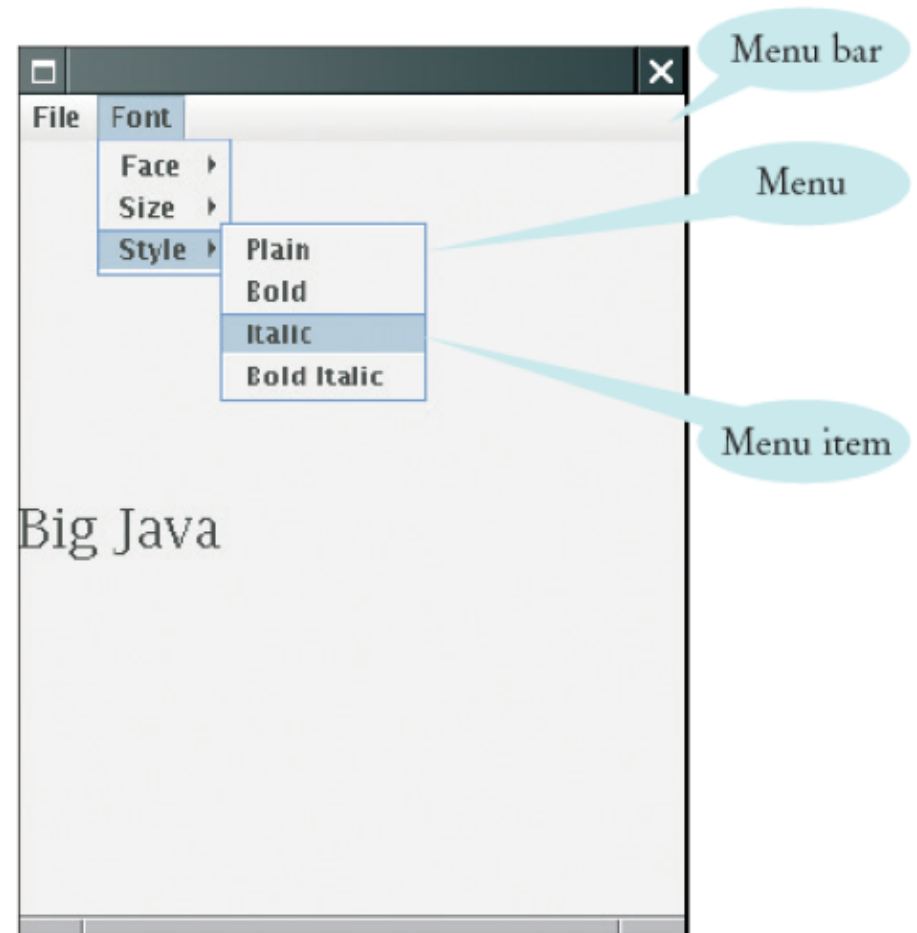


Menu

- Ogni frame contiene una barra dei menu
- La barra contiene dei menu
- Un menu contiene sub-menu e item del menu





Nuove classi javax.swing utilizzate

- **JMenuBar**
 - implementazione della barra dei menu di una finestra
- **JMenuItem**
 - implementazione di un elemento di un menu
 - praticamente un pulsante posizionato in una lista
- **JMenu** (sottoclasse di **JMenuItem**)
 - implementazione di un menu
 - essenzialmente un pulsante con un pop-up menu associato
 - contiene lista di oggetti di tipo **JMenuItem** e **Jseparator**
- **JComponent** è un supertipo per ciascuna di queste classi



Voci del menu

- Gli item e i sub-menu si aggiungono con il metodo add:

```
JMenuItem fileExitItem = new JMenuItem("Exit");  
fileMenu.add(fileExitItem);
```

- Un item non ha ulteriori sub-menu
- Gli item generano eventi del tipo **ActionEvent**
- Si può aggiungere un ascoltatore ad ogni item:

```
fileExitItem.addActionListener(listener);
```

- In genere si aggiungono ascoltatori di azioni solo agli item e non ai menu o alla barra dei menu



Programma esempio

- Costruire un menu tipo
- Catturare gli eventi generati dai menu item
- Per una migliore leggibilità, scrivere un metodo per ogni menu o insieme di menu correlati
 - **createFaceItem**: crea item per cambiare il font
 - **createSizeItem**
 - **createStyleItem**



File MenuFrameViewer.java

```
01: import javax.swing.JFrame;
02:
03: /**
04:     This program tests the MenuFrame.
05: */
06: public class MenuFrameViewer
07: {
08:     public static void main(String[] args)
09:     {
10:         JFrame frame = new MenuFrame();
11:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:         frame.setVisible(true);
13:     }
14: }
15:
```



File MenuFrame.java

```
001: import java.awt.BorderLayout;  
002: import java.awt.Font;  
003: import java.awt.GridLayout;  
004: import java.awt.event.ActionEvent;  
005: import java.awt.event.ActionListener;  
006: import javax.swing.ButtonGroup;  
007: import javax.swing.JButton;  
008: import javax.swing.JCheckBox;  
009: import javax.swing.JComboBox;  
010: import javax.swing.JFrame;  
011: import javax.swing.JLabel;  
012: import javax.swing.JMenu;  
013: import javax.swing.JMenuBar;  
014: import javax.swing.JMenuItem;  
015: import javax.swing.JPanel;  
016: import javax.swing.JRadioButton;
```




File MenuFrame.java

```
017: import javax.swing.border.EtchedBorder;
018: import javax.swing.border.TitledBorder;
019:
020: /**
021:     This frame has a menu with commands to change the font
022:     of a text sample.
023: */
024: public class MenuFrame extends JFrame
025: {
026:     /**
027:         Constructs the frame.
028:     */
029:     public MenuFrame()
030:     {
031:         // Construct text sample
032:         sampleField = new JLabel("Big Java");
033:         add(sampleField, BorderLayout.CENTER);
034:
```

Nota:

Il gestore di layout
si può usare
anche
per JFrame



File MenuFrame.java

```
035:         // Construct menu
036:         JMenuBar menuBar = new JMenuBar();
037:         setJMenuBar(menuBar);
038:         menuBar.add(createFileMenu());
039:         menuBar.add(createFontMenu());
040:
041:         facename = "Serif";
042:         fontsize = 24;
043:         fontstyle = Font.PLAIN;
044:
045:         setSampleFont();
046:         setSize(FRAME_WIDTH, FRAME_HEIGHT);
047:     }
048:
049:     /**
050:      * Creates the File menu.
051:      * @return the menu
052:      */
```




File MenuFrame.java

```
053:     public JMenu createFileMenu()
054:     {
055:         JMenu menu = new JMenu("File");
056:         menu.add(createFileExitItem());
057:         return menu;
058:     }
059:
060:     /**
061:         Creates the File->Exit menu item and sets its
            // action listener.
062:         @return the menu item
063:     */
064:     public JMenuItem createFileExitItem()
065:     {
066:         JMenuItem item = new JMenuItem("Exit");
067:         class MenuItemListener implements ActionListener
068:         {
069:             public void actionPerformed(ActionEvent event)
```



File MenuFrame.java

```
070:         {
071:             System.exit(0);
072:         }
073:     }
074:     ActionListener listener = new MenuItemListener();
075:     item.addActionListener(listener);
076:     return item;
077: }
078:
079: /**
080:     Creates the Font submenu.
081:     @return the menu
082: */
083: public JMenu createFontMenu()
084: {
085:     JMenu menu = new JMenu("Font");
086:     menu.add(createFaceMenu());
```



File MenuFrame.java

```
087:         menu.add(createSizeMenu());
088:         menu.add(createStyleMenu());
089:         return menu;
090:     }
091:
092:     /**
093:         Creates the Face submenu.
094:         @return the menu
095:     */
096:     public JMenu createFaceMenu()
097:     {
098:         JMenu menu = new JMenu("Face");
099:         menu.add(createFaceItem("Serif"));
100:         menu.add(createFaceItem("SansSerif"));
101:         menu.add(createFaceItem("Monospaced"));
102:         return menu;
103:     }
104:
```




File MenuFrame.java

```
105:    /**
106:        Creates the Size submenu.
107:        @return the menu
108:    */
109:    public JMenu createSizeMenu()
110:    {
111:        JMenu menu = new JMenu("Size");
112:        menu.add(createSizeItem("Smaller", -1));
113:        menu.add(createSizeItem("Larger", 1));
114:        return menu;
115:    }
116:
117:    /**
118:        Creates the Style submenu.
119:        @return the menu
120:    */
121:    public JMenu createStyleMenu()
122:    {
```



File MenuFrame.java

```
123:         JMenu menu = new JMenu("Style");
124:         menu.add(createStyleItem("Plain", Font.PLAIN));
125:         menu.add(createStyleItem("Bold", Font.BOLD));
126:         menu.add(createStyleItem("Italic", Font.ITALIC));
127:         menu.add(createStyleItem("Bold Italic", Font.BOLD
128:             + Font.ITALIC));
129:         return menu;
130:     }
131:
132:
133:     /**
134:         Creates a menu item to change the font face and
            // set its action listener.
135:         @param name the name of the font face
136:         @return the menu item
137:     */
138:     public JMenuItem createFaceItem(final String name)
139:     {
```



File MenuFrame.java

```
140:         JMenuItem item = new JMenuItem(name);
141:         class MenuItemListener implements ActionListener
142:         {
143:             public void actionPerformed(ActionEvent event)
144:             {
145:                 facename = name;
146:                 setSampleFont();
147:             }
148:         }
149:         ActionListener listener = new MenuItemListener();
150:         item.addActionListener(listener);
151:         return item;
152:     }
153:
```



File MenuFrame.java

```
154:    /**
155:        Creates a menu item to change the font size
156:        and set its action listener.
157:        @param name the name of the menu item
158:        @param ds the amount by which to change the size
159:        @return the menu item
160:    */
161:    public JMenuItem createSizeItem(String name, final int ds)
162:    {
163:        JMenuItem item = new JMenuItem(name);
164:        class MenuItemListener implements ActionListener
165:        {
166:            public void actionPerformed(ActionEvent event)
167:            {
168:                fontsize = fontsize + ds;
169:                setSampleFont();
170:            }
171:        }
```




File MenuFrame.java

```
172:         ActionListener listener = new MenuItemListener();
173:         item.addActionListener(listener);
174:         return item;
175:     }
176:
177:     /**
178:      Creates a menu item to change the font style
179:      and set its action listener.
180:      @param name the name of the menu item
181:      @param style the new font style
182:      @return the menu item
183:     */
184:     public JMenuItem createStyleItem(String name,
        final int style)
185:     {
186:         JMenuItem item = new JMenuItem(name);
187:         class MenuItemListener implements ActionListener
188:         {
```




File MenuFrame.java

```
189:         public void actionPerformed(ActionEvent event)
190:         {
191:             fontstyle = style;
192:             setSampleFont();
193:         }
194:     }
195:     ActionListener listener = new MenuItemListener();
196:     item.addActionListener(listener);
197:     return item;
198: }
199:
200: /**
201:     Sets the font of the text sample.
202: */
203: public void setSampleFont()
204: {
```



File MenuFrame.java

```
205:         Font f = new Font(facename, fontstyle, fontsize);
206:         sampleField.setFont(f);
207:         sampleField.repaint();
208:     }
209:
210:     private JLabel sampleField;
211:     private String facename;
212:     private int fontstyle;
213:     private int fontsize;
214:
215:     private static final int FRAME_WIDTH = 300;
216:     private static final int FRAME_HEIGHT = 400;
217: }
218:
219:
```



Area di testo

- Si usa **JTextArea** per mostrare linee di testo multiple
- Si possono specificare numero di righe e colonne:

```
final int ROWS = 10;  
final int COLUMNS = 30;  
JTextArea textArea = new JTextArea(ROWS, COLUMNS);
```

- **setText**: per impostare il testo di un campo o un' area di testo
- **append**: per aggiungere testo alla fine di un' area di testo



Area di testo

- Si usa “\n” carattere per separare righe:

```
textArea.append(account.getBalance() + "\n");
```

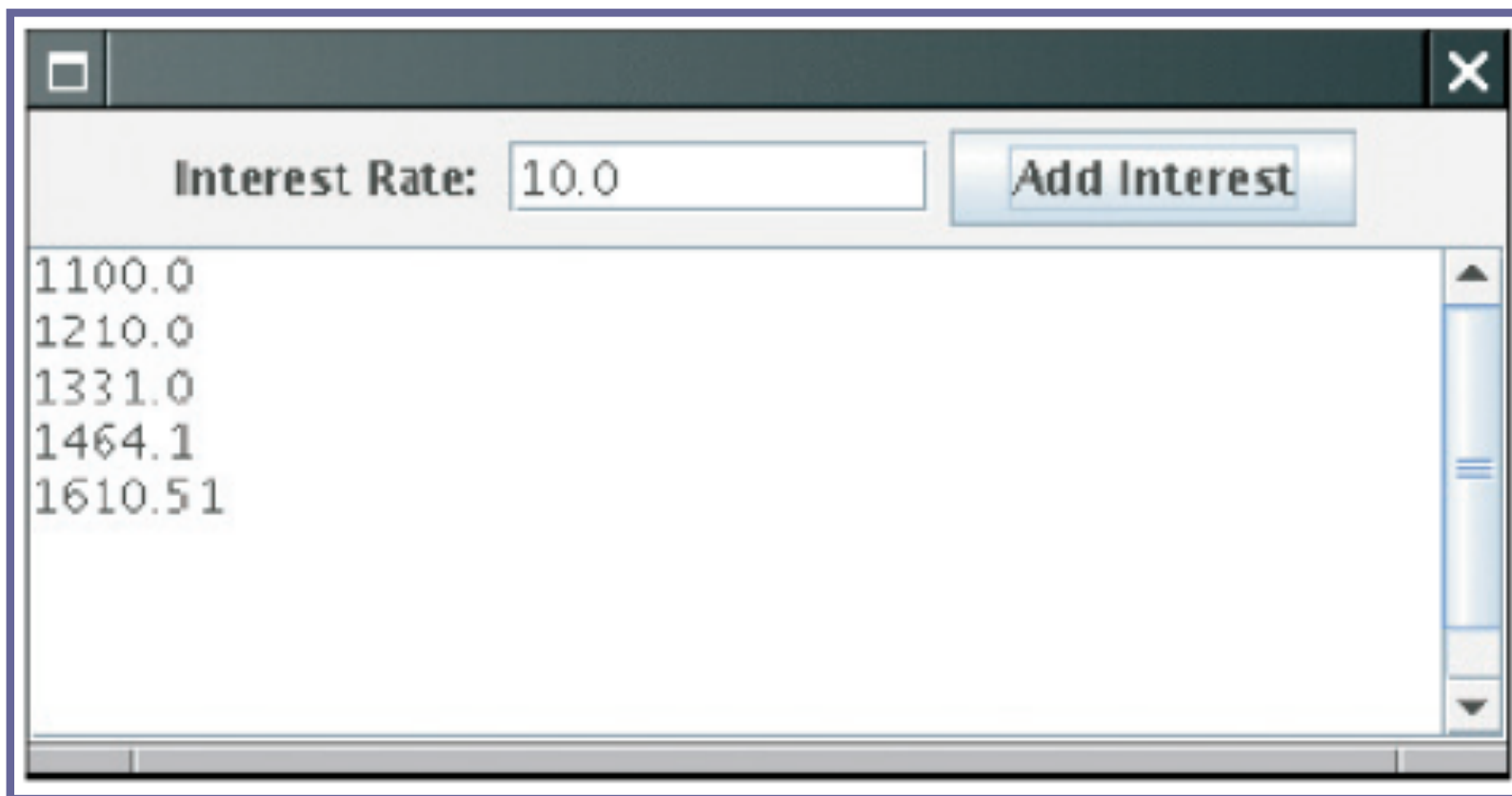
- Se si vuole usare un'area di testo solo per visualizzare un testo:

```
textArea.setEditable(false);  
// program can call setText and append to change it
```

- Si possono aggiungere barre di scorrimento:

```
JTextArea textArea = new JTextArea(ROWS, COLUMNS);  
JScrollPane scrollPane = new JScrollPane(textArea);
```

Area di testo



The screenshot shows a software window with a title bar containing a minimize button, a maximize button, and a close button. The window contains a label "Interest Rate:" followed by a text input field containing the value "10.0". To the right of the input field is a button labeled "Add Interest". Below these elements is a large text area containing a list of values: 1100.0, 1210.0, 1331.0, 1464.1, and 1610.51. A vertical scrollbar is visible on the right side of the text area.

Interest Rate:

1100.0
1210.0
1331.0
1464.1
1610.51



File TextAreaViewer.java

```
01: import java.awt.BorderLayout;
02: import java.awt.event.ActionEvent;
03: import java.awt.event.ActionListener;
04: import javax.swing.JButton;
05: import javax.swing.JFrame;
06: import javax.swing.JLabel;
07: import javax.swing.JPanel;
08: import javax.swing.JScrollPane;
09: import javax.swing.JTextArea;
10: import javax.swing.JTextField;
11:
12: /**
13:     This program shows a frame with a text area that
14:     displays the growth of an investment.
15: */
16: public class TextAreaViewer
17: {
```



File TextAreaViewer.java

```
18:    public static void main(String[] args)
19:    {
20:        JFrame frame = new JFrame();
21:
22:        // The application adds interest to this bank account
23:        final BankAccount account =
                new BankAccount(INITIAL_BALANCE);
24:        // The text area for displaying the results
25:        final int AREA_ROWS = 10;
26:        final int AREA_COLUMNS = 30;
27:
28:        final JTextArea textArea = new JTextArea(
29:            AREA_ROWS, AREA_COLUMNS);
30:        textArea.setEditable(false);
31:        JScrollPane scrollPane = new JScrollPane(textArea);
32:
33:        // The label and text field for entering the
            // interest rate
```



File TextAreaViewer.java

```
34:         JLabel rateLabel = new JLabel("Interest Rate: ");
35:
36:         final int FIELD_WIDTH = 10;
37:         final JTextField rateField =
38:             new JTextField(FIELD_WIDTH);
39:         rateField.setText("" + DEFAULT_RATE);
40:         // The button to trigger the calculation
41:         JButton calculateButton = new JButton("Add Interest");
42:
43:         // The panel that holds the input components
44:         JPanel northPanel = new JPanel();
45:         northPanel.add(rateLabel);
46:         northPanel.add(rateField);
47:         northPanel.add(calculateButton);
48:
49:         frame.add(northPanel, BorderLayout.NORTH);
50:         frame.add(scrollPane);
51:
```




File TextAreaViewer.java

```
52:         class CalculateListener implements ActionListener
53:         {
54:             public void actionPerformed(ActionEvent event)
55:             {
56:                 double rate = Double.parseDouble(
57:                     rateField.getText());
58:                 double interest = account.getBalance()
59:                     * rate / 100;
60:                 account.deposit(interest);
61:                 textArea.append(account.getBalance() + "\n");
62:             }
63:         }
64:
65:         ActionListener listener = new CalculateListener();
66:         calculateButton.addActionListener(listener);
67:
```



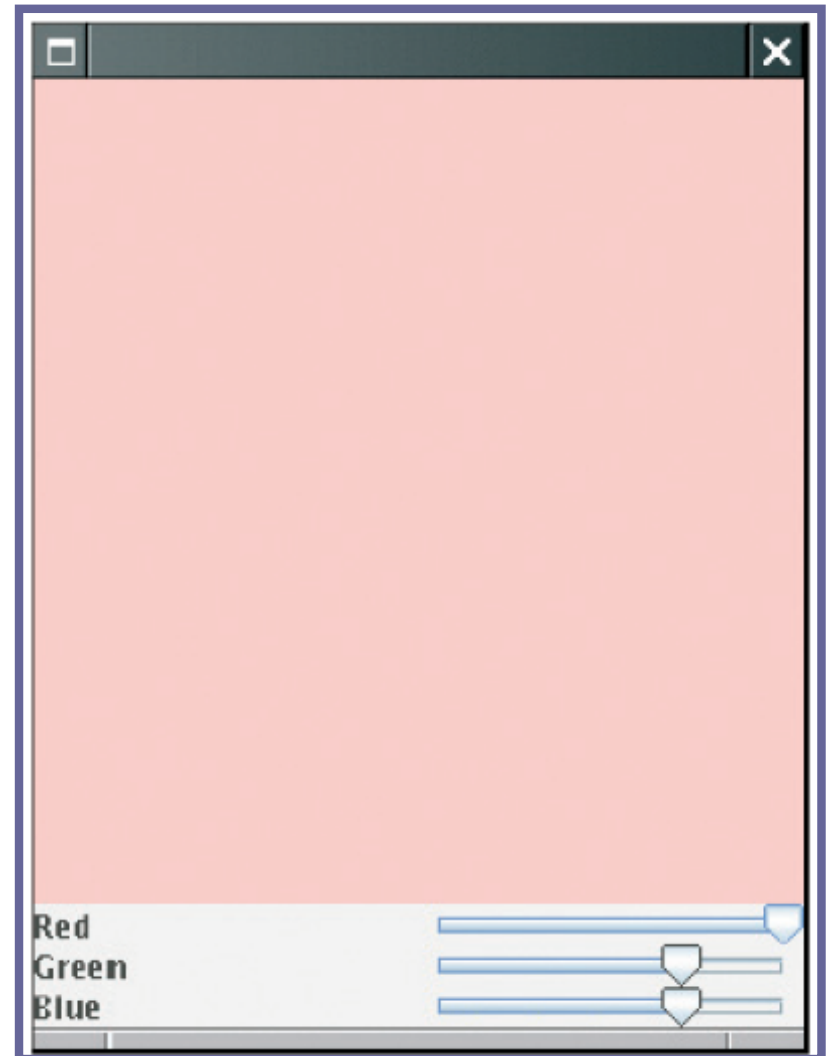
File TextAreaViewer.java

```
68:         frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
69:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
70:         frame.setVisible(true);
71:     }
72:
73:     private static final double DEFAULT_RATE = 10;
74:     private static final double INITIAL_BALANCE = 1000;
75:
76:     private static final int FRAME_WIDTH = 400;
77:     private static final int FRAME_HEIGHT = 200;
78: }
```



Consultare la documentazione di Swing

- Per ottenere effetti grafici particolari si può consultare la documentazione del pacchetto Swing
- Consideriamo il problema di realizzare un mixer dei colori Rosso, Verde, Blu per visualizzare ogni possibile colore





Esempio

- Come facciamo a stabilire se c'è un cursore a scorrimento (slider) in Swing?
 - Consulta la documentazione e controlla tutti i nomi che cominciano per J
 - **JSlider** può essere un buon candidato
- Domande successive:
 - Come costruisco un **JSlider**?
 - Come ricevo la notifica quando l'utente lo muove?
 - Come posso determinare i valori corrispondenti alla posizione del cursore?
- Dopo aver trovato la risposta a queste domande possiamo migliorare altri aspetti grafici del mixer.



Esempio: osservazioni

- Ci sono più di 50 metodi descritti direttamente nella documentazione di **JSlider** e oltre 250 metodi ereditati
- Alcune descrizioni non sono di facile comprensione
- Bisogna sviluppare l'attitudine a sorvolare dettagli meno significativi



Come si istanzia uno JSlider?

- Nella documentazione di Java 5.0 ci sono sei costruttori per **JSlider**
- Scegli il più appropriato
 - `public JSlider()`
Crea uno slider orizzontale con intervallo da 0 a 100 e valore iniziale 50
 - `public JSlider(BoundedRangeModel brm)`
Crea uno slider orizzontale usando lo specificato **BoundedRangeModel**
 - `public JSlider(int min, int max, int value)`
Crea uno slider orizzontale usando min, max e value specificati



Come riceviamo la notifica che uno `JSlider` è stato mosso?

- Non c'è un metodo `addActionListener`
- C'è un metodo

```
public void addChangeListener(ChangeListener l)
```

- Segui il link su `ChangeListener`
 - ha un solo metodo:

```
void stateChanged(ChangeEvent e)
```




Come riceviamo la notifica che uno **JSlider** è stato mosso?

- Sembra che il metodo venga invocato ogni volta che l'utente muove lo slider
- Cos'è un **ChangeEvent**?
 - Eredita **getSource** da **EventObject**
 - **getSource**: ci dice quale componente ha generato l'evento



Come possiamo stabilire il valore impostato dall'utente con gli `JSlider`?

- Aggiungi un **`ChangeListener`** a ogni slider
- Quando uno slider cambia stato, si chiama il metodo **`stateChanged`**
- Determina il nuovo valore dello slider
- Ricalcola il valore del colore
- Ridisegna il pannello del colore



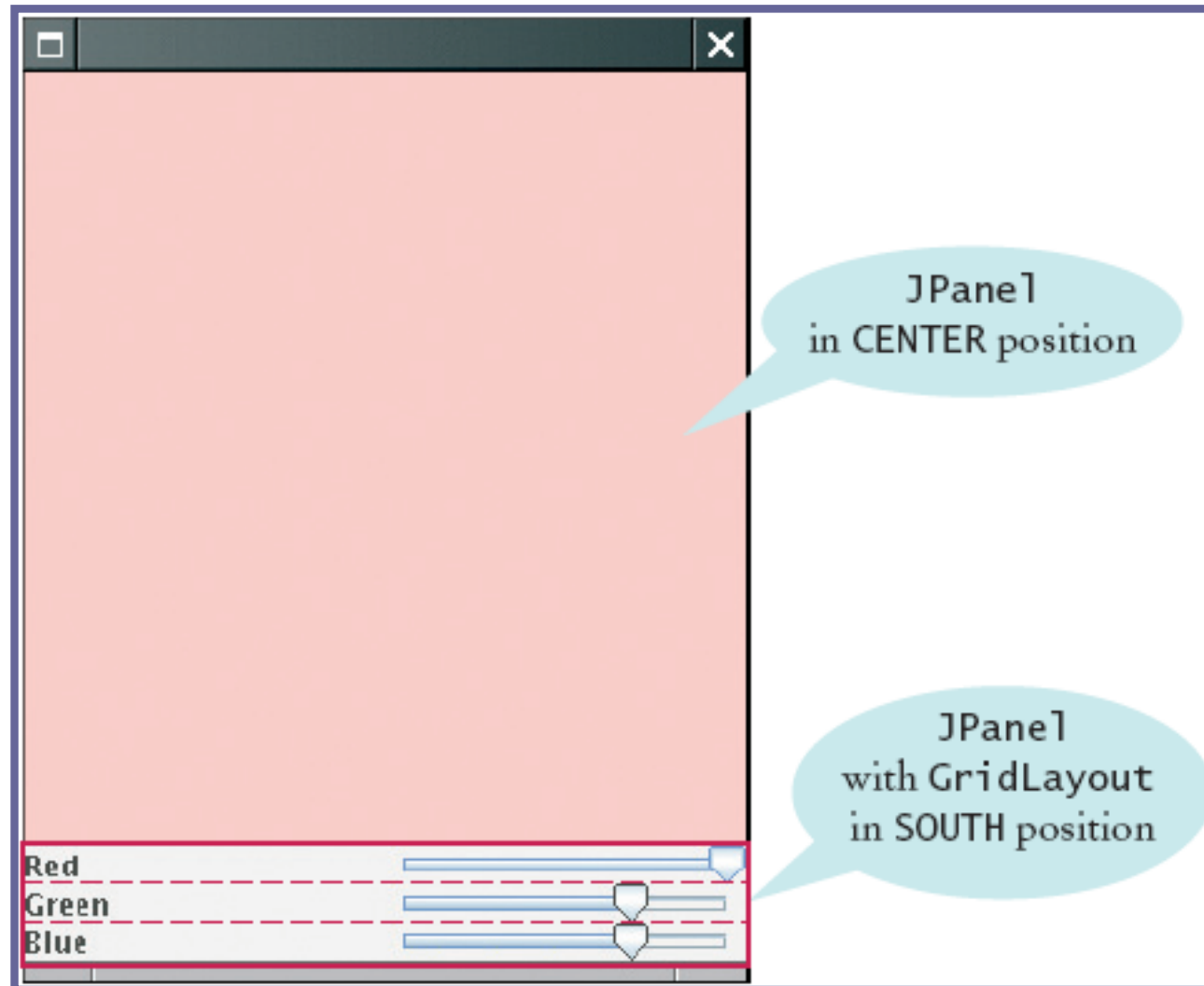
Come possiamo stabilire il valore impostato dall'utente con gli `JSlider`?

- Dobbiamo recuperare il valore corrente dallo slider
- Controlla tra i metodi che cominciano con `get`:

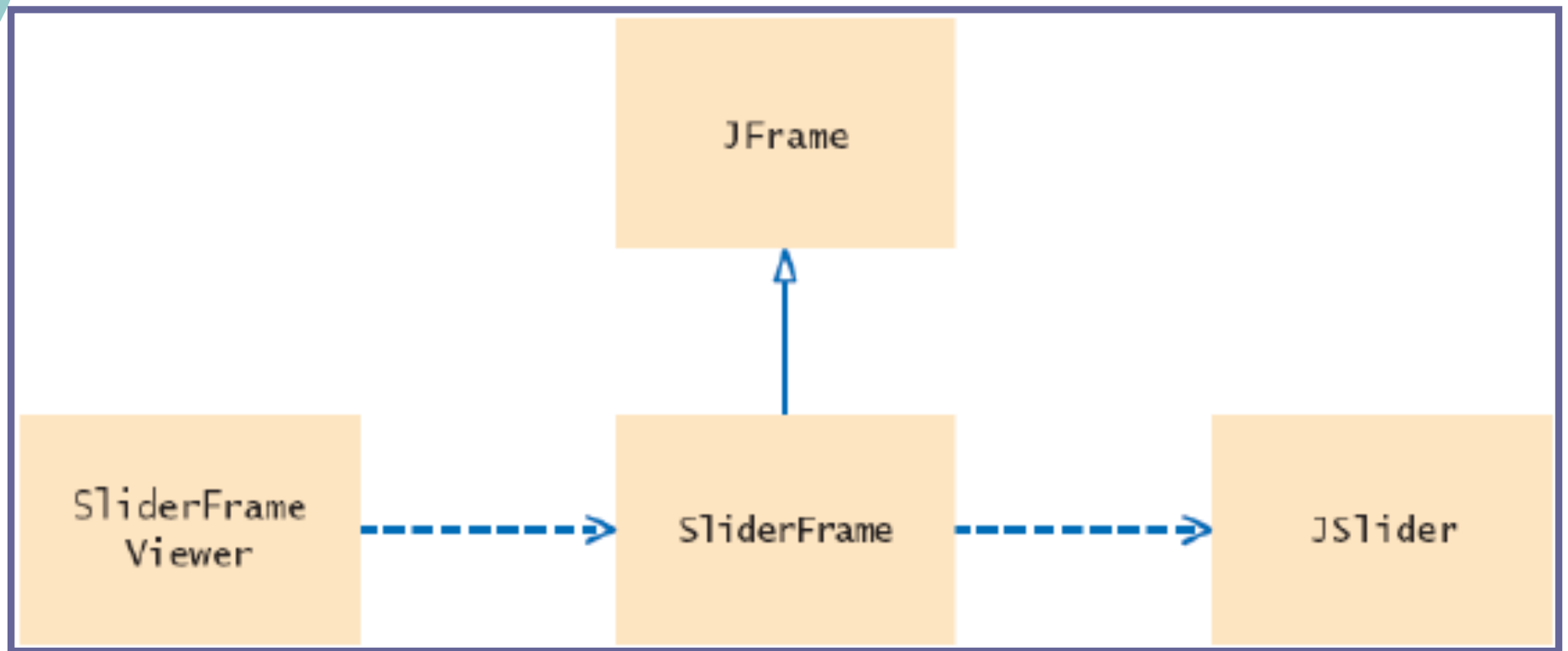
```
public int getValue()
```

- Restituisce il valore dello slider.

I componenti dello `SliderFrame`



Classi del programma





File SliderFrameViewer.java

```
01: import javax.swing.JFrame;  
02:  
03: public class SliderFrameViewer  
04: {  
05:     public static void main(String[] args)  
06:     {  
07:         SliderFrame frame = new SliderFrame();  
08:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
09:         frame.setVisible(true);  
10:     }  
11: }  
12:
```



File SliderFrame.java

```
01: import java.awt.BorderLayout;
02: import java.awt.Color;
03: import java.awt.GridLayout;
04: import javax.swing.JFrame;
05: import javax.swing.JLabel;
06: import javax.swing.JPanel;
07: import javax.swing.JSlider;
08: import javax.swing.event.ChangeListener;
09: import javax.swing.event.ChangeEvent;
10:
11: public class SliderFrame extends JFrame
12: {
13:     public SliderFrame()
14:     {
15:         colorPanel = new JPanel();
16:
```



File SliderFrame.java

```
17:         add(colorPanel, BorderLayout.CENTER);
18:         createControlPanel();
19:         setSampleColor();
20:         setSize(FRAME_WIDTH, FRAME_HEIGHT);
21:     }
22:
23:     public void createControlPanel()
24:     {
25:         class ColorListener implements ChangeListener
26:         {
27:             public void stateChanged(ChangeEvent event)
28:             {
29:                 setSampleColor();
30:             }
31:         }
32:
```




File SliderFrame.java

```
33:         ChangeListener listener = new ColorListener();
34:
35:         redSlider = new JSlider(0, 100, 100);
36:         redSlider.addChangeListener(listener);
37:
38:         greenSlider = new JSlider(0, 100, 70);
39:         greenSlider.addChangeListener(listener);
40:
41:         blueSlider = new JSlider(0, 100, 70);
42:         blueSlider.addChangeListener(listener);
43:
44:         JPanel controlPanel = new JPanel();
45:         controlPanel.setLayout(new GridLayout(3, 2));
46:
47:         controlPanel.add(new JLabel("Red"));
48:         controlPanel.add(redSlider);
49:
```



File SliderFrame.java

```
50:         controlPanel.add(new JLabel("Green"));
51:         controlPanel.add(greenSlider);
52:
53:         controlPanel.add(new JLabel("Blue"));
54:         controlPanel.add(blueSlider);
55:
56:         add(controlPanel, BorderLayout.SOUTH);
57:     }
58:
59:     /**
60:      Reads the slider values and sets the panel to
61:      the selected color.
62:     */
63:     public void setSampleColor()
64:     {
65:         // Read slider values
66:
```



File SliderFrame.java

```
67:         float red = 0.01F * redSlider.getValue();
68:         float green = 0.01F * greenSlider.getValue();
69:         float blue = 0.01F * blueSlider.getValue();
70:
71:         // Set panel background to selected color
72:
73:         colorPanel.setBackground(new Color(red, green, blue));
74:         colorPanel.repaint();
75:     }
76:
77:     private JPanel colorPanel;
78:     private JSlider redSlider;
79:     private JSlider greenSlider;
80:     private JSlider blueSlider;
81:
82:     private static final int FRAME_WIDTH = 300;
83:     private static final int FRAME_HEIGHT = 400;
84: }
```

Container

- Top-level container

JApplet



JDialog



JFrame

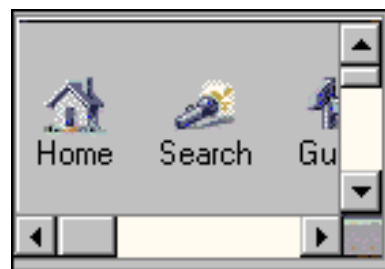


- General-purpose container

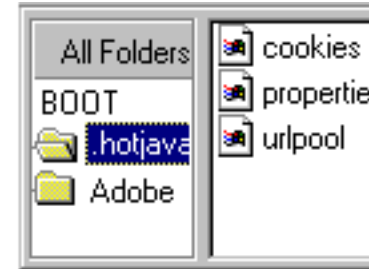
JPanel



JScrollPane



JSplitPane



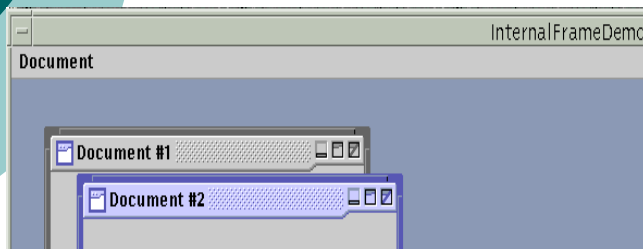
JTabbedPane



Container

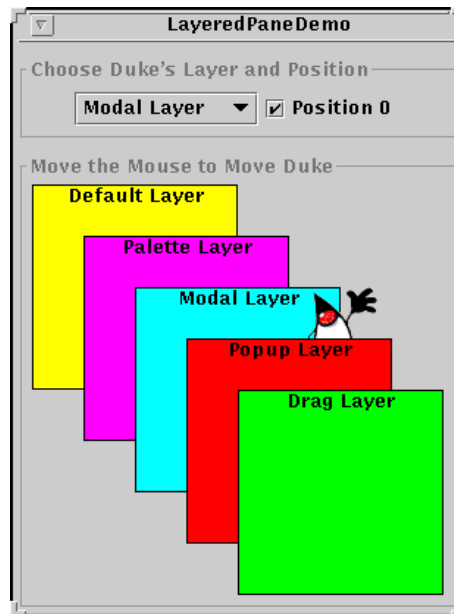
- Special-purpose container

JInternalFrame



Permette di inserire frame dentro altri frame

JLayeredPane



Permette di inserire componenti a vari livelli di profondità

JToolBar



Permette di semplificare l'attivazione di determinate funzioni per mezzo di semplici pulsanti

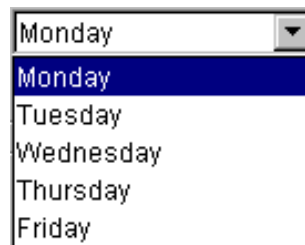
Controlli di base

JButtons

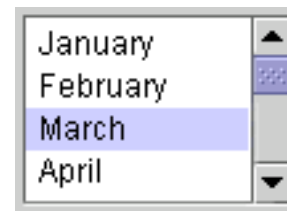


Include buttons,
radioButtons, checkbox,
MenuItem, ToggleButton

JComboBox



JList



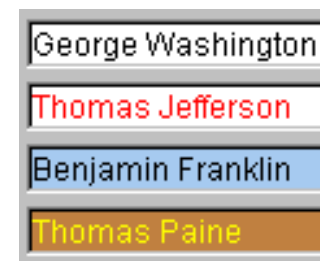
JMenu



JSlider



TextField



Include
JPasswordField,
JTextArea

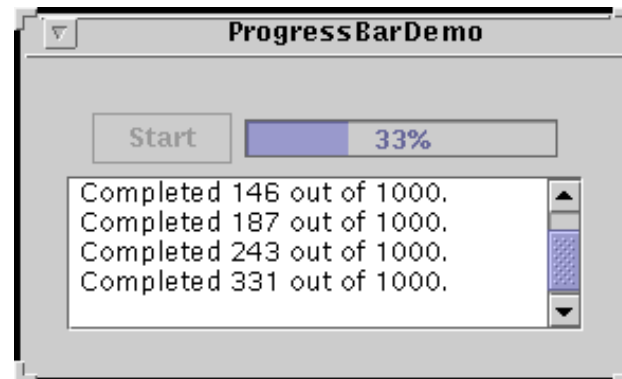
Visualizzatori di informazioni non editabili

JLabel



**Può includere
immagini e/o testo**

JProgressBar

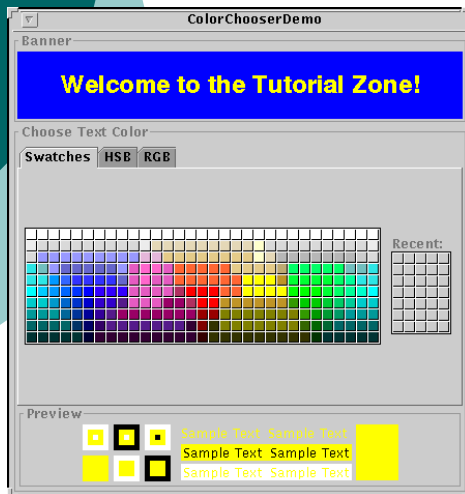


Jcomponent.setToolTipText(String)

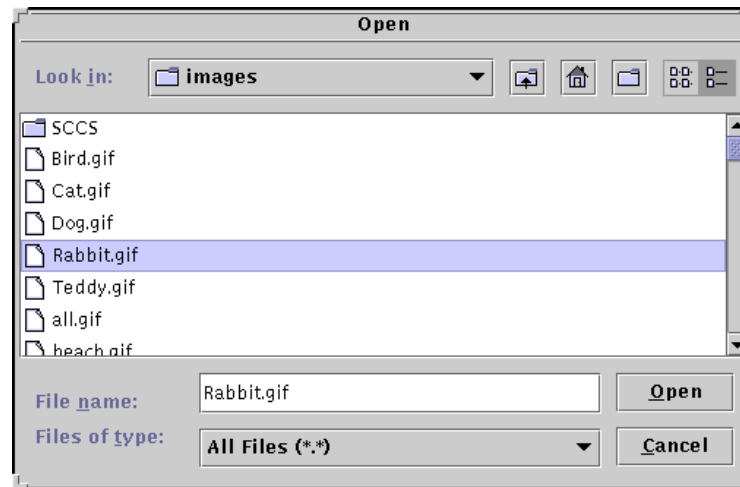


Visualizzatori di informazioni formattate editabili

JColorChooser



JFileChooser



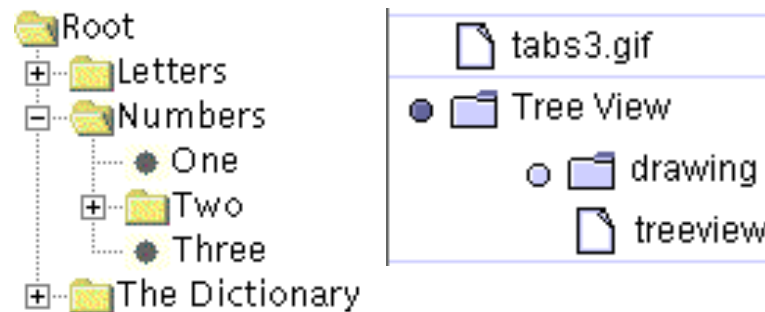
JTable

| TableDemo | | | | |
|------------|-----------|------------------|------------|-------------------------------------|
| First Name | Last Name | Sport | # of Years | Vegetarian |
| Mary | Campione | Snowboarding | 5 | <input type="checkbox"/> |
| Alison | Huml | Rowing | 3 | <input checked="" type="checkbox"/> |
| Kathy | Walrath | Chasing toddl... | 2 | <input type="checkbox"/> |
| Mark | Andrews | Speed reading | 20 | <input checked="" type="checkbox"/> |

JTextComponent

Verify that the RJ45 cable is connected to the WAN plug on the back of the Pipeline unit.

JTree



JTextField,
JPasswordField,
JTextArea, JEditorPane,
JTextPane