ca.gif"),

new ImageIcon("image/china.gif"),

new ImageIcon("image/denmark.gif"),

new ImageIcon("image/fr.gif"),

new ImageIcon("image/germany.gif"),

new ImageIcon("image/india.gif"),

new ImageIcon("image/norway.gif"),

new ImageIcon("image/uk.gif"),

new ImageIcon("image/us.gif")

};

23

// Declare an array of strings for flag descriptions

private String[] flagDescription = new String[9];

26

// Declare and create a description panel

private DescriptionPanel descriptionPanel = new DescriptionPanel();

29

// Create a combo box for selecting countries

private JComboBox jcbo = new JComboBox(flagTitles);

32

public static void main(String[] args) {

ComboBoxDemo frame = new ComboBoxDemo();

frame.pack();

frame.setTitle("ComboBoxDemo");

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

41

public ComboBoxDemo() {

// Set text description

flagDescription[0] = "The Maple Leaf flag \n\n" +

"The Canadian National Flag was adopted by the Canadian " +

"Parliament on October 22, 1964 and was proclaimed into law " +

"by Her Majesty Queen Elizabeth II (the Queen of Canada) on " +

"February 15, 1965. The Canadian Flag (colloquially known " +

"as The Maple Leaf Flag) is a red flag of the proportions " +

"two by length and one by width, containing in its center a " +

"white square, with a single red stylized eleven-point " +

"maple leaf centered in the white square.";

flagDescription[1] = "Description for China ... ";

flagDescription[2] = "Description for Denmark ... ";

flagDescription[3] = "Description for France ... ";

flagDescription[4] = "Description for Germany ... ";

flagDescription[5] = "Description for India ... ";

flagDescription[6] = "Description for Norway ... ";

flagDescription[7] = "Description for UK ... ";

flagDescription[8] = "Description for US ... ";

61

// Set the first country (Canada) for display

setDisplay(0);

64

// Add combo box and description panel to the list

add(jcbo, BorderLayout.NORTH);

add(descriptionPanel, BorderLayout.CENTER);

68

// Register listener

jcbo.addItemListener(new ItemListener() {

/\*\* Handle item selection \*/

public void itemStateChanged(ItemEvent e) {

setDisplay(jcbo.getSelectedIndex());

}

});

}

77

/\*\* Set display information on the description panel \*/

public void setDisplay(int index) {

descriptionPanel.setTitle(flagTitles[index]);

descriptionPanel.setImageIcon(flagImage[index]);

descriptionPanel.setDescription(flagDescription[index]);

}

}

ListDemo :

1 import java.awt.\*;

import javax.swing.\*;

import javax.swing.event.\*;

4

public class ListDemo extends JFrame {

final int NUMBER\_OF\_FLAGS = 9;

7

// Declare an array of Strings for flag titles

private String[] flagTitles = {"Canada", "China", "Denmark",

"France", "Germany", "India", "Norway", "United Kingdom",

"United States of America"};

12

// The list for selecting countries

private JList jlst = new JList(flagTitles);

15

// Declare an ImageIcon array for the national flags of 9 countries

private ImageIcon[] imageIcons = {

new ImageIcon("image/ca.gif"),

new ImageIcon("image/china.gif"),

new ImageIcon("image/denmark.gif"),

new ImageIcon("image/fr.gif"),

new ImageIcon("image/germany.gif"),

new ImageIcon("image/india.gif"),

new ImageIcon("image/norway.gif"),

new ImageIcon("image/uk.gif"),

new ImageIcon("image/us.gif")

};

28

// Arrays of labels for displaying images

private JLabel[] jlblImageViewer = new JLabel[NUMBER\_OF\_FLAGS];

31

public static void main(String[] args) {

ListDemo frame = new ListDemo();

frame.setSize(650, 500);

frame.setTitle("ListDemo");

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

40

public ListDemo() {

// Create a panel to hold nine labels

JPanel p = new JPanel(new GridLayout(3, 3, 5, 5));

44

for (int i = 0; i < NUMBER\_OF\_FLAGS; i++) {

p.add(jlblImageViewer[i] = new JLabel());

jlblImageViewer[i].setHorizontalAlignment

(SwingConstants.CENTER);

}

50

// Add p and the list to the frame

add(p, BorderLayout.CENTER);

add(new JScrollPane(jlst), BorderLayout.WEST);

54

// Register listeners

jlst.addListSelectionListener(new ListSelectionListener() {

/\*\* Handle list selection \*/

public void valueChanged(ListSelectionEvent e) {

// Get selected indices

int[] indices = jlst.getSelectedIndices();

61

int i;

// Set icons in the labels

for (i = 0; i < indices.length; i++) {

jlblImageViewer[i].setIcon(imageIcons[indices[i]]);

}

67

// Remove icons from the rest of the labels

for (; i < NUMBER\_OF\_FLAGS; i++) {

jlblImageViewer[i].setIcon(null);

}

}

});

}

}

ScrollBarDemo :

1 import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

4

public class ScrollBarDemo extends JFrame {

// Create horizontal and vertical scroll bars

private JScrollBar jscbHort =

new JScrollBar(JScrollBar.HORIZONTAL);

private JScrollBar jscbVert =

new JScrollBar(JScrollBar.VERTICAL);

11

// Create a MessagePanel

private MessagePanel messagePanel =

new MessagePanel("Welcome to Java");

15

public static void main(String[] args) {

ScrollBarDemo frame = new ScrollBarDemo();

frame.setTitle("ScrollBarDemo");

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

}

24

public ScrollBarDemo() {

// Add scroll bars and message panel to the frame

setLayout(new BorderLayout());

add(messagePanel, BorderLayout.CENTER);

add(jscbVert, BorderLayout.EAST);

add(jscbHort, BorderLayout.SOUTH);

31

// Register listener for the scroll bars

jscbHort.addAdjustmentListener(new AdjustmentListener() {

public void adjustmentValueChanged(AdjustmentEvent e) {

// getValue() and getMaximumValue() return int, but for better

// precision, use double

double value = jscbHort.getValue();

double maximumValue = jscbHort.getMaximum();

double newX = (value \* messagePanel.getWidth() /

maximumValue);

messagePanel.setXCoordinate((int)newX);

}

});

jscbVert.addAdjustmentListener(new AdjustmentListener() {

public void adjustmentValueChanged(AdjustmentEvent e) {

// getValue() and getMaximumValue() return int, but for better

// precision, use double

double value = jscbVert.getValue();

double maximumValue = jscbVert.getMaximum();

double newY = (value \* messagePanel.getHeight() /

maximumValue);

messagePanel.setYCoordinate((int)newY);

}

});

}

}

SliderDemo :

1 import java.awt.\*;

import javax.swing.\*;

import javax.swing.event.\*;

4

public class SliderDemo extends JFrame {

// Create horizontal and vertical sliders

private JSlider jsldHort = new JSlider(JSlider.HORIZONTAL);

private JSlider jsldVert = new JSlider(JSlider.VERTICAL);

9

// Create a MessagePanel

private MessagePanel messagePanel =

new MessagePanel("Welcome to Java");

13

public static void main(String[] args) {

SliderDemo frame = new SliderDemo();

frame.setTitle("SliderDemo");

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

}

22

public SliderDemo() {

// Add sliders and message panel to the frame

setLayout(new BorderLayout(5, 5));

add(messagePanel, BorderLayout.CENTER);

add(jsldVert, BorderLayout.EAST);

add(jsldHort, BorderLayout.SOUTH);

29

// Set properties for sliders

jsldHort.setMaximum(50);

jsldHort.setPaintLabels(true);

jsldHort.setPaintTicks(true);

jsldHort.setMajorTickSpacing(10);

jsldHort.setMinorTickSpacing(1);

jsldHort.setPaintTrack(false);

jsldVert.setInverted(true);

jsldVert.setMaximum(10);

jsldVert.setPaintLabels(true);

jsldVert.setPaintTicks(true);

jsldVert.setMajorTickSpacing(10);

jsldVert.setMinorTickSpacing(1);

43

// Register listener for the sliders

jsldHort.addChangeListener(new ChangeListener() {

/\*\* Handle scroll-bar adjustment actions \*/

public void stateChanged(ChangeEvent e) {

// getValue() and getMaximumValue() return int, but for better

// precision, use double

double value = jsldHort.getValue();

double maximumValue = jsldHort.getMaximum();

double newX = (value \* messagePanel.getWidth() /

maximumValue);

messagePanel.setXCoordinate((int)newX);

}

});

jsldVert.addChangeListener(new ChangeListener() {

/\*\* Handle scroll-bar adjustment actions \*/

public void stateChanged(ChangeEvent e) {

// getValue() and getMaximumValue() return int, but for better

// precision, use double

double value = jsldVert.getValue();

double maximumValue = jsldVert.getMaximum();

double newY = (value \* messagePanel.getHeight() /

maximumValue);

messagePanel.setYCoordinate((int) newY);

}

});

}

}

MultipleWindowsDemo :

1 import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

4

public class MultipleWindowsDemo extends JFrame {

private JTextArea jta;

private JButton jbtShowHistogram = new JButton("Show Histogram");

private Histogram histogram = new Histogram();

9

// Create a new frame to hold the histogram panel

private JFrame histogramFrame = new JFrame();

12

public MultipleWindowsDemo() {

// Store text area in a scroll pane

JScrollPane scrollPane = new JScrollPane(jta = new JTextArea());

scrollPane.setPreferredSize(new Dimension(300, 200));

jta.setWrapStyleWord(true);

jta.setLineWrap(true);

19

// Place scroll pane and button in the frame

add(scrollPane, BorderLayout.CENTER);

add(jbtShowHistogram, BorderLayout.SOUTH);

23

// Register listener

jbtShowHistogram.addActionListener(new ActionListener() {

/\*\* Handle the button action \*/

public void actionPerformed(ActionEvent e) {

// Count the letters in the text area

int[] count = countLetters();

30

// Set the letter count to histogram for display

histogram.showHistogram(count);

33

// Show the frame

histogramFrame.setVisible(true);

}

});

38

// Create a new frame to hold the histogram panel

histogramFrame.add(histogram);

histogramFrame.pack();

histogramFrame.setTitle("Histogram");

}

44

/\*\* Count the letters in the text area \*/

private int[] countLetters() {

// Count for 26 letters

int[] count = new int[26];

49

// Get contents from the text area

String text = jta.getText();

52

// Count occurrences of each letter (case insensitive)

for (int i = 0; i < text.length(); i++) {

char character = text.charAt(i);

56

if ((character >= 'A') && (character <= 'Z')) {

count[character - 'A']++;

}

else if ((character >= 'a') && (character <= 'z')) {

count[character - 'a']++;

}

}

64

return count; // Return the count array

}

67

public static void main(String[] args) {

MultipleWindowsDemo frame = new MultipleWindowsDemo();

frame.setLocationRelativeTo(null); // Center the frame

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setTitle("MultipleWindowsDemo");

frame.pack();

frame.setVisible(true);

}

}

Histogram :

1 import javax.swing.\*;

import java.awt.\*;

3

public class Histogram extends JPanel {

// Count the occurrences of 26 letters

private int[] count;

7

/\*\* Set the count and display histogram \*/

public void showHistogram(int[] count) {

this.count = count;

repaint();

}

13

/\*\* Paint the histogram \*/

protected void paintComponent(Graphics g) {

if (count == null) return; // No display if count is null

17

super.paintComponent(g);

19

// Find the panel size and bar width and interval dynamically

int width = getWidth();

int height = getHeight();

int interval = (width - 40) / count.length;

int individualWidth = (int)(((width - 40) / 24) \* 0.60);

25

// Find the maximum count. The maximum count has the highest bar

int maxCount = 0;

for (int i = 0; i < count.length; i++) {

if (maxCount < count[i])

maxCount = count[i];

}

32

// x is the start position for the first bar in the histogram

int x = 30;

35

// Draw a horizontal base line

g.drawLine(10, height - 45, width - 10, height - 45);

for (int i = 0; i < count.length; i++) {

// Find the bar height

int barHeight =

(int)(((double)count[i] / (double)maxCount) \* (height - 55));

42

// Display a bar (i.e. rectangle)

g.drawRect(x, height - 45 - barHeight, individualWidth,

barHeight);

46

// Display a letter under the base line

g.drawString((char)(65 + i) + "", x, height - 30);

49

// Move x for displaying the next character

x += interval;

}

}

54

/\*\* Override getPreferredSize \*/

public Dimension getPreferredSize() {

return new Dimension(300, 300);

}

}