



# Interfacce grafiche

---

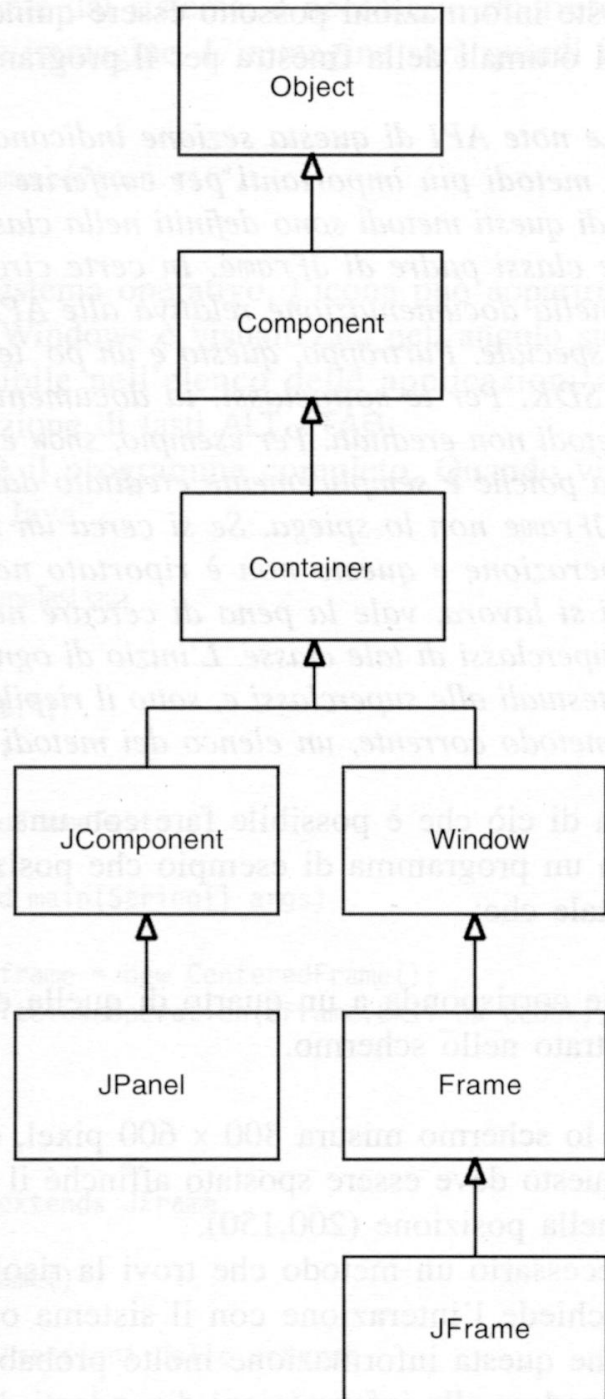


# Ereditarietà per personalizzare i frame

---

- Usare l'ereditarietà per scomporre frames complessi in unità facilmente comprensibili
- Progettare sottoclassi di **JFrame**
- Memorizzare le componenti come variabili di istanza
- Inizializzare le variabili nei costruttori delle sottoclassi
- Se l'inizializzazione diventa complessa utilizza alcuni metodi di servizio

# Gerarchia di ereditarietà



- La classe **JFrame** in sé contiene solo alcuni metodi per modificare l'aspetto dei frame
- Quasi tutti i metodi per lavorare con le dimensioni e con la posizione di un frame provengono dalle varie superclassi di **JFrame**
  - Le classi **Component** e **Window** contengono i metodi per modificare le dimensioni e la forma dei frame

# Esempio: programma visualizzazione investimento

---

- Progettare sottoclassi di **JFrame** e memorizzare componenti nelle variabili di istanza, inizializzandole nel costruttore della vostra sottoclasse
- Usando anche metodi ausiliari se il codice del costruttore diventa troppo complesso
- Ad Esempio:





# Esempio: programma visualizzazione investimento

---

```
01: import java.awt.event.ActionEvent;
02: import java.awt.event.ActionListener;
03: import javax.swing.JButton;
04: import javax.swing.JFrame;
05: import javax.swing.JLabel;
06: import javax.swing.JPanel;
07: import javax.swing.JTextField;
08:
09: /**
10:     This program displays the growth of an investment.
11: */
12: public class InvestmentFrame extends JFrame
13: {
14:     public InvestmentFrame()
15:     {
16:         account = new BankAccount(INITIAL_BALANCE);
17:
```



# Esempio: continuazione

---

```
18:         // Use instance fields for components
19:         resultLabel = new JLabel(
20:             "balance=" + account.getBalance());
21:
22:         // Use helper methods
23:         createRateField();
24:         createButton();
25:         createPanel();
26:
27:         setSize(FRAME_WIDTH, FRAME_HEIGHT);
28:     }
29:
30:     public void createRateField()
31:     {
32:         rateLabel = new JLabel("Interest Rate: ");
33:         final int FIELD_WIDTH = 10;
34:         rateField = new JTextField(FIELD_WIDTH);
```





# Esempio: continuazione

---

```
53:         }
54:     }
55:
56:     ActionListener listener = new AddInterestListener();
57:     button.addActionListener(listener);
58: }
59:
60: public void createPanel()
61: {
62:     JPanel panel = new JPanel();
63:     panel.add(rateLabel);
64:     panel.add(rateField);
65:     panel.add(button);
66:     panel.add(resultLabel);
67:     add(panel);
68: }
69:
```





# Esempio: continuazione

---

```
70:     private JLabel rateLabel;
71:     private JTextField rateField;
72:     private JButton button;
73:     private JLabel resultLabel;
74:     private BankAccount account;
75:
76:     private static final double DEFAULT_RATE = 10;
77:     private static final double INITIAL_BALANCE = 1000;
78:
79:     private static final int FRAME_WIDTH = 500;
80:     private static final int FRAME_HEIGHT = 200;
81: }
```



# Esempio: continuazione

---

Classe con metodo `main`:

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



# Gestione del layout delle componenti

---

- Le componenti di un'interfaccia utente sono organizzate mettendole all'interno di un contenitore
  - Ad esempio **JPanel**
- Ogni contenitore ha un *layout manager* (*gestore di layout*) che si occupa del posizionamento delle sue componenti
  - Le componenti in un **JPanel** sono inserite da sinistra a destra
- Tre gestori di layout più diffusi (**java.awt**):
  - gestore di layout a bordi (**BorderLayout**)
  - gestore di layout a scorrimento (**FlowLayout**)
  - gestore di layout a griglia (**GridLayout**)



# Gestione del layout

---

- Per default, **JPanel** organizza le componenti da sinistra a destra e comincia una nuova riga se necessario
- Il layout di **JPanel** è gestito da **FlowLayout**
  - gestore di layout a scorrimento
- Possiamo richiedere altri gestori di layout

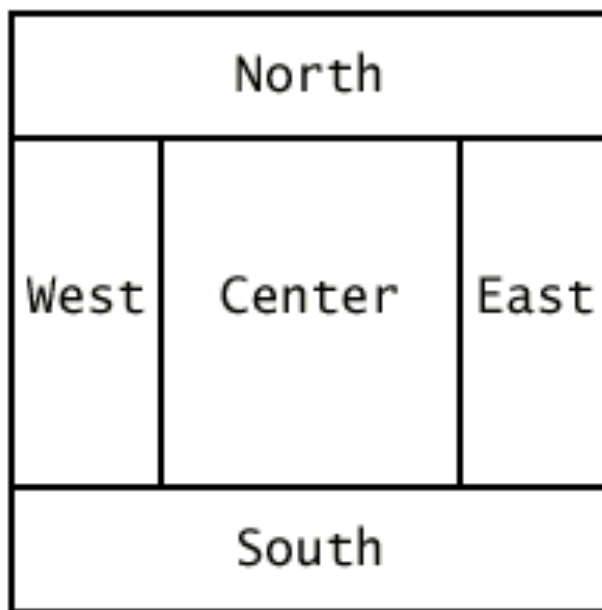
```
panel.setLayout(new BorderLayout());
```



## Layout a bordi

---

- Il contenitore è diviso in 5 aree:  
**center, north, west, south e east**





# BorderLayout

---

- Quando si aggiunge una componente si specifica una posizione:

```
panel.add(component, BorderLayout.NORTH) ;
```

- Ogni componente è estesa per coprire l'intera area allocata. Se si ha l'esigenza di condividere l'area con altri componenti, si possono inserire i componenti in un **JPanel**



## Layout a griglia

---

- Posiziona le componenti in una griglia con un numero fissato di righe e colonne
- La taglia di ogni componente viene opportunamente modificata in modo che tutte le componenti hanno la stessa taglia
- Ogni componente viene espansa in modo che occupi tutta l'area allocata



# GridLayout

---

- Aggiungere le componenti, riga per riga, da sinistra a destra:

```
JPanel numberPanel = new JPanel();  
numberPanel.setLayout(new GridLayout(4, 3));  
numberPanel.add(button7);  
numberPanel.add(button8);  
numberPanel.add(button9);  
numberPanel.add(button4);  
. . .
```



# GridLayout

---





# Altro tipo di layout a griglia

---

- **GridBagLayout:**

- le componenti sono disposte in una tabella
- le colonne possono avere taglie differenti
- le componenti possono ricoprire colonne multiple

- Difficile da usare

- Possiamo ovviare con **JPanel** annidati

- Ogni oggetto **JPanel** ha un gestore appropriato
- Oggetti **JPanel** non hanno bordi visibili
- Si possono usare tanti **JPanel** quanti ne servono per organizzare le componenti

# Scelte

---

- Caselle combinate
- Caselle di controllo
- Pulsanti radio





# Pulsanti radio

---

- In un insieme di pulsanti radio uno solo alla volta può essere selezionato
- Adatto ad un insieme di scelte mutuamente esclusive
- Se un pulsante è selezionato, tutti gli altri nell'insieme sono automaticamente deselezionati
- Ogni pulsante è un oggetto di **JRadioButton**
  - pacchetto **javax.swing**
  - sottoclasse di **JComponent**




# ButtonGroup

---

- Nell'esempio la taglia del font è realizzata con pulsanti radio:

```
JRadioButton smallButton = new JRadioButton("Small");  
JRadioButton mediumButton = new JRadioButton("Medium");  
JRadioButton largeButton = new JRadioButton("Large");  
// Aggiungi pulsanti radio in un ButtonGroup così  
// soltanto un pulsante nel gruppo può essere selezionato  
ButtonGroup group = new ButtonGroup();  
group.add(smallButton);  
group.add(mediumButton);  
group.add(largeButton);
```



# ButtonGroup (javax.swing)

---

- Un gruppo di pulsanti non piazza i pulsanti insieme in un contenitore (non è di tipo **JComponent**), serve solo a stabilire tra quali pulsanti la scelta deve essere mutualmente esclusiva

- **isSelected**: invocato per verificare se un pulsante è selezionato o no

```
if (largeButton.isSelected()) size = LARGE_SIZE;
```

- Prima di visualizzare un frame che contiene pulsanti radio, **setSelected(true)** deve essere invocato su un pulsante di ogni gruppo di pulsanti radio (uno dei pulsanti deve essere selezionato)



# Bordi

---

- Per default, pannelli non hanno bordi visibili
- Può essere utile aggiungere un bordo visibile
- EtchedBorder: un bordo con effetto tri-dimensionale
- Si può aggiungere un bordo a ogni componente:

```
JPanel panel = new JPanel();  
panel.setBorder(new EtchedBorder());
```

- TitledBorder: bordo con titolo

```
panel.setBorder(new TitledBorder(new EtchedBorder(), "Size"));
```



# Caselle di controllo

---

- Ogni casella ha due stati: selezionata e non selezionata
- Nei gruppi di caselle di controllo la scelta non è mutuamente esclusiva
- Esempio: "bold" e "italic" nella figura precedente
- Si costruiscono dandone il nome nel costruttore:

```
JCheckBox italicCheckBox = new JCheckBox("Italic");
```

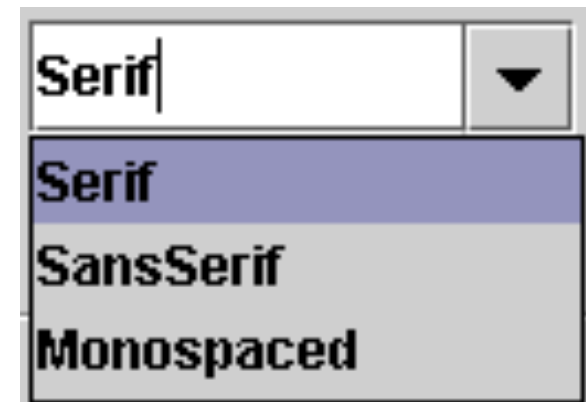
- Non si devono mettere in un gruppo di pulsanti
- **JCheckBox** è nel pacchetto `javax.swing` ed è una sottoclasse di **JComponent**



# Caselle combinate

---

- Per grandi insiemi di scelte mutuamente esclusive
  - usa meno spazio dei pulsanti radio
- Combinazione di una lista e un campo di testo
  - Il campo di testo visualizza il nome della selezione corrente





# Caselle combinate

---

- Se la casella combinata è editabile, allora l'utente può digitare la sua selezione
- Si usa il metodo **setEditable(true)** per rendere editabile il campo di testo
- Si aggiungono le stringhe di testo con il metodo **addItem**:

```
JComboBox facenameCombo = new JComboBox();  
facenameCombo.addItem("Serif");  
facenameCombo.addItem("SansSerif");  
...
```

- **JComboBox** è nel pacchetto **javax.swing** ed è una sottoclasse di **JComponent**



# Caselle combinate

---

- La selezione dell'utente si prende con **getSelectedItem** (restituisce un **Object**)

```
String selectedString =  
    (String) facenameCombo.getSelectedItem();
```

- Seleziona un elemento della lista da visualizzare all'inizio con **setSelectedItem** (**anObject**)



## Nota

---

- Pulsanti radio, caselle di controllo e caselle combinate di un frame generano un **ActionEvent** ogni volta che l'utente seleziona un elemento
- Nel programma che segue:
  - Tutte le componenti notificano l'evento allo stesso listener
  - Quando un utente clicca su una componente, si chiede alla componente il suo contenuto corrente
  - Quindi riscriviamo il testo campione con la scelta corrente

# I componenti di choiceFrame

---

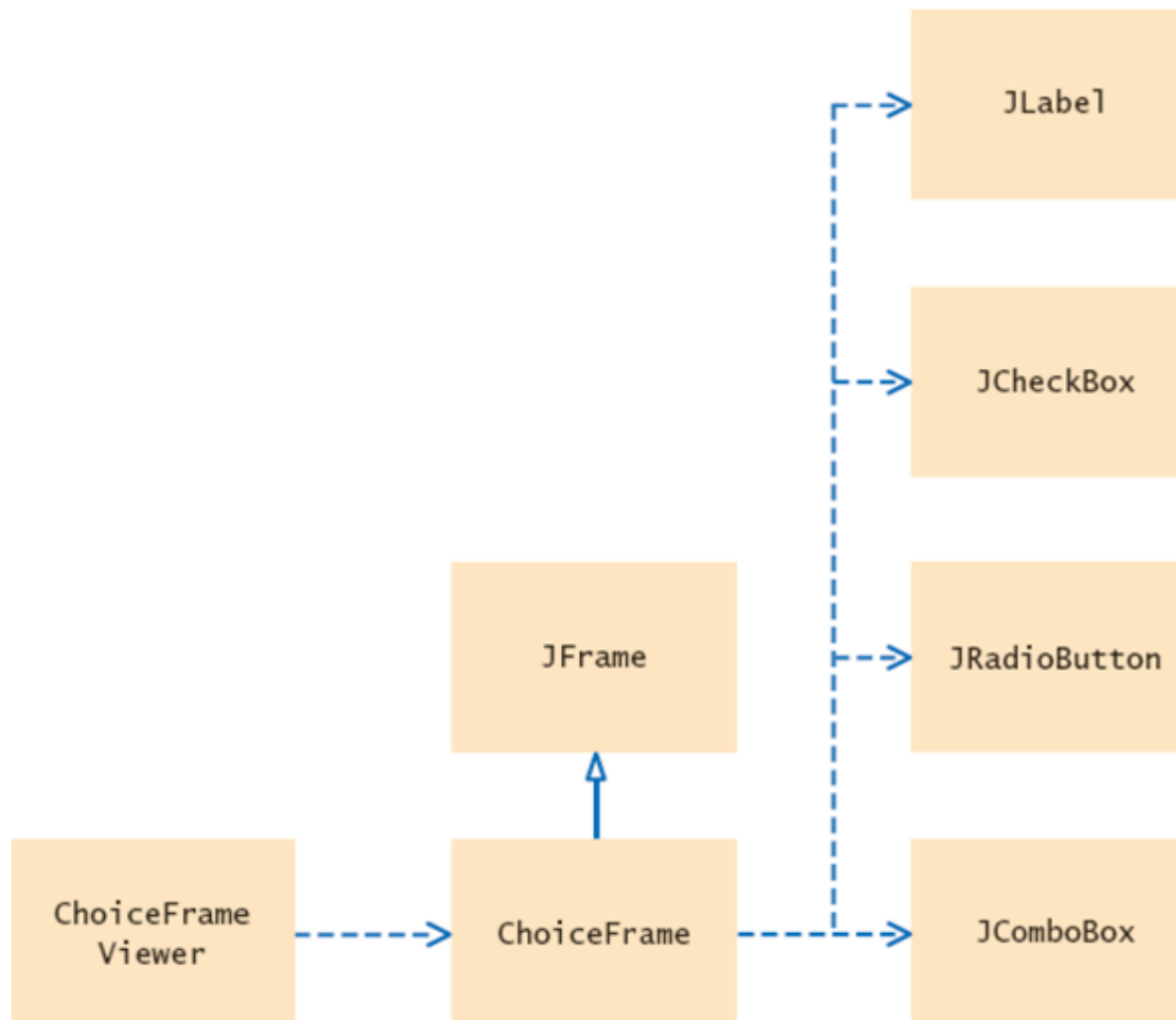
JLabel in posizione  
Center

JPanel in posizione  
South gestito da  
GridLayout



# Classi del programma scelta font

---





# File ChoiceFrameViewer.java

---

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



# File ChoiceFrame.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.JPanel;
013: import javax.swing.JRadioButton;
014: import javax.swing.border.EtchedBorder;
015: import javax.swing.border.TitledBorder;
016:
```





# File ChoiceFrame.java

---

```
017: /**
018:     This frame contains a text field and a control panel
019:     to change the font of the text.
020: */
021: public class ChoiceFrame extends JFrame
022: {
023:     /**
024:         Constructs the frame.
025:     */
026:     public ChoiceFrame()
027:     {
028:         // Construct text sample
029:         sampleField = new JLabel("Big Java");
030:         add(sampleField, BorderLayout.CENTER);
031:
```



# File ChoiceFrame.java

---

```
032:         // This listener is shared among all components
033:         class ChoiceListener implements ActionListener
034:         {
035:             public void actionPerformed(ActionEvent event)
036:             {
037:                 setSampleFont();
038:             }
039:         }
040:
041:         listener = new ChoiceListener();
042:
043:         createControlPanel();
044:         setSampleFont();
045:         setSize(FRAME_WIDTH, FRAME_HEIGHT);
046:     }
047:
```



# File ChoiceFrame.java

---

```
048:    /**
049:        Creates the control panel to change the font.
050:    */
051:    public void createControlPanel()
052:    {
053:        JPanel facenamePanel = createComboBox();
054:        JPanel sizeGroupPanel = createCheckBoxes();
055:        JPanel styleGroupPanel = createRadioButtons();
056:
057:        // Line up component panels
058:
059:        JPanel controlPanel = new JPanel();
060:        controlPanel.setLayout(new GridLayout(3, 1));
061:        controlPanel.add(facenamePanel);
062:        controlPanel.add(sizeGroupPanel);
063:        controlPanel.add(styleGroupPanel);
064:
```



# File ChoiceFrame.java

---

```
065:         // Add panels to content pane
066:
067:         add(controlPanel, BorderLayout.SOUTH);
068:     }
069:
070:     /**
071:      * Creates the combo box with the font style choices.
072:      * @return the panel containing the combo box
073:      */
074:     public JPanel createComboBox()
075:     {
076:         facenameCombo = new JComboBox();
077:         facenameCombo.addItem("Serif");
078:         facenameCombo.addItem("SansSerif");
079:         facenameCombo.addItem("Monospaced");
080:         facenameCombo.setEditable(true);
081:         facenameCombo.addActionListener(listener);
082:
```



# File ChoiceFrame.java

---

```
083:         JPanel panel = new JPanel();
084:         panel.add(facenameCombo);
085:         return panel;
086:     }
087:
088:     /**
089:         Creates the check boxes for selecting bold and
            // italic styles.
090:         @return the panel containing the check boxes
091:     */
092:     public JPanel createCheckBoxes()
093:     {
094:         italicCheckBox = new JCheckBox("Italic");
095:         italicCheckBox.addActionListener(listener);
096:
097:         boldCheckBox = new JCheckBox("Bold");
098:         boldCheckBox.addActionListener(listener);
099:
```



# File ChoiceFrame.java

---

```
100:         JPanel panel = new JPanel();
101:         panel.add(italicCheckBox);
102:         panel.add(boldCheckBox);
103:         panel.setBorder
104:             (new TitledBorder(new EtchedBorder(), "Style"));
105:
106:         return panel;
107:     }
108:
109:     /**
110:         Creates the radio buttons to select the font size
111:         @return the panel containing the radio buttons
112:     */
113:     public JPanel createRadioButtons()
114:     {
115:         smallButton = new JRadioButton("Small");
116:         smallButton.addActionListener(listener);
```



# File ChoiceFrame.java

---

```
117:
118:     mediumButton = new JRadioButton("Medium");
119:     mediumButton.addActionListener(listener);
120:
121:     largeButton = new JRadioButton("Large");
122:     largeButton.addActionListener(listener);
123:     largeButton.setSelected(true);
124:
125:     // Add radio buttons to button group
126:
127:     ButtonGroup group = new ButtonGroup();
128:     group.add(smallButton);
129:     group.add(mediumButton);
130:     group.add(largeButton);
131:
```



# File ChoiceFrame.java

---

```
132:         JPanel panel = new JPanel();
133:         panel.add(smallButton);
134:         panel.add(mediumButton);
135:         panel.add(largeButton);
136:         panel.setBorder
137:             (new TitledBorder(new EtchedBorder(), "Size"));
138:
139:         return panel;
140:     }
141:
142:     /**
143:         Gets user choice for font name, style, and size
144:         and sets the font of the text sample.
145:     */
146:     public void setSampleFont()
147:     {
```





# File ChoiceFrame.java

---

```
148:         // Get font name
149:         String facename
150:             = (String) facenameCombo.getSelectedItem();
151:
152:         // Get font style
153:
154:         int style = 0;
155:         if (italicCheckBox.isSelected())
156:             style = style + Font.ITALIC;
157:         if (boldCheckBox.isSelected())
158:             style = style + Font.BOLD;
159:
160:         // Get font size
161:
162:         int size = 0;
163:
```



# File ChoiceFrame.java

---

```
164:         final int SMALL_SIZE = 24;
165:         final int MEDIUM_SIZE = 36;
166:         final int LARGE_SIZE = 48;
167:
168:         if (smallButton.isSelected())
169:             size = SMALL_SIZE;
170:         else if (mediumButton.isSelected())
171:             size = MEDIUM_SIZE;
172:         else if (largeButton.isSelected())
173:             size = LARGE_SIZE;
174:
175:         // Set font of text field
176:
177:         sampleField.setFont(new Font(facename, style, size));
178:         sampleField.repaint();
179:     }
```



# File ChoiceFrame.java

---

```
180:
181:     private JLabel sampleField;
182:     private JCheckBox italicCheckBox;
183:     private JCheckBox boldCheckBox;
184:     private JRadioButton smallButton;
185:     private JRadioButton mediumButton;
186:     private JRadioButton largeButton;
187:     private JComboBox facenameCombo;
188:     private ActionListener listener;
189:
190:     private static final int FRAME_WIDTH = 300;
191:     private static final int FRAME_HEIGHT = 400;
192: }
```

# Osservazioni sulla gestione del layout

---

- Passo 1: Annotare su un foglio il layout delle componenti desiderate

Size

☒ Small

☐ Medium

☐ Large

☒ Pepperoni

☒ Anchovies

Your Price:

# Osservazioni sulla gestione del layout

---

- Passo 2: Raggruppare componenti adiacenti con lo stesso layout

The diagram illustrates a pizza order form layout. It features a purple border enclosing the entire form. Inside, there are three red rectangular boxes highlighting adjacent components:

- Size Selection:** A box containing the label "Size" followed by three radio button options: "Small", "Medium", and "Large".
- Toppings Selection:** A box containing two checked checkbox options: "Pepperoni" and "Anchovies".
- Price Input:** A box containing the label "Your Price:" followed by an empty text input field.



# Osservazioni sulla gestione del layout

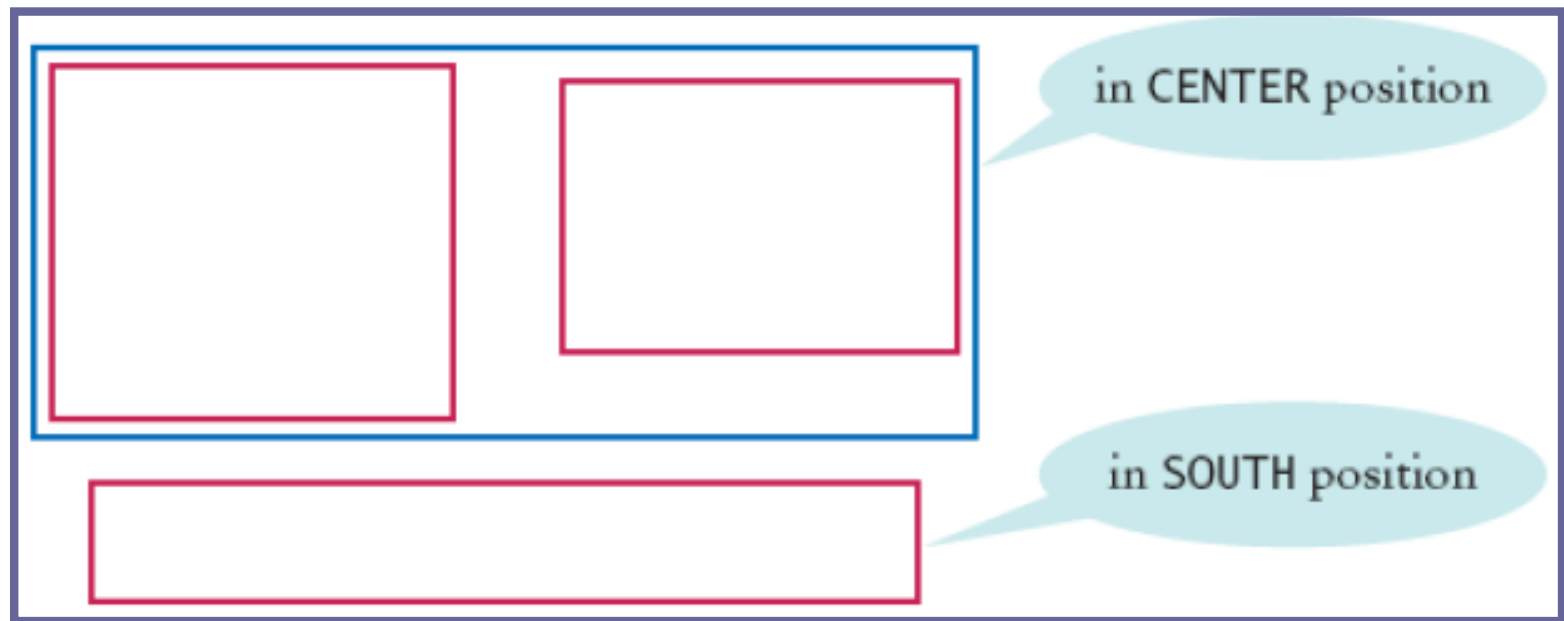
---

- Passo 3: determina un layout per ogni gruppo (i.e., identificare i gestori della disposizione per ciascun gruppo)
  - Quando i componenti sono disposti orizzontalmente, scegliete un gestore a scorrimento
  - Quando i componenti sono disposti verticalmente, usate un gestore a griglia, che abbia tante righe quanti sono i componenti e una sola colonna
- Nell'esempio
  - Un gestore a griglia (3,1) per i pulsanti radio
  - Un gestore a griglia (2,1) per le caselle di testo
  - Un gestore a scorrimento per l'etichetta e il campo di testo

# Osservazioni sulla gestione del layout

---

- Passo 4: raggruppa i gruppi



Avete terminato quando tutti i gruppi si trovano in un unico contenitore

- Passo 5: Scrivi il codice per generare il layout



```
JPanel radioButtonPanel = new JPanel();
radioButtonPanel.setLayout(new GrdiLayout(3,1));
radioButtonPanel.setBorder(new TitleBorder(new
                                EtcheBorder(),"Size"));

radioButtonPanel.add(smallButton);
radioButtonPanel.add(mediumButton);
radioButtonPanel.add(largeButton);

JPanel checkBoxPanel = new JPanel();
checkBoxPanel.setLayout(new GrdiLayout(2,1));
checkBoxPanel.add(pepperoniButton);
checkBoxPanel.add(anchovies);

JPanel pricePanel = new JPanel(); // usa FlowLayout
pricePanel.add(new JLabel("Your Price:"));
pricePanel.add(priceTextField);

JPanel centerPanel = new JPanel(); // usa FlowLayout
centerPanel.add(radioButtonPanel);
centerPanel.add(checkBoxPanel);

// il frame è gestito in modo predefinito da un BorderLayout
Add(centerPanel, BorderLayout.CENTER)
Add(pricePanel, BorderLayout.SOUTH)
```