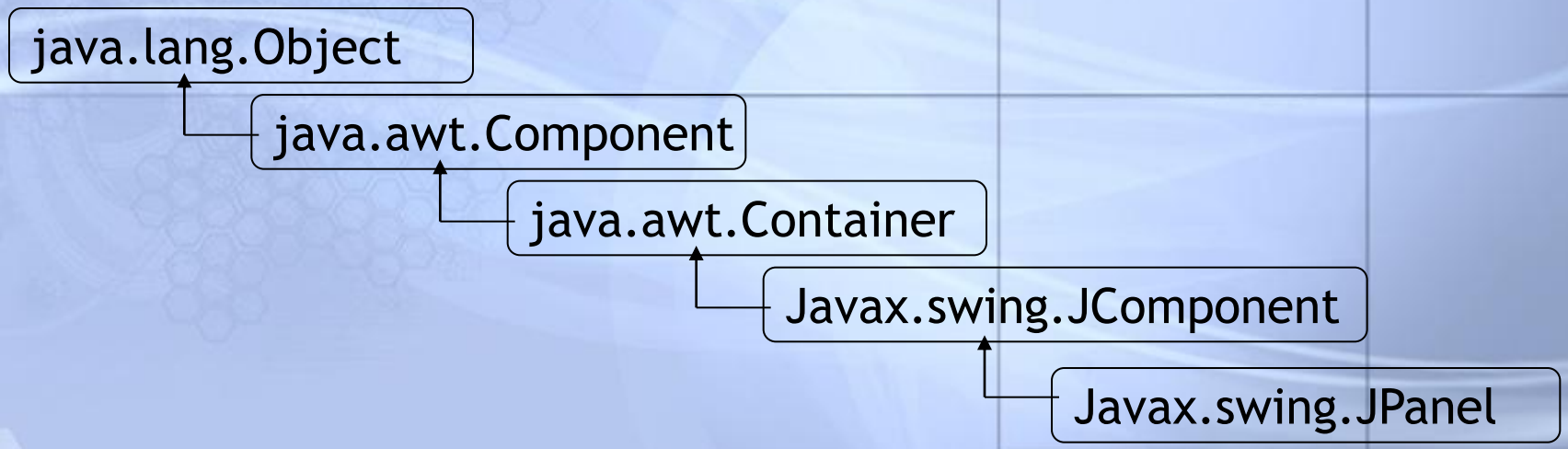
Session 10

Graphical User Interface (GUI) (Cont'd)

JPanel

- To group/organize components
- A custom component that requires embedded components
- JPanel Class is the all purpose container class of Swing
- Inheritance diagram of JPanel is



Example : A simple JPanel

```
import javax.swing.*;
import java.awt.*;
class panelTest extends JPanel
{
  panelTest()
    {
      setBackground(Color.white);
    }
  public void paintComponent(Graphics g)
    {
      super.paintComponent(g);
      g.drawString("Hello from Swing!", 50, 30);
      g.drawString("There is a problem here!", 70, 120);
      g.drawString("This visible form is Testing!", 35, 150);
    }
}
```

Example

```
import java.awt.*;
import javax.swing.*;
import.java.awt.event.*;
public class jpanel extends JFrame
{ panelTest J;
   public jpanel()
     {   super("Swing Application");
         Container contentpane=getcontentpane();
         J=new panelTest();
         contentpane.add(J);
      }
  public static void main (String args[])
     {    final JFrame f= new jpanel();
          f.setBounds (100, 100, 300, 300);
          f.setVisible(true);
      }
}
```

Output

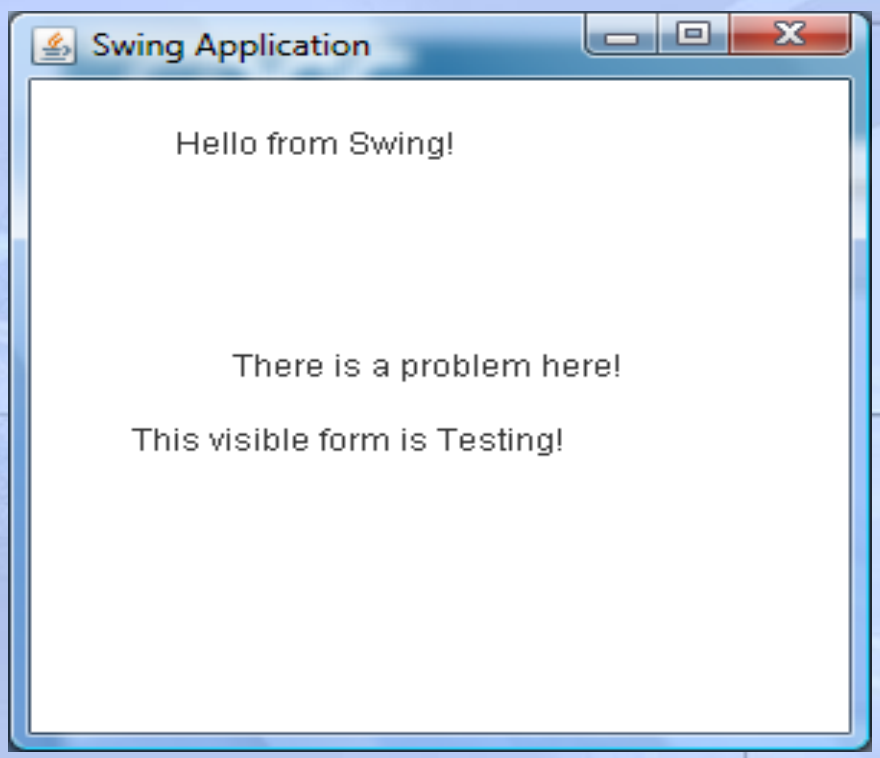


Figure : Output Frame of the previous program

JScrollPane

In Swing, a text area does not have scroll bars. If you want scroll bars, you have to insert the text area inside a *scroll pane*. Then insert the scroll pane inside the content pane.

Usage:

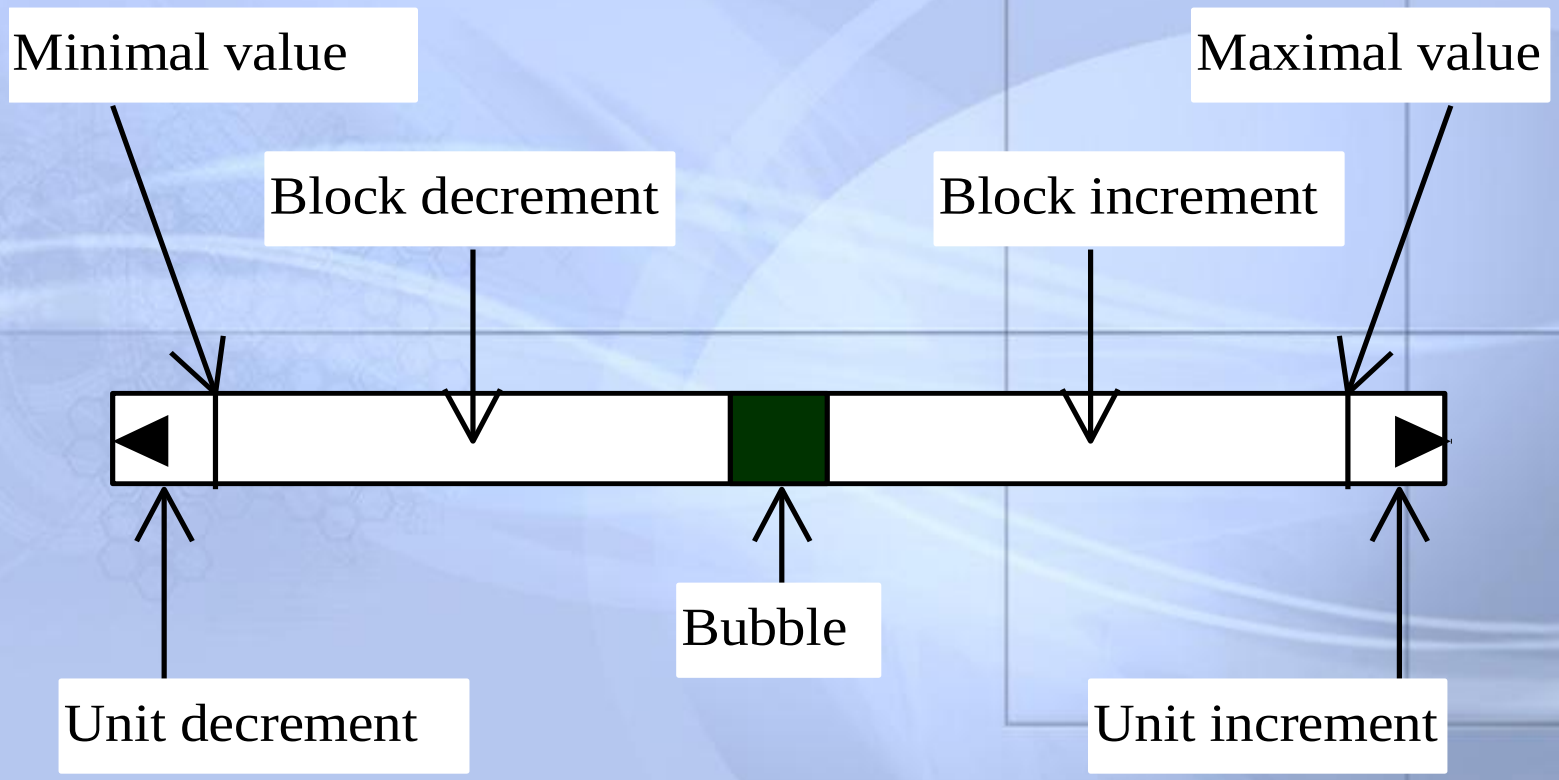
Javax.swing.JScrollPane

JScrollPane(component c)

Parameters: c The component to scroll

JScrollPane

- Properties



Example

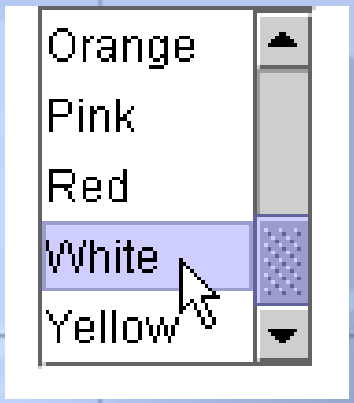
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class textscroll extends JFrame implements ActionListener,KeyListener
{
  …….
  JTextArea area=new JTextArea(20,70);
  ……..

  public textscroll()
  {
    ……..
    ……..
    JScrollPane jsp = new JScrollPane(area);
    panel.add(jsp,BorderLayout.CENTER);
    …….
    setContentPane(panel);
    setVisible(true);
  }
```

Example

```
public void actionPerformed(ActionEvent e)
{
  …….
}
public void keyPressed(KeyEvent key)
{
 ……..
}
public void keyTyped(KeyEvent key)
{
}
public void keyReleased(KeyEvent key)
{
}
public static void main(String[] args)
{
  new textscroll();
}
}
```

JList

- **List**
  - Displays series of items, may select one or more
  - This section, discuss single-selection lists



- Class **JList**
  - Constructor **JList(arrayOfNames)**
    - Takes array of **Objects** (**String**s) to display in list
  - **setVisibleRowCount(n)**
    - Displays **n** items at a time
    - Does not provide automatic scrolling

JList

- **JScrollPane** object used for scrolling
  c.add(new JScrollPane(colorList));
  - Takes component to which to add scrolling as argument
  - Add **JScrollPane** object to content pane
- **JList** methods
  - **setSelectionMode(selection_CONSTANT)**
  - **SINGLE_SELECTION**
    - One item selected at a time
  - **SINGLE_INTERVAL_SELECTION**
    - Multiple selection list, allows contiguous items to be selected
  - **MULTIPLE_INTERVAL_SELECTION**
    - Multiple-selection list, any items can be selected

JList

- **JList** methods
  - **getSelectedIndex()**
    - Returns index of selected item

- **Event handlers**
  - Implement interface **ListSelectionListener** (**javax.swing.event**)
  - Define method **valueChanged**
  - Register handler with **addListSelectionListener**

- **Example**
  - Use a **JList** to select the background color

Example

```
import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;
public class ListTest extends JFrame
{    private JList colorList;
     private Container c;
     private String colorName[ ] = {"Black", "Blue", "Cyan", "Gray",
          "Green", "Megenta", "Orange", "Pink", "Red", "White",
           "Yellow"};
     private Color colors[ ] = { Color.black, Color.blue, Color.cyan,
           Color.gray,      Color.green,    Color.magenta, Color.orange,
           Color.pink, Color.red,  Color.white, Color.yellow};
```

Example

```
public ListTest()
    {      super ("List Test" );
           c=getContentPane();
            c.setLayout (new FlowLayout());
            colorList = new JList(colorName);
            colorList.setVisibleRowCount(5);
            colorList.setSelectionMode(
ListSelectionModel.SINGLE_SELECTION);
            c.add (new JScrollPane (colorList) );
            colorList.addListSelectionListener (new ListSelectionListener()
           {
             public void valueChanged (ListSelectionEvent e)
               { c.setBackground ( colors[ colorList.getSelectedIndex()]);
                }
           });
```

Example

```
            setSize(350, 150);
            show();
      }

public static void main(String args[ ])
     { ListTest app = new ListTest();
       app.addWindowListener(new WindowAdapter(){
          public void windowClosing(WindowEvent e)
          {  System.exit (0);  }
        });
     }
 }
```

Output

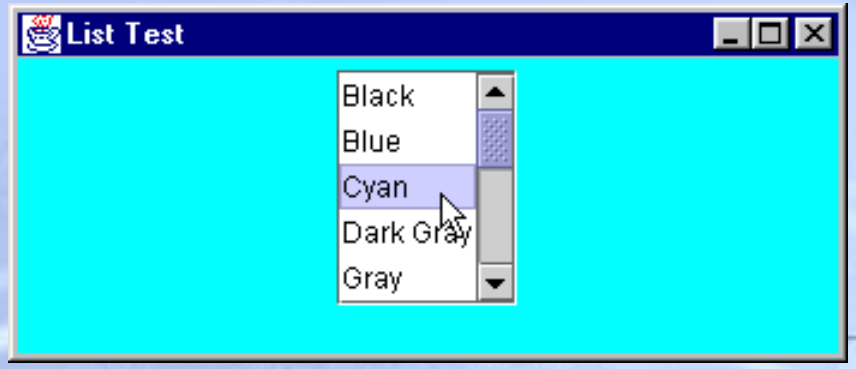


Figure : Output Frame of the previous program

JCheckBox

- State buttons
  - **JToggleButton**
    - Subclasses **JCheckBox**, **JRadioButton**
  - Have on/off (true/false) values

- Class **JCheckBox**
  - Text appears to right of checkbox
  - Constructor

    JCheckBox myBox = new JCheckBox(“Bold");

JCheckBox

- When **JCheckBox** changes
  - **ItemEvent** generated
    - Handled by an **ItemListener**, which must define **itemStateChanged**
  - Register handlers with with **addItemListener**

```
private class CheckBoxHandler implements ItemListener
{
    public void itemStateChanged( ItemEvent e )
    {
            ….
     }
}
```

JCheckBox

Javax.swing. JCheckBox

JCheckBox(String label)

JCheckBox(String label, boolean state)

JCheckBox(String label, Icon icon)

boolean isSelected()

void setSelected()

Example

```
import java.awt.event.*;
import javax.swing.*;
import java.awt.BorderLayout;
class CheckTest extends JFrame implements ActionListener
{
   JCheckBox chk1=new JCheckBox("Green");
   JCheckBox chk2=new JCheckBox("Red");
   JCheckBox chk3=new JCheckBox("Yellow");
   JCheckBox chk4=new JCheckBox("Orange");
   JCheckBox chk5=new JCheckBox("White");
   JTextArea area=new JTextArea(10,40);
   public CheckTest()
  {
         setTitle("TextEditTest");
         setSize(400, 200);
         JPanel p=new JPanel();
```

Example

```
p.add(chk1);    p.add(chk2);
p.add(chk3);    p.add(chk4);   p.add(chk5);
JPanel p1=new JPanel();
p1.setLayout(new BorderLayout());
JScrollPane scr = new JScrollPane(area);
p1.add(scr,BorderLayout.CENTER);
p1.add(p,BorderLayout.SOUTH);
chk1.addActionListener(this);
chk2.addActionListener(this);
chk3.addActionListener(this);
chk4.addActionListener(this);
chk5.addActionListener(this);
setContentPane(p1);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
show();
}
```

Example

```
public void actionPerformed(ActionEvent evt)
  {Object obj=evt.getSource();
    if(obj==chk1)
    {   if(chk1.isSelected())
        area.append("green is selected!");
      else
        area.append("green is unselected!");
    }
    else if(obj==chk2)
      {if(chk2.isSelected())
        area.append("Red is selected!");
        else
        area.append("Red is unselected");
      }
```

Example

```
else if(obj==chk3)
    {if(chk3.isSelected())
     area.append("Yellow is selected!");
     else
     area.append("Yellow is unselected!");
    }
  else if(obj==chk4)
   {if(chk4.isSelected())
    area.append("Orange is selected!");
    else
    area.append("Orange is unselected!");
    }
```

Example

```
else if(obj==chk5)
    {if(chk5.isSelected())
      area.append("White is selected!");
      else
      area.append("White is unselected!");
    }
    area.append("\n");
  }
  public static void main(String[] args)
  {  CheckTest tt=new CheckTest();
  }
}
```

Output

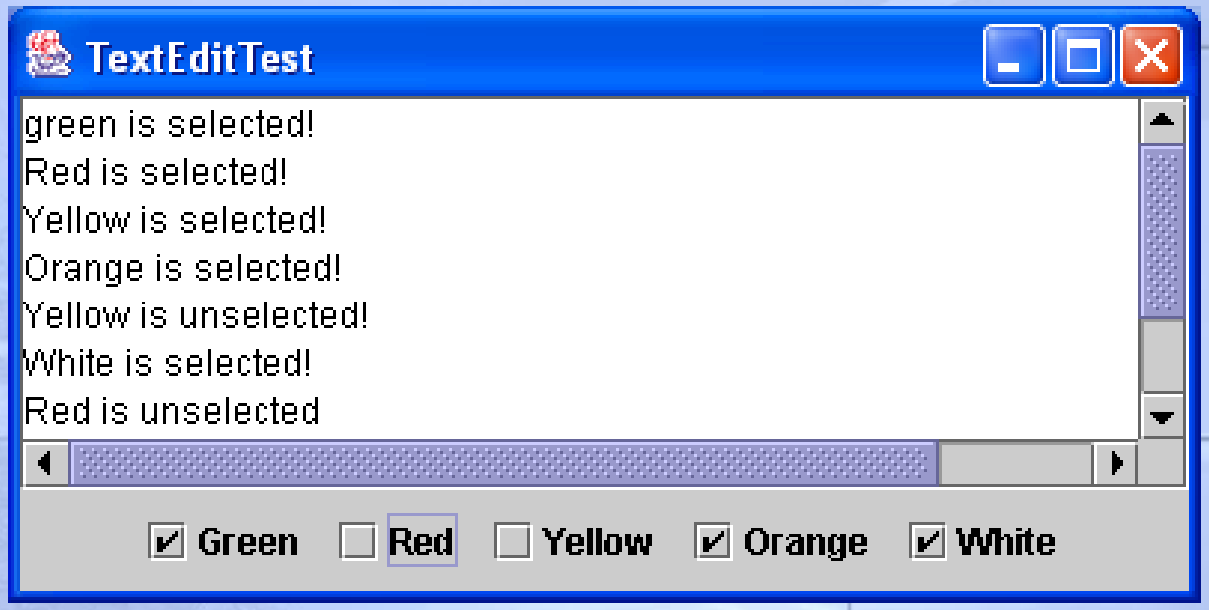


Figure : Output of the previous program

JRadioButton

- Radio buttons look just like checkboxes, but they are grouped and only one radio button in a group can be checked at any given time.

- In many cases, we want to require the user to check only one of several boxes. When another box is checked, the previous box is automatically unchecked.

- Such a group of boxes is called a **radio button group** because the buttons works like the station selector buttons on a radio.

JRadioButton

JRadioButton ()

Constructs the radio button that initially unselected

JRadioButton (Action a)

Parameters: a The action to do when click on the radio button

JRadioButton (Icon icon)

Parameters: icon The icon on the radio button

JRadioButton (Icon icon, boolean selected)

Parameters: icon The icon on the radio button
selected The initial state of the radio button

JRadioButton (String text)

Parameters: text The text to display on the radio button

JRadioButton

JRadioButton (String text, boolean selected)

Parameters: text — The text to display on the radio button
selected — The initial state of the radio button

JRadioButton (String text, Icon icon)

Parameters: text — The text to display on the radio button
icon — The icon on the radio button

JRadioButton (String text, Icon icon, boolean selected)

Parameters: text — The text to display on the radio button
icon — The icon on the radio button
selected — The initial state of the radio button

Javax.swing. ButtonGroup

void add(AbstractButton b)

- *adds the button to the group*

Example

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class RadioTest extends JFrame implements
ActionListener
{
   private JPanel buttonPanel;
   private ButtonGroup group;
   private JLabel label;
   JRadioButton rb1=new JRadioButton("small");
   JRadioButton rb2=new JRadioButton("medium",true);
   JRadioButton rb3=new JRadioButton("large");
   JRadioButton rb4=new JRadioButton("extra large");
   private static final int DEFAULT_SIZE = 12;
```

Example

```
public RadioTest()
   { setSize(400,200);
     setTitle("Radio Button Test");
     Container contentPane = getContentPane();
     label = new JLabel("The quick brown fox jumps over the lazy
dog.");
     label.setFont(new Font("Serif", Font.PLAIN, DEFAULT_SIZE));
     contentPane.add(label, BorderLayout.CENTER);
     buttonPanel=new JPanel();
     buttonPanel.add(rb1);
     buttonPanel.add(rb2);
     buttonPanel.add(rb3);
     buttonPanel.add(rb4);
     contentPane.add(buttonPanel, BorderLayout.SOUTH);
```

Example

```
    rb1.addActionListener(this);
    rb2.addActionListener(this);
    rb3.addActionListener(this);
    rb4.addActionListener(this);
    group=new ButtonGroup();
    group.add(rb1);
    group.add(rb2);
    group.add(rb3);
    group.add(rb4);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    show();
 }
public static void main(String[] args)
  { RadioTest frame = new RadioTest();
  }
```

Example

```
public void actionPerformed(ActionEvent evt)
   {
      if(evt.getSource()==rb1)
         label.setFont(new Font("Serif", Font.PLAIN, 8));
      else if(evt.getSource()==rb2)
         label.setFont(new Font("Serif", Font.PLAIN, 12));
      else if(evt.getSource()==rb3)
         label.setFont(new Font("Serif", Font.PLAIN, 16));
     else if(evt.getSource()==rb4)
         label.setFont(new Font("Serif", Font.PLAIN, 20));
   }
}
```

Example

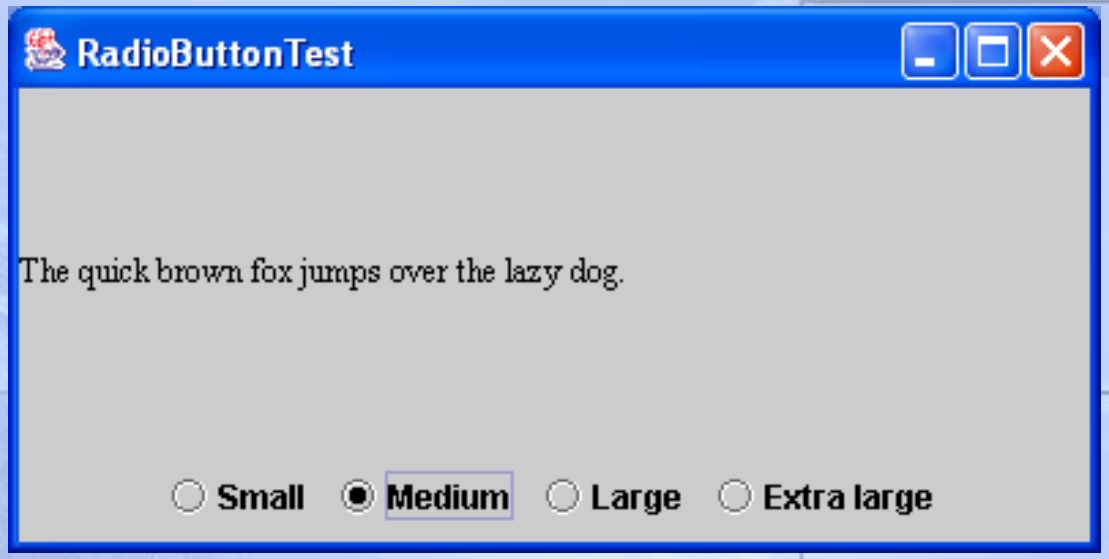


Figure : Output of the previous program

JTextArea

It is used to collect user input that is more than one line long.

javax.swing.JTextArea

JTextArea( )
JTextArea (int rows, int clos)
JTextArea (String text, int rows, int cols)

void setRows(int rows)
void setColumns(int cols)
void append(String newText)
void setLineWrap(boolean wrap)

JTextArea

e.g.,

```
JPanel p=new JPanel();
JTextArea  tA=new JTextArea  (“Hello Welcome to UCSY!”, 20,20);
p.add(tA);
```

e.g.,

```
JTextArea  tf=new JTextArea (30,20);
tA.setText(“Hello Welcome to”);
tA.append(“UCSY!!!”);
```

e.g.,

```
JTextArea  tA=new JTextArea ();
TA.setRows(30);
TA.setColumns(20);
tA.setText(“Hello…..”);
```

Example

```
import java.awt.event.*;
import javax.swing.*;
import java.awt.BorderLayout;
class textExample extends JFrame implements ActionListener
{ JTextField text=new JTextField();
  JTextArea area=new JTextArea(10,20);
  JButton show=new JButton("Show");
  JButton clear=new JButton("Clear");
  public textExample()
  { this.setTitle("Text Example");
     setSize(300,200);
     setLocation(100,100);
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Example

```
JPanel btnPanel=new JPanel();
btnPanel.add(show);
btnPanel.add(clear);
show.addActionListener(this);
clear.addActionListener(this);
JPanel panel=new JPanel();
panel.setLayout(new BorderLayout());
panel.add(text,BorderLayout.NORTH);
panel.add(area,BorderLayout.CENTER);
  panel.add(btnPanel,BorderLayout.SOUTH);
setContentPane(panel);
setVisible(true);
}
```

Example

```
public void actionPerformed(ActionEvent e)
{
  Object obj=e.getSource();
  if(obj==show)
  {
     area.append(text.getText() + "\n");
     text.setText("");
  }
  if(obj==clear)
    area.setText("");
}
public static void main(String[] args)
{
  new textExample();
}
}
```

Output

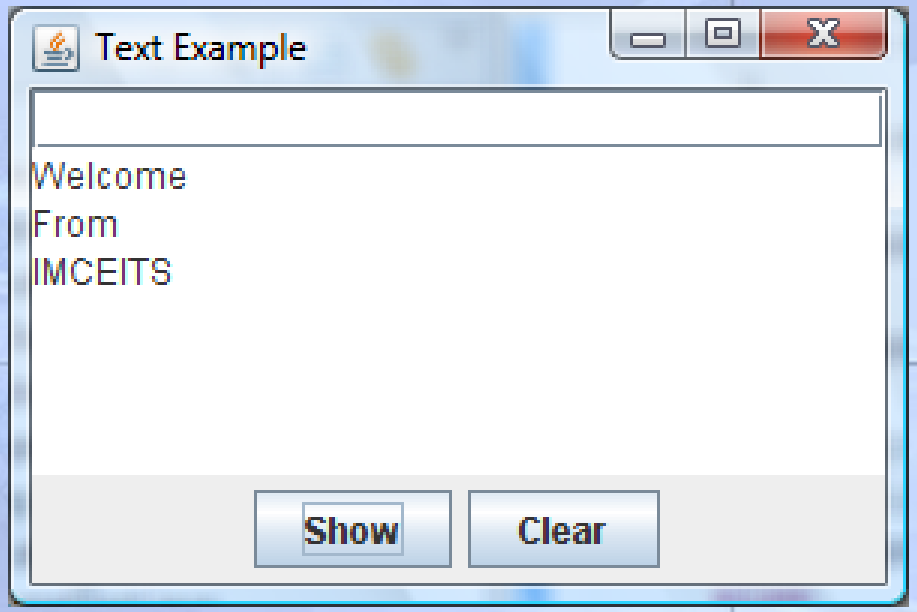


Figure : Output of the previous program

Menu

- Java provides several classes—
  - **JMenuBar**
  - **JMenu**
  - **JMenuItem**
  - **JCheckBoxMenuItem**
  - **JRadioButtonMenuItem**

- A JFrame or JApplet can hold a *menu bar* to which the *pull-down menus* are attached.

- A menu bar on the top of the window contains the name of the *pull-down* menus. Clicking on a name opens the menu containing menu items and submenus. When the user clicks on a menu item, all menus are closed and a message is sent to the program.

Menu

Building Menu

1. Create a menu bar as follows:

```
JMenuBar mb=new JMenuBar();
```

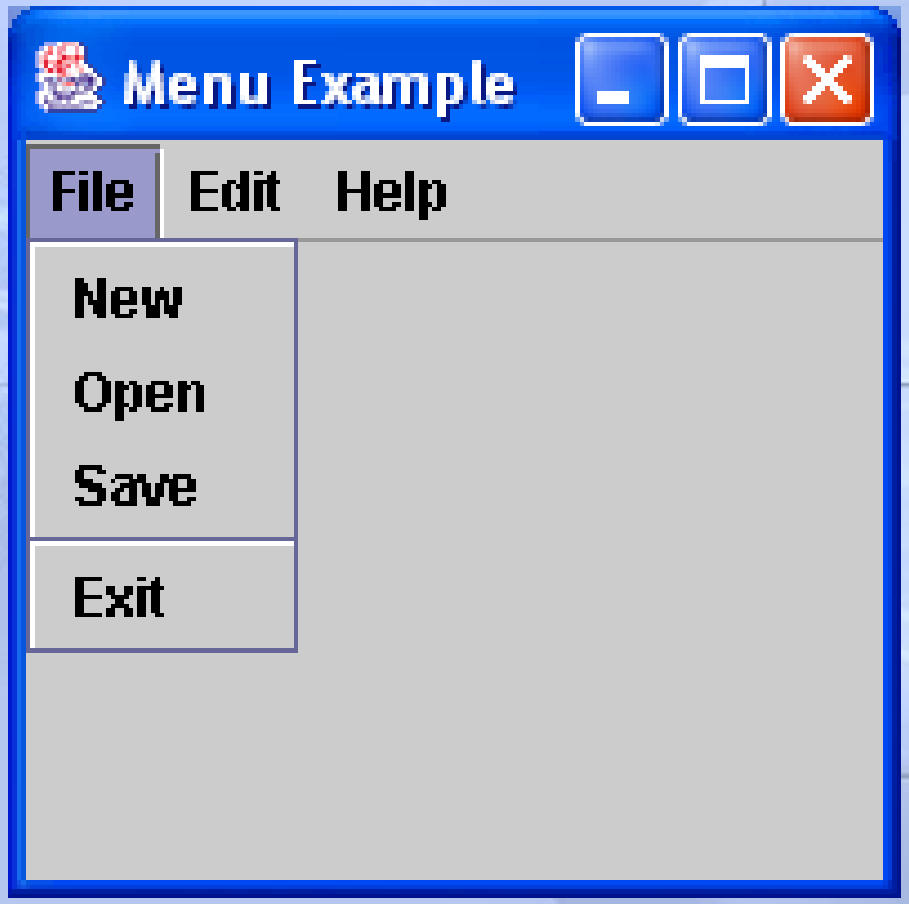
2. Creat the menu objects as follows:

```
JMenu file=new JMenu(“File”);
```

3. Creating the menuItem object as follows:

```
JMenuItem newItem=new JMenuItem("New");
```

Example of Creating Menu

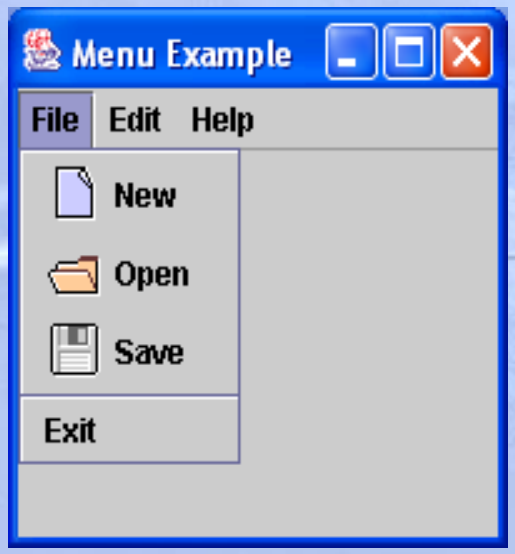


JPopup Menus

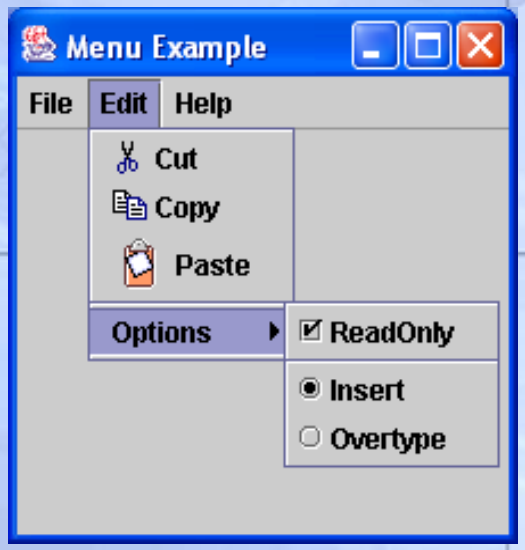
- Context-sensitive popup menus
  - Created with class **JPopupMenu**
    - Subclass of **JComponent**
  - Options specific to component
    - Popup trigger event
    - Most systems, right mouse click

Example of JMenus with Icon and JPopup Menus

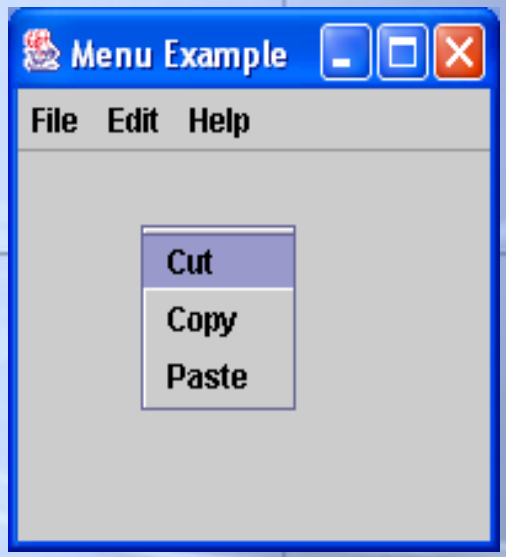
Icons in Menu Items



Checkbox and Radiobutton Menu Items



Popup Menu



Example

- **Icons in Menu Item**

```
JMenuItem newItem=new JMenuItem("New",new ImageIcon("NEW.gif"));
JMenuItem openItem=new JMenuItem("Open",new ImageIcon("OPEN.gif"));
JMenuItem saveItem=new JMenuItem("Save",new ImageIcon("SAVE.gif"));
```

- **CheckBox and Radio Button Menu Item**

```
JMenuItem cutItem=new JMenuItem("Cut", new ImageIcon("CUT.gif"));
JMenuItem copyItem=new JMenuItem("Copy", new ImageIcon("COPY.gif"));
JMenuItem pasteItem=new JMenuItem("Paste", new
                                        ImageIcon("PASTE.gif"));
JMenu optionsMenu=new JMenu("Options");
JCheckBoxMenuItem readOnlyItem=new JCheckBoxMenuItem("ReadOnly");
JRadioButtonMenuItem insertItem=new JRadioButtonMenuItem("Insert");
JRadioButtonMenuItem overtypeItem=new
                                JRadioButtonMenuItem("Overtype");
```

Example

- Pop-up Menu

```
PopupMenu popup = new Popupmenu("Edit");
MemuItem cutItem= new MenuItem ("Cut");
MemuItem copyItem= new MenuItem ("Copy");
MemuItem pasteItem= new MenuItem ("Paste");
popup.add(cutItem);
popup.add(copyItem);
popup.add(pasteItem);

getContentPane().addMouseListener(new MouseAdapter()
{ public void mouseReleased(MouseEvent event)
   {
    if(event.isPopupTrigger())
    popup.show(event.getComponent(),event.getX(),event.getY());
   }
});
```

Thank You!