Swing

Java e la grafica

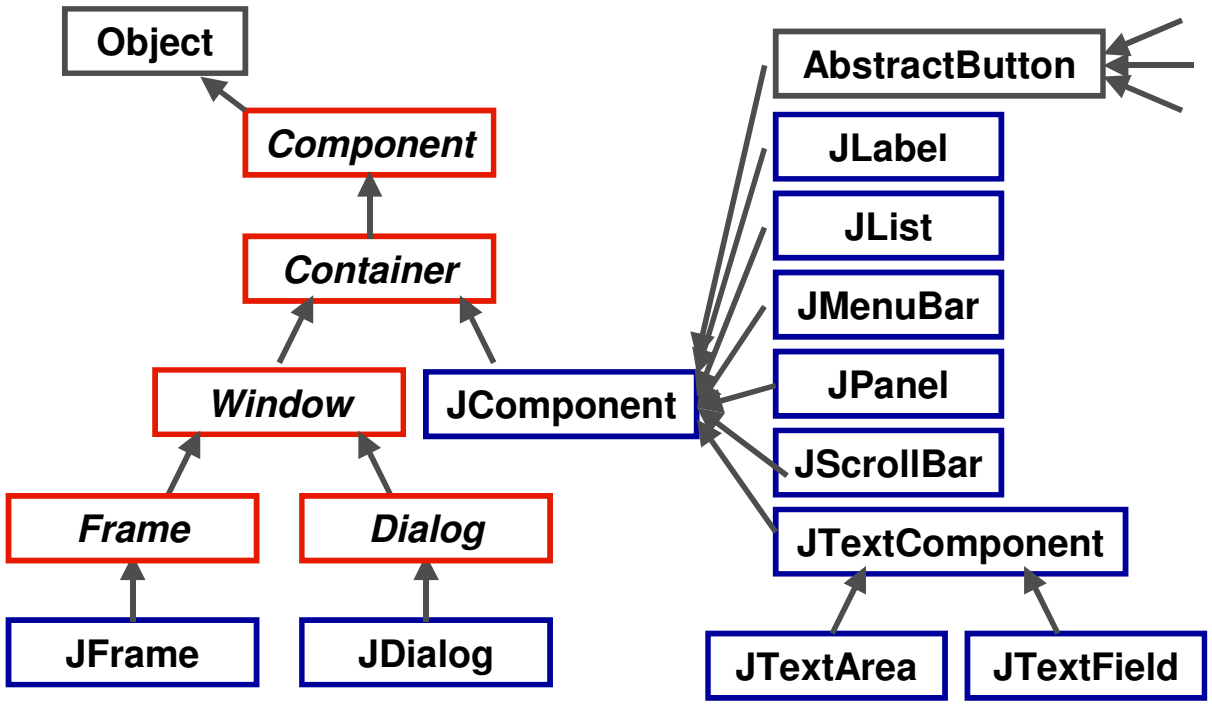
Java permette di realizzare agevolmente applicazioni grafiche

- Package **java.awt**
  - il primo package grafico (Java 1.0)
  - indipendente dalla piattaforma... o quasi!
- Package **javax.swing**
  - il nuovo package grafico (da Java 1.2)
  - scritto interamente in Java, realmente indipendente dalla piattaforma

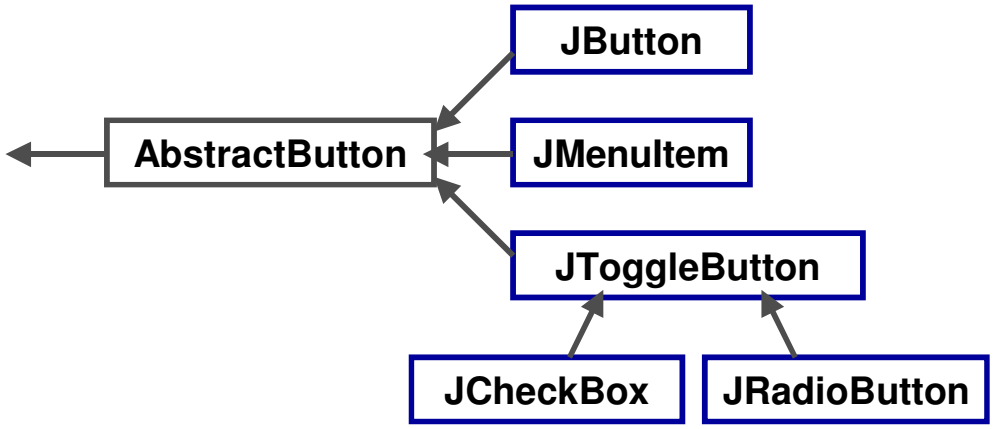
Swing: Architettura

- Swing definisce una gerarchia di classi che forniscono ogni tipo di componente grafico
  - finestre, pannelli, frame, bottoni, aree di testo, checkbox, liste a discesa, ecc. ecc.
- Programmazione **event-driven**:
  - Il programma reagisce agli eventi che l'utente, in modo interattivo, genera sui componenti grafici
- Concetti di **evento** e di **ascoltatore degli eventi**

Swing – Gerarchia di classi



Swing – I bottoni

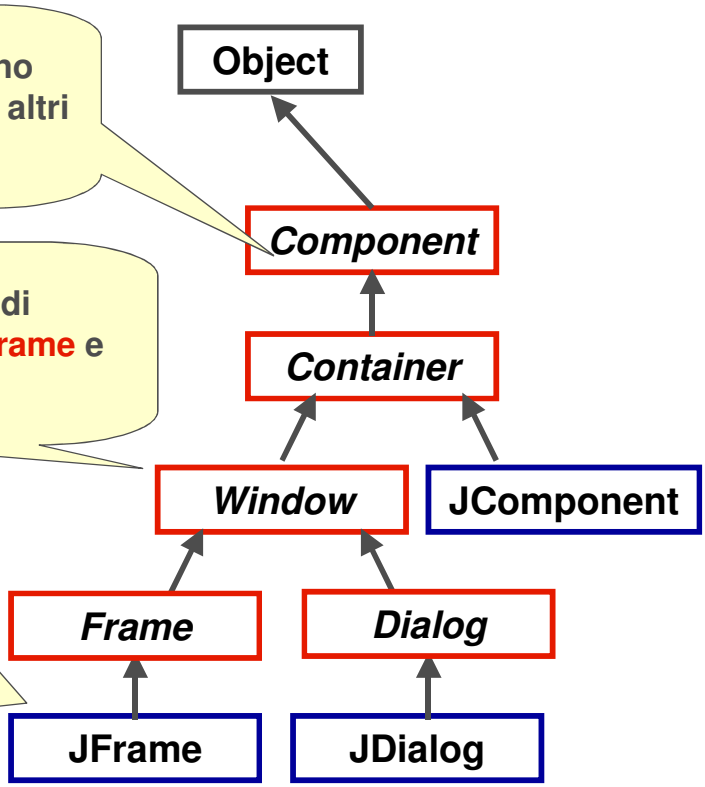


Swing – Contenitori e finestre

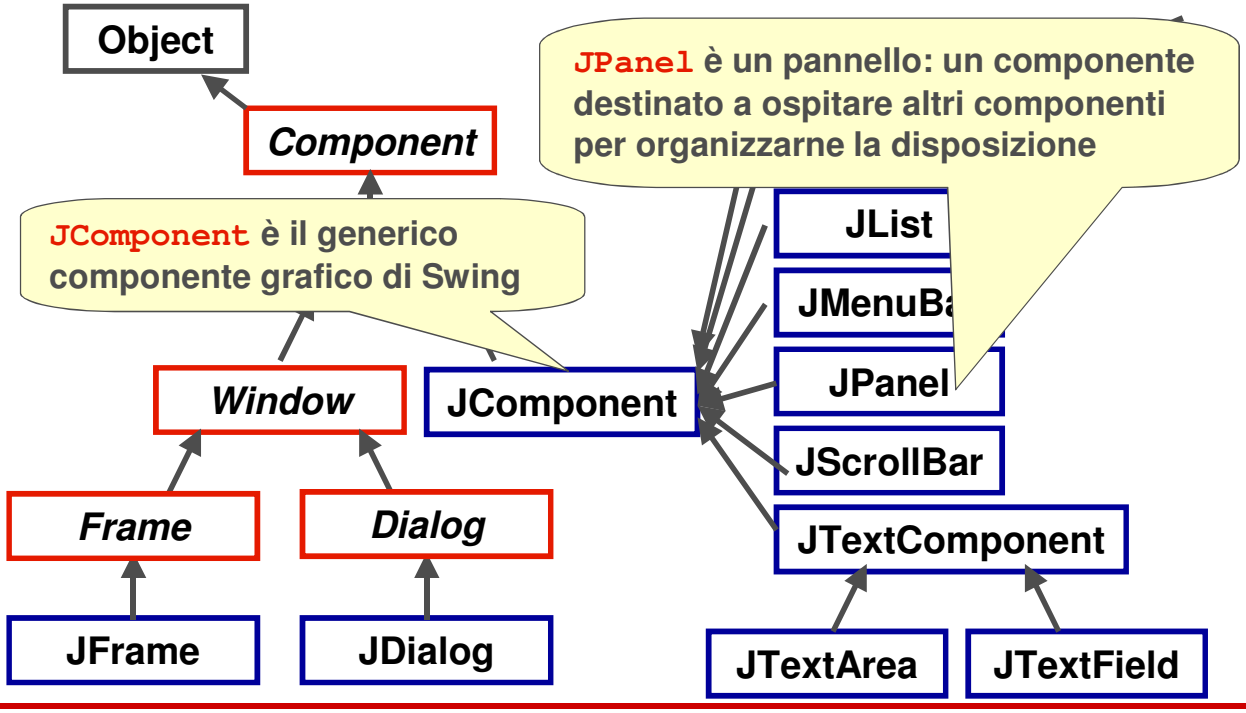
Tutti i componenti principali sono **contenitori**: possono contenere altri componenti

Le finestre sono casi particolari di contenitori e si distinguono in **Frame** e **Finestre di Dialogo**

Il `JFrame` è il componente "finestra principale": ha un aspetto grafico, una cornice (ridimensionabile), un titolo



Swing – Componenti e pannelli



Swing: un esempio

- La più semplice applicazione grafica consiste in una classe il cui main crea un **JFrame** e lo rende visibile con **show()**:

```
import java.awt.*;
import javax.swing.*;
public class EsSwing1
{
 public static void main(String[] v)
 {
   JFrame f = new JFrame("Esempio 1");
   f.show();
 }
}
```

Crea un nuovo JFrame, **inizialmente invisibile**, con il titolo specificato

Swing: un esempio

- La più semplice applicazione grafica consiste in una classe il cui main crea un JFram

I comandi standard sono già attivi (la chiusura per default nasconde il frame senza chiuderlo realmente)

```
import
import javax.swing.*;
public class EsSwing1
{
 public static vo                    v)
 {
   JFrame f = new                    1");
   f.show();
 }
}
```

Swing: un esempio

- La finestra che così nasce ha però dimensioni nulle (bisogna allargarla "a mano")
- Per impostare le dimensioni di un qualunque contenitore si usa setSize:

```
f.setSize(300,150);
```

Larghezza (x), Altezza (y)
Le misure sono in pixel (tutto lo schermo = 800x600, 1024x768, etc)

Swing: un esempio

- Inoltre, la finestra viene visualizzata nell'angolo superiore sinistro dello schermo
- Per impostare la posizione di un qualunque contenitore si usa **setLocation()**:

```
f.setLocation(200,100);
```

Ascissa, Ordinata (in pixel)
Origine (0,0) = angolo superiore sinistro

- Posizione e dimensioni si possono anche fissare insieme, col metodo **setBounds()**

Swing: un esempio

- Quando si chiude la finestra il comportamento standard del JFrame è quello di chiudere la finestra senza terminare l'applicazione
- Il metodo **setDefaultCloseOperation()** permette di cambiare questo comportamento
- L'opzione da indicare è EXIT_ON_CLOSE:

```
f.setDefaultCloseOperation(EXIT_ON_CLOSE);
```

Swing: un esempio

- Un esempio di finestra già dimensionata e collocata nel punto previsto dello schermo:

```
import java.awt.*;
import javax.swing.*;
public class EsSwing1
{
 public static void main(String[] v)
 {
   JFrame f = new JFrame("Esempio 1");
   f.setDefaultCloseOperation(EXIT_ON_CLOSE);
   f.setBounds(200,100,300,150)
   f.show();
 }
}
```

Posizione iniziale: x = 200 y = 100
Larghezza = 300, Altezza = 150

Personalizzare il JFrame

- Un approccio efficace consiste nell'estendere JFrame, definendo una nuova classe:

```
public class MyFrame extends JFrame
{
 public MyFrame()
 {
   super(); setBounds(200,100,300,150);
 }
 public MyFrame(String titolo)
 {
   super(titolo);
   setDefaultCloseOperation(EXIT_ON_CLOSE);
   setBounds(200,100,300,150);
 }
}
```

Un nuovo esempio

Questo esempio usa un MyFrame:

```
import java.awt.*;
import javax.swing.*;
public class EsSwing2
{
 public static void main(String[] v)
 {
   MyFrame f = new MyFrame("Esempio 2");
   f.show();
 }
}
```


Struttura del Frame

- In Swing non si possono aggiungere nuovi componenti direttamente sul JFrame
- Dentro a ogni JFrame c'è un **Container**, recuperabile col metodo **getContentPane()**
- I componenti vanno messi sul container
- Tipicamente, si aggiunge un pannello (un JPanel o un suo discendente specifico), tramite il metodo add()
- Sul pannello possiamo quindi aggiungere pulsanti, etichette, icone…

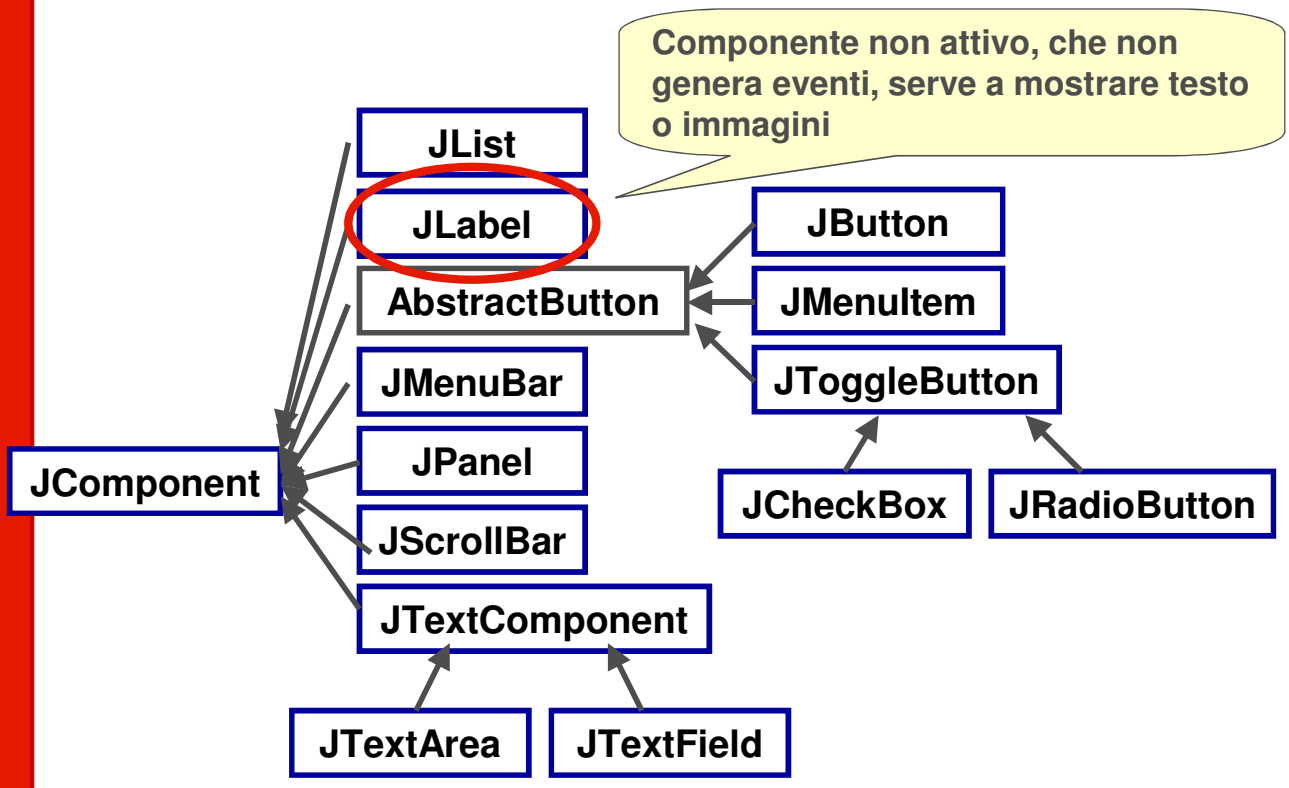
Esempio 3

Aggiunta di un pannello al Container di un frame, tramite l'uso di getContentPane()

```
import java.awt.*; import javax.swing.*;
public class EsSwing3
{
 public static void main(String[] v)
 {
   MyFrame f = new MyFrame("Esempio 3");
   Container c = f.getContentPane();
   JPanel panel = new JPanel();
   c.add(panel);
   f.show();
 }
}
```

Ora che abbiamo un pannello, possiamo usarlo per disegnare e per metterci altri componenti!

Componenti Swing: JLabel



Esempio: uso di JLabel

Il solito main:

```
import java.awt.*; import javax.swing.*;
public class EsSwing7
{
 public static void main(String[] v)
 {
  JFrame f = new JFrame(”Esempio 7");
  Container c = f.getContentPane();
  Es7Panel p = new Es7Panel();
  c.add(p);
  f.pack(); f.show();
 }
}
```

Il metodo `pack()` dimensiona il frame in modo da contenere esattamente il pannello dato

Esempio: uso di JLabel

```
public class Es7Panel extends JPanel
{
 public Es7Panel()
 {
  super();
  JLabel lb1 = new JLabel("Etichetta");
  add(lb1);
 }
}
```

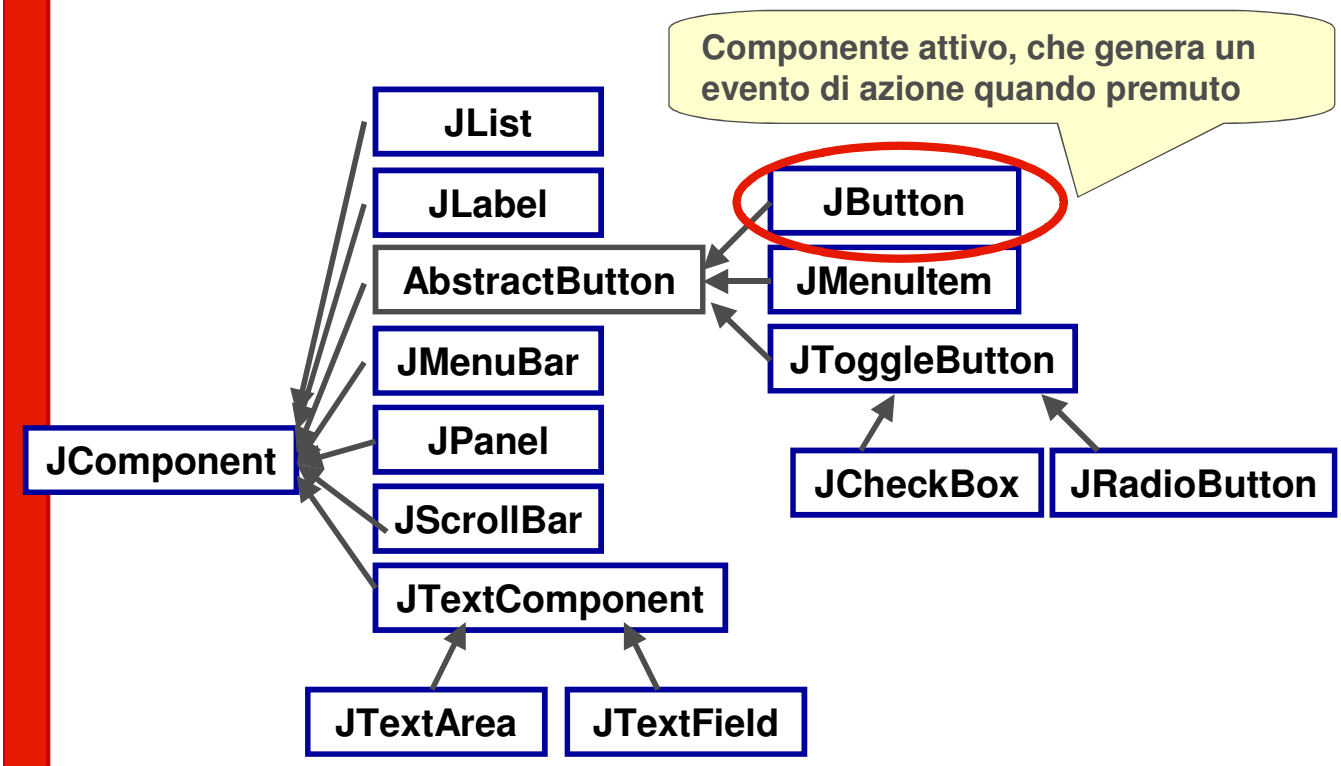
Variante: JLabel con Icona

```
public class Es7Panel extends JPanel
{
 public Es7Panel()
 {
  super();
  JLabel lb2 =
    new JLabel(new ImageIcon("image.gif"));
  add(lb2);
 }
}
```

Interattività

- La costruzione di interfacce grafiche richiede **interattività**
- L'utente deve poter premere bottoni, scrivere testo, scegliere elementi da liste, etc etc
- Si usano componenti attivi, che generano eventi

Componenti Swing: JButton

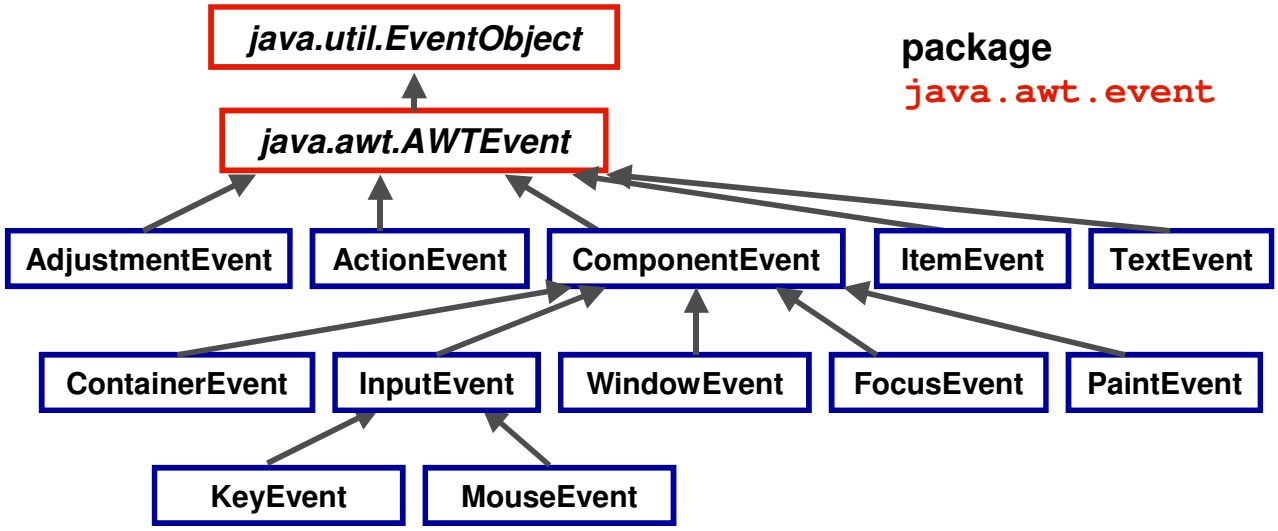


Eventi

- Ogni componente grafico, quando si opera su di esso, genera un **evento** che descrive cosa è accaduto
- Tipicamente, ogni componente può generare molti tipi diversi di eventi, in relazione a ciò che sta accadendo:
  - un bottone può generare l'evento "azione" che significa che è stato premuto
  - una casella di opzione può generare l'evento "stato modificato" per dire che la casella è stata selezionata / deselezionata

Eventi in Java

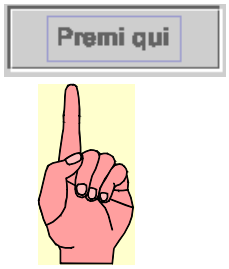
- In Java, un evento è un oggetto, istanza di **java.util.EventObject** o di una sua sottoclasse



Gestione degli eventi

Ogni componente viene associato a un ***ascoltatore degli eventi*** (un oggetto che implementa l'opportuna interfaccia `Listener`)

L'ascoltatore gestisce l'evento



Event Listener

Quando si agisce sul componente (ad es., si preme il pulsante) **si ha un evento, che è inviato all'ascoltatore**

Gestione degli eventi

- Quando si interagisce con un componente "attivo" si genera un **evento**, che è un oggetto Event della sottoclasse opportuna
- L'oggetto Event contiene tutte le informazioni sull'evento (chi l'ha creato, cosa è successo, etc)
- Il sistema invia tale "oggetto Evento" all'oggetto ascoltatore degli eventi preventivamente registrato come tale, che gestisce l'evento.
- L'attività non è più algoritmica (input / computazione / output), ma **interattiva** e **reattiva**

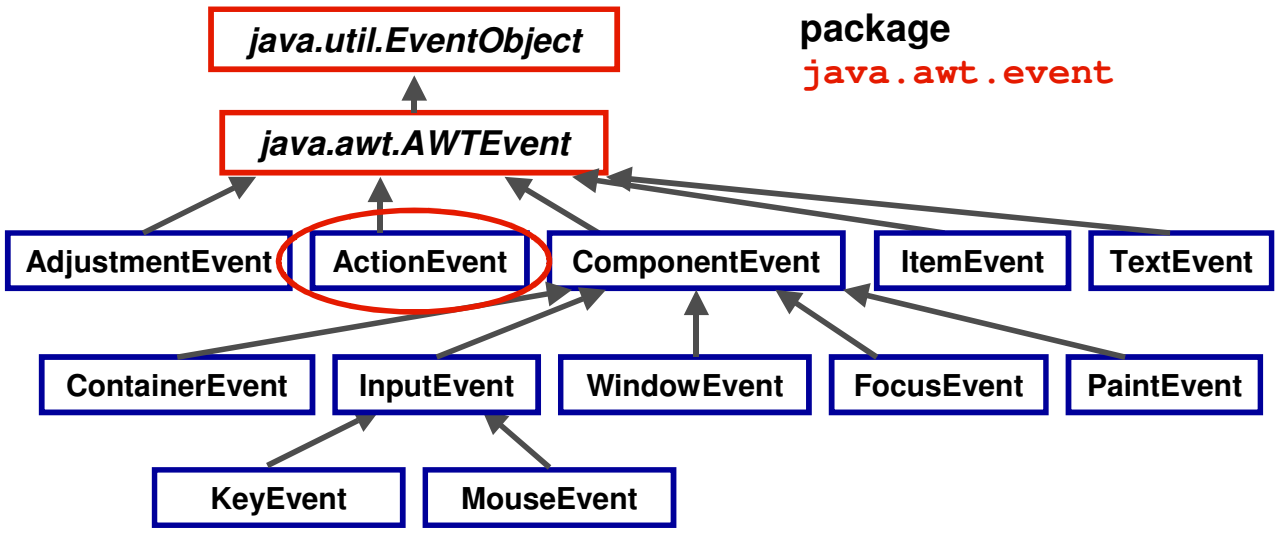
Il pulsante JButton

- Quando viene premuto, un bottone genera un evento di classe **ActionEvent**
- Questo evento viene inviato dal sistema allo specifico ascoltatore degli eventi, di classe ActionListener, registrato per quel bottone
  - può essere un oggetto di un'altra classe…
  - .. o anche il pannello stesso (this)
- Tale ascoltatore degli eventi deve implementare il metodo

```
void actionPerformed(ActionEvent ev);
```

ActionEvent

Un bottone premuto genera un ActionEvent



Esempio: uso di JButton - 1

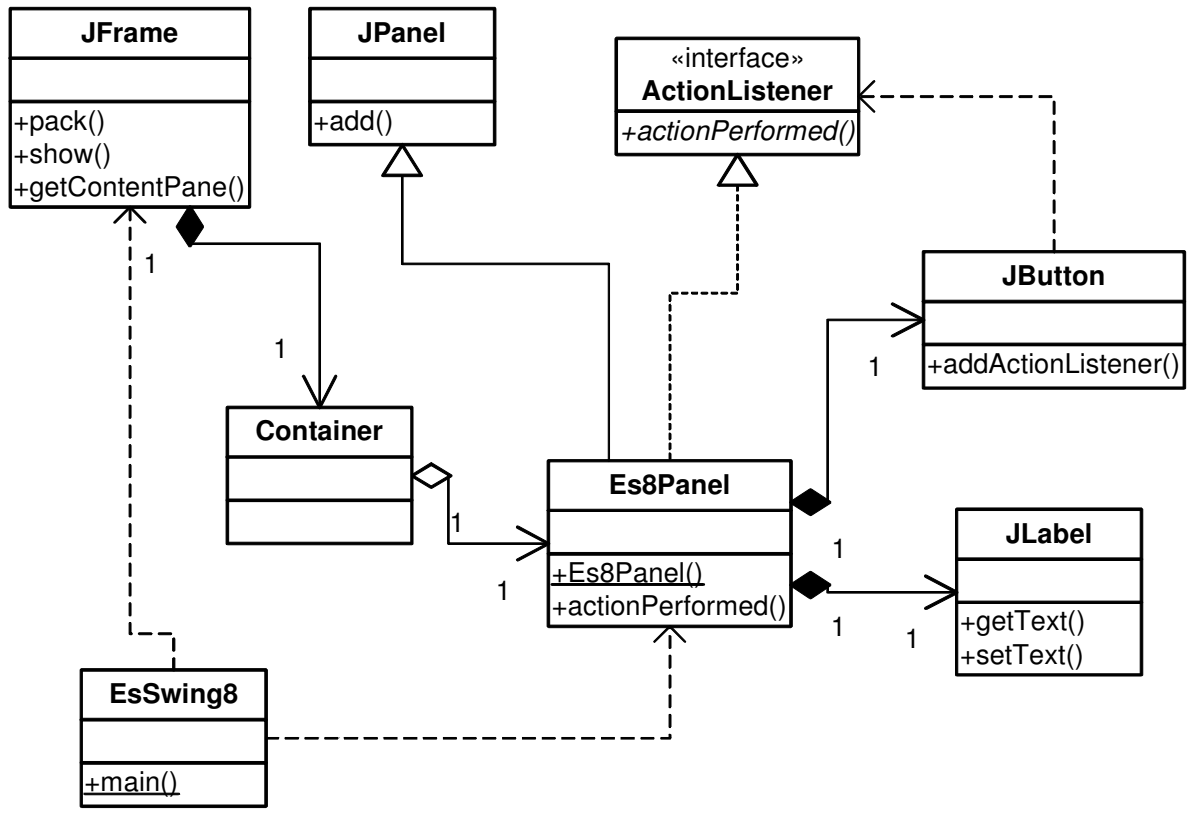
- Un'applicazione fatta da un'etichetta (JLabel) e un pulsante (JButton)
- L'etichetta può mostrare la scritta "Tizio" o "Caio"
- All'inizio vale "Tizio"
- Premendo il bottone, l'etichetta deve commutare, diventando "Caio" se era "Tizio", o "Tizio" se era "Caio"

Esempio: uso di JButton - 2

Architettura dell'applicazione:

- Un pannello (**sottoclasse di JPanel**) contiene un'etichetta e un pulsante
- Il costruttore del pannello crea l'etichetta (**JLabel**) e il pulsante **(JButton)**
- Il pannello fa da **ascoltatore degli eventi** per il pulsante
- Il costruttore del pannello imposta il pannello stesso come ascoltatore degli eventi del pulsante

Esempio: uso di JButton - 3



Esempio: uso di JButton - 4

Eventi da gestire:

- l'evento di azione sul pulsante deve provocare il cambio del testo dell'etichetta

Come si fa?

- il testo dell'etichetta si può recuperare con getText() e cambiare con setText()
- l'ascoltatore dell'evento, che implementa il metodo ActionPerformed(), deve recuperare il testo dell'etichetta e cambiarlo

Esempio: uso di JButton - 5

```
public class Es8Panel extends JPanel
   implements ActionListener
{
 private JLabel l;
 public Es8Panel()
 {
  super();
  l = new JLabel("Tizio");
  add(l);
  JButton b = new JButton("Tizio/Caio");
  b.addActionListener(this);
  add(b);
 }
 ...
```

Per fungere da ascoltatore degli eventi di azione, deve implementare l'interfaccia `ActionListener`

Registra se stesso (`this`) come ascoltatore degli eventi generati dal pulsante `b`

Esempio: uso di JButton - 6

```
...
public void actionPerformed(ActionEvent e)
{
 if (l.getText().equals("Tizio"))
    l.setText("Caio");
 else
    l.setText("Tizio");
}
}
```

Esempio: uso di JButton - Main

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class EsSwing8
{
 public static void main(String[] v)
 {
  JFrame f = new JFrame("Esempio 7");
  Container c = f.getContentPane();
  Es8Panel p = new Es8Panel();
  c.add(p);
  f.pack(); f.show();
 }
}
```

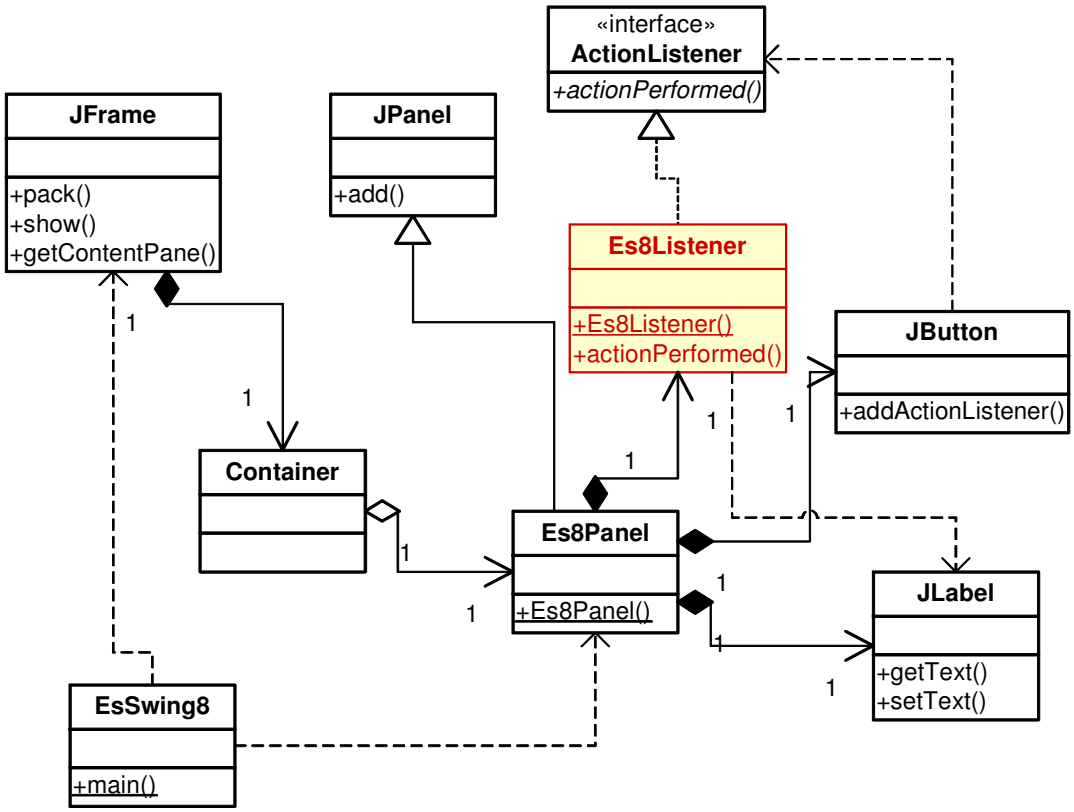
E' necessario importare il package `java.awt.event`

Una variante - 1

Architettura dell'applicazione:

- Un pannello che contiene etichetta e pulsante
- Il costruttore del pannello crea l'etichetta e il pulsante
- **L'ascoltatore degli eventi per il pulsante è un oggetto separato**
- Il costruttore del pannello imposta tale oggetto come ascoltatore degli eventi del pulsante

Una variante - 2



Una variante - 3

L'ascoltatore degli eventi:

```
class Es8Listener implements ActionListener
{
 private JLabel l;
 public void actionPerformed(ActionEvent e)
 {
   if (l.getText().equals("Tizio"))
     l.setText("Caio");
   else l.setText("Tizio");
 }
 public Es8Listener(JLabel label)
 { l=label;}
}
```

L'ascoltatore deve farsi dare come parametro, nel costruttore, la `JLabel` su cui dovrà agire

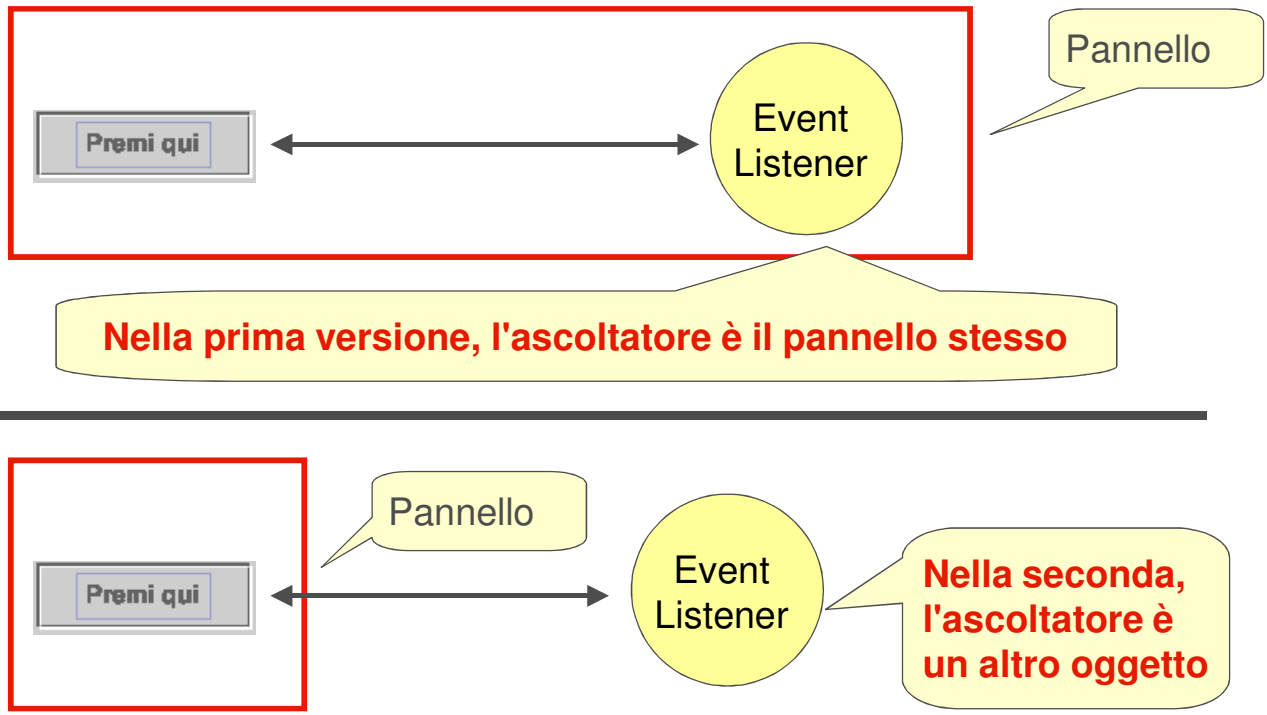
Una variante - 4

Il pannello:

```
public class Es8Panel extends JPanel
{
 public Es8Panel()
 {
  super();
  JLabel l = new JLabel("Tizio");
  add(l);
  JButton b = new JButton("Tizio/Caio");
  b.addActionListener(new Es8Listener(l));
  add(b);
 }
}
```

Crea un oggetto `Es8Listener` e lo imposta come ascoltatore degli eventi per il pulsante `b`

Confronto



Esempio con due pulsanti

Scopo dell'applicazione:

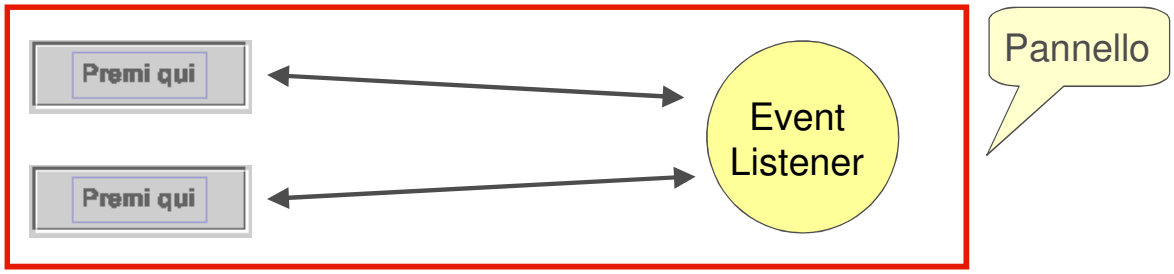
- Cambiare il colore di sfondo tramite due pulsanti: uno lo rende rosso, l'altro azzurro

Architettura dell'applicazione:

- Un pannello che contiene i due pulsanti creati dal costruttore del pannello
- Un unico ascoltatore degli eventi per entrambi i pulsanti
- Abbiamo la necessità di capire, in **actionPerformed()**, quale pulsante è stato premuto

Esempio con due pulsanti

Versione con un unico ascoltatore per entrambi i pulsanti



Il metodo **`actionPerformed()`** dell'ascoltatore dovrà *discriminare quale pulsante* ha generato l'evento

Esempio con due pulsanti

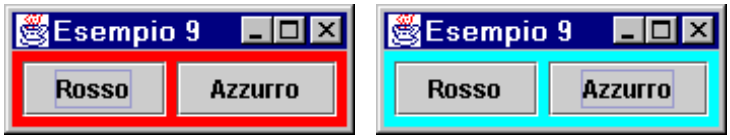
```
public class Es9Panel extends JPanel
   implements ActionListener

{
 JButton b1, b2;
 public Es9Panel()
 {
  super();
  b1 = new JButton("Rosso");
  b2 = new JButton("Azzurro");
  b1.addActionListener(this);
  b2.addActionListener(this);
  add(b1);
  add(b2);
 }
 ...
```

Il pannello fa da ascoltatore degli eventi per entrambi i pulsanti

Esempio con due pulsanti

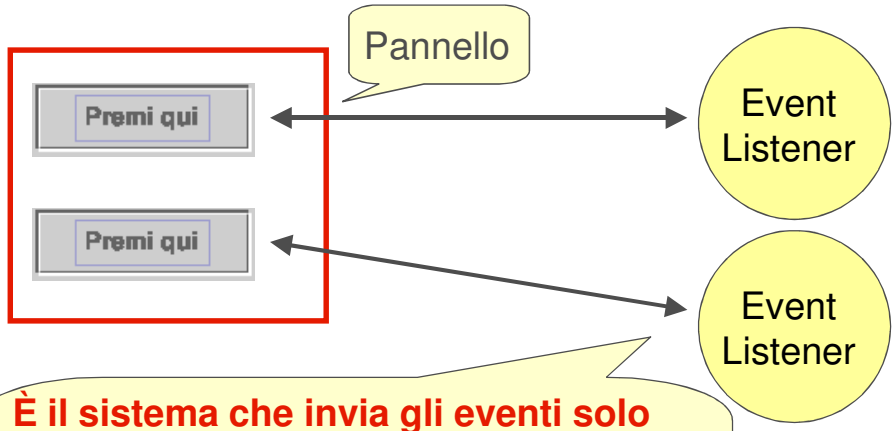
```
...
public void actionPerformed(ActionEvent e)
{
   Object pulsantePremuto = e.getSource();
   if (pulsantePremuto==b1)
      setBackground(Color.red);
   if (pulsantePremuto==b2)
      setBackground(Color.cyan);
}
}
```

Occorre controllare l'identità dell'oggetto che ha generato l'evento



Esempio con 2 bottoni: variante

Un ascoltatore per ciascun pulsante



È il sistema che invia gli eventi solo all'ascoltatore opportuno!

Il metodo actionPerformed() non deve più preoccuparsi di chi è stato premuto.

Esempio con 2 pulsanti: variante

```
class Es9PanelBis extends JPanel
{
 public Es9PanelBis()
 {
  super();
  JButton b1 = new JButton("Rosso");
  JButton b2 = new JButton("Azzurro");
  b1.addActionListener(
    new Es9Listener(this,Color.red) );
  b2.addActionListener(
    new Es9Listener(this,Color.cyan) );
  add(b1);
  add(b2);
 }
}
```

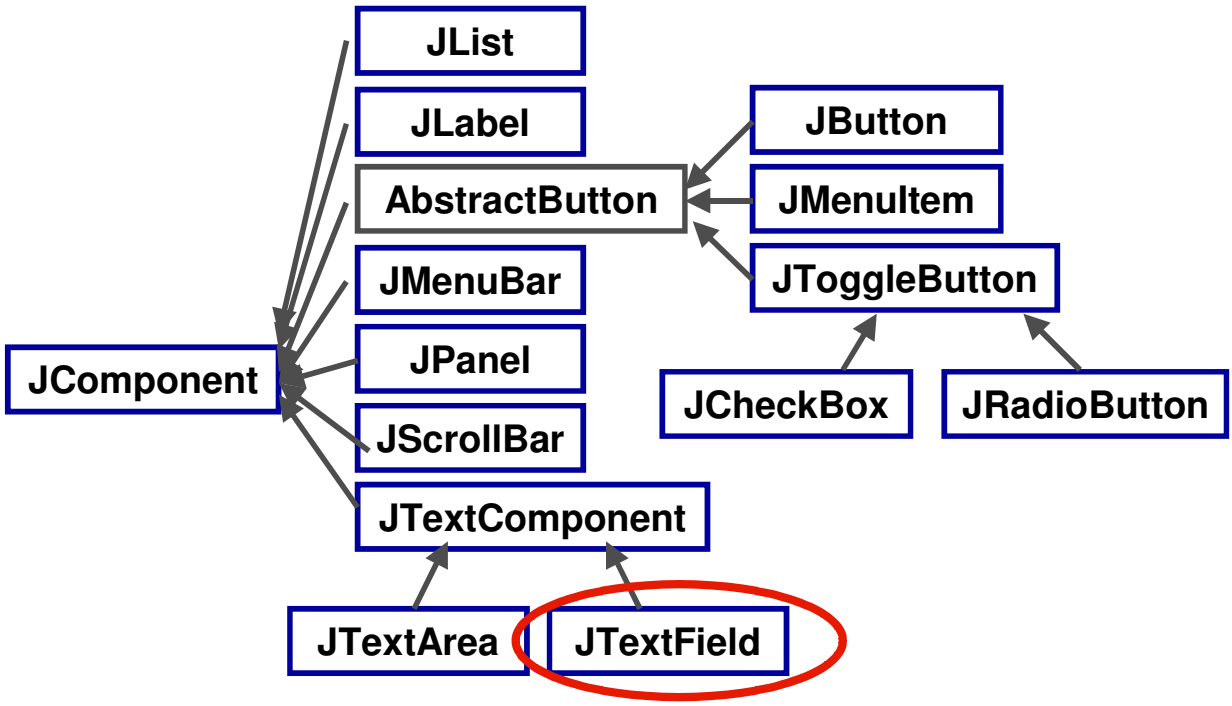
Crea due istanze di `Es9Listener` e li imposta ognuno come ascoltatore degli eventi per un pulsante, passando a ognu-no il pannello su cui agire e il colore da usare

Esempio con 2 pulsanti: variante

L'ascoltatore degli eventi:

```
class Es9Listener implements ActionListener
{
 private JPanel pannello;
 private Color  colore;
 public Es9Listener(JPanel p, Color c)
 {
   pannello = p;  colore = c;
 }
 public void actionPerformed(ActionEvent e)

 {
   pannello.setBackground(colore);
 }
}
```

Componenti Swing: JTextField

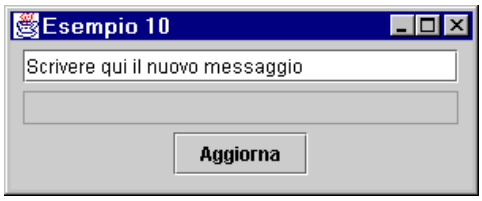


JTextField

- JTextField è un componente "campo di testo"
- Si può usare usabile per scrivere e visualizzare una riga di testo
  - il campo di testo può essere editabile o no
  - il testo è accessibile con **getText()** / **setText()**
- Ogni volta che il testo in esso contenuto cambia si genera un DocumentEvent nel documento che contiene il campo di testo

Esempio con JTextField

- Applicazione con un pulsante e due campi di testo
- Uno per scrivere testo, l'altro per visualizzarlo



- Quando si preme il pulsante, il testo del secondo campo (non modificabile dall'utente) viene cambiato, e reso uguale a quello scritto nel primo

Esempio con JTextField

Il solito main:

```
public class EsSwing10
{
 public static void main(String[] v)
 {
  JFrame f = new JFrame("Esempio 10");
  Container c = f.getContentPane();
  Es10Panel p = new Es10Panel();
  c.add(p);
  f.addWindowListener(new Terminator());
  f.setSize(300,120);
  f.show();
 }
}
```

Esempio con JTextField

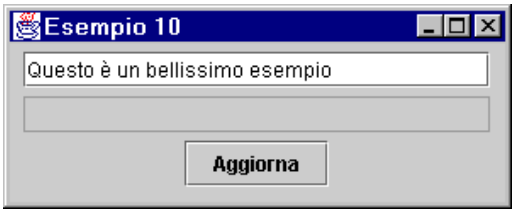
```
class Es10Panel extends JPanel
      implements ActionListener
{
 JButton b;
 JTextField txt1, txt2;
 public Es10Panel()
{
   super();
   b = new JButton("Aggiorna");
   txt1 = new JTextField("Scrivere qui il testo",25);
   txt2 = new JTextField(25);
   txt2.setEditable(false);
   b.addActionListener(this);
   add(txt1);
   add(txt2);
   add(b);
 }
 ...
```

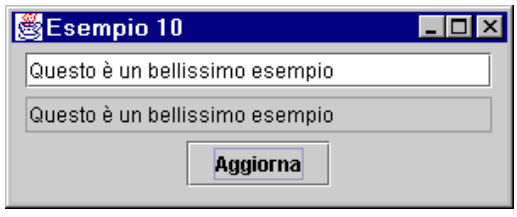
Il secondo campo di testo non è modificabile dall'utente

Esempio con JTextField

La gestione dell'evento "pulsante premuto":

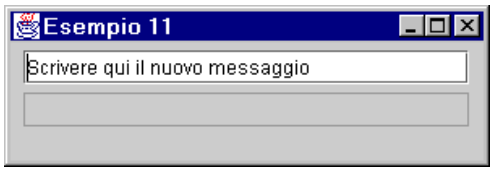
```
 ...

public void actionPerformed(ActionEvent e)
{
     txt2.setText(txt1.getText());
}

}
```





JTextField e documento

- Associato ad ogni campo di testo c'è un **documento** che gestisce il testo presente



- A ogni modifica del contenuto questo documento genera un **DocumentEvent** per segnalare l'avvenuto cambiamento
- Tale evento dev'essere gestito da un opportuno **DocumentListener**

JTextField e documento

- L'interfaccia DocumentListener dichiara tre metodi:

```
void insertUpdate(DocumentEvent e);
void removeUpdate(DocumentEvent e);
void changedUpdate(DocumentEvent e);
```

- Il terzo non è mai chiamato da un JTextField, serve solo per altri tipi di componenti
- L'oggetto DocumentEvent passato come parametro non ha nessuna utilità nel nostro esempio.

JTextField e documento

- Nel nostro caso l'azione da svolgere in caso di inserimento o rimo-zione di caratteri è identica, quindi i due metodi

```
void insertUpdate(DocumentEvent e);
void removeUpdate(DocumentEvent e);
```

  saranno identici (purtroppo vanno comunque imple-mentati entrambi)
- Anche metodo:

```
changedUpdate(DocumentEvent e)
```

  è inutile, dato che JTextField non lo chiama
- Va comunque formalmente implementato.

Il codice del nuovo esempio

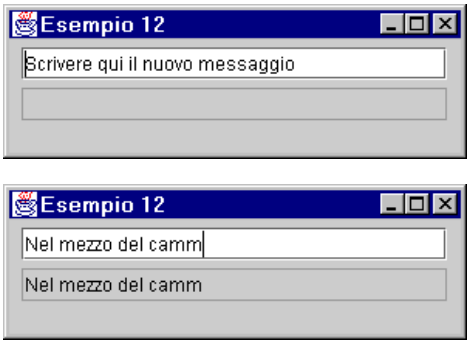
```
import javax.swing.event.*;

class Es12Panel extends JPanel
     implements DocumentListener
{
 JTextField txt1, txt2;

 public Es12Panel()

 {
   super();
   txt1 = new JTextField("Scrivere qui il testo", 25);
   txt2 = new JTextField(25); txt2.setEditable(false);
   txt1.getDocument().addDocumentListener(this);
   add(txt1);
   add(txt2);
 }
 ...
```

Ricava il documento di cui il campo di testo **txt1** fa parte, e gli associa come listener il pannello

Il codice del nuovo esempio

La gestione dell'evento:

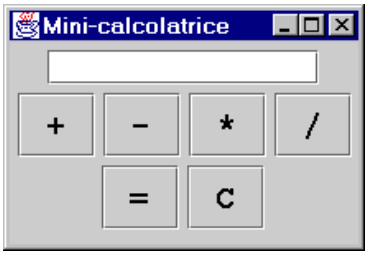
```
public void insertUpdate(DocumentEvent e)
  { txt2.setText(txt1.getText());  }
public void removeUpdate(DocumentEvent e)
  { txt2.setText(txt1.getText());  }
public void changedUpdate(DocumentEvent e)
  { }
```



Ora, a ogni carattere inserito o cancellato, l'aggiornamento è istantaneo e automatico

Una minicalcolatrice

Architettura:



- un pannello con un campo di testo e sei pulsanti
- un unico ActionListener per tutti i pulsanti (è il vero calcolatore)

Gestione degli eventi:

Ogni volta che si preme un pulsante:

- si recupera il nome del pulsante (è la successiva operazione da svolgere)
- si legge il valore nel campo di testo
- si svolge l'operazione precedente

Una minicalcolatrice

Esempio: 15 + 14 - 3 = + 8 =

- quando si preme +, si memorizzano sia 15 sia l'operazione +
- quando si preme -, si legge 14, si fa la somma 15+14, si memorizza 29, e si memorizza l'operazione -
- quando si preme =, si legge 3, si fa la sottrazione 29-3, si memorizza 26, e si memorizza l'operazione =
- quando si preme + (dopo l' =), è come essere all'inizio: si memorizzano 26 (risultato precedente) e l'operazione +
- quando si preme =, si legge 8, si fa la somma 26+8, si memorizza 34, e si memorizza l'operazione =
- ...eccetera...

Una minicalcolatrice

Il solito main:

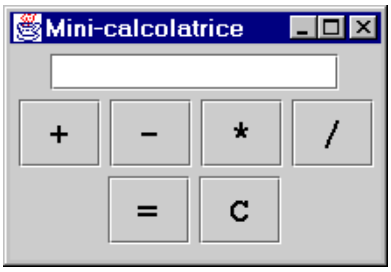
```
public class EsSwingCalculator
{
 public static void main(String[] v)
 {
  JFrame f = new JFrame("Mini-calcolatrice");
  Container c = f.getContentPane();
  CalcPanel p = new CalcPanel();
  c.add(p);
  f.setSize(220,150);
  f.addWindowListener(new Terminator());
  f.show();
 }
}
```

Una minicalcolatrice

Un pulsante con un font "personalizzato" :

```
class CalcButton extends JButton
{
  CalcButton(String n)
  {
    super(n);
    setFont(new Font("Courier",Font.BOLD,20));
 }
}
```

Un tipo di pulsante che si comporta come JButton, ma usa il font da noi specificato per l'etichetta



Una minicalcolatrice

Il pannello:

```
class CalcPanel extends JPanel
{
 JTextField txt;
 CalcButton sum, sub, mul, div, calc, canc;
 public CalcPanel()
 {
  super();
  txt =  new JTextField(15);
  txt.setHorizontalAlignment(JTextField.RIGHT);
  calc = new CalcButton("=");
  sum =  new CalcButton("+");
  sub =  new CalcButton("-");
  mul =  new CalcButton("*");
  div =  new CalcButton("/");
  canc = new CalcButton("C");
  ...
```

Il display ha i numeri allineati a destra

Una minicalcolatrice

Il pannello:

```
    ...
    add(txt);
    add(sum); add(sub);  add(mul);
    add(div); add(calc); add(canc);
    Calculator calcolatore =
      new Calculator(txt);
    sum.addActionListener(calcolatore);
    sub.addActionListener(calcolatore);
    mul.addActionListener(calcolatore);
    div.addActionListener(calcolatore);
    calc.addActionListener(calcolatore);
    canc.addActionListener(calcolatore);
  }
}
```

Un unico listener gestisce gli eventi di tutti i pulsanti

Una minicalcolatrice

Il listener / calcolatore:

```
class Calculator implements ActionListener
{
 double res = 0; JTextField display;
 String opPrec = "nop";
 public Calculator(JTextField t) { display = t; }
 public void actionPerformed(ActionEvent e)
 {
   double valore =
     Double.parseDouble(display.getText());
   display.setText("");
   display.requestFocus();
   ...
```

Fa sì che il campo di testo sia già selezionato, pronto per scriverci dentro

Recupera il valore dal campo di testo e lo converte da stringa a double

Una minicalcolatrice

Il listener / calcolatore:

Recupera il nome del pulsante premuto

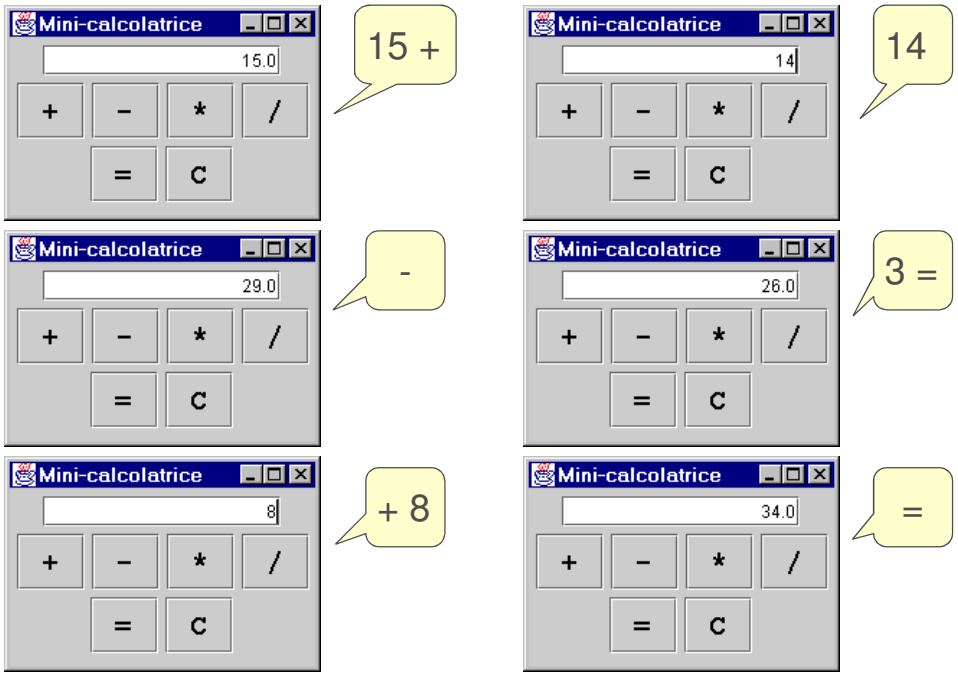
```
    ...
    String operazione = e.getActionCommand();
    if (operazione.equals("C"))
    { // cancella tutto
     res = valore = 0; opPrec = new String("nop");
    } else
    { // esegui l'operazione precedente
     if (opPrec.equals("+")) res += valore; else
     if (opPrec.equals("-")) res -= valore; else
     if (opPrec.equals("*")) res *= valore; else
     if (opPrec.equals("/")) res /= valore; else
     if (opPrec.equals("nop")) res = valore;
     display.setText(""+res);
     opPrec = operazione;
    }
  }
}
```

L'operazione attuale è quella da eseguire la prossima volta

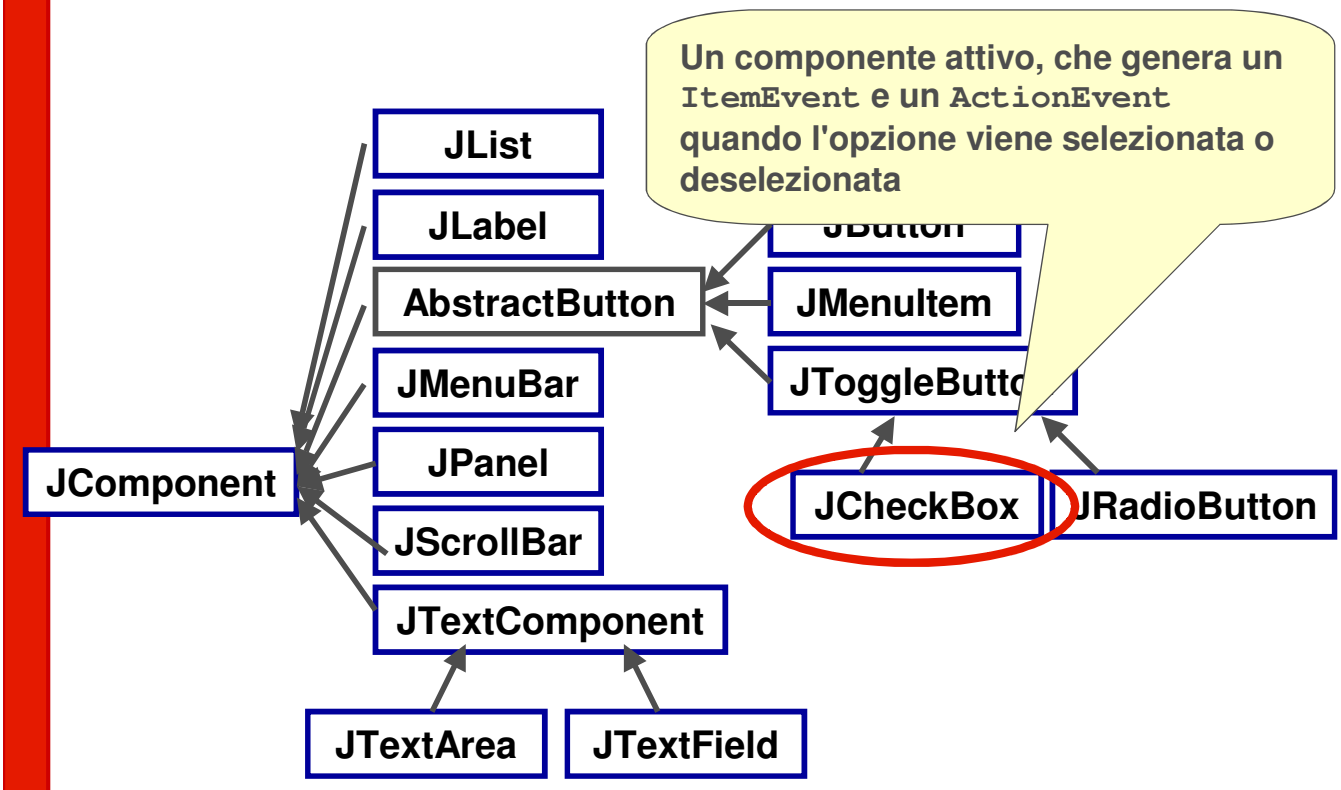
Se non c'è nessuna operazione precedente, memorizza solo il valore

Una minicalcolatrice

Esempio di uso:



Componenti Swing: CheckBox



JCheckBox (casella di opzione)

- Il **JCheckBox** è una "casella di opzione", che può essere selezionata o deselezionata
  - lo stato è verificabile con **isSelected()** e modificabile con **setSelected()**
- Ogni volta che lo stato della casella cambia, si generano:
  - un **ActionEvent**, come per ogni pulsante
  - un **ItemEvent**, gestito da un ItemListener
- Solitamente conviene gestire l'ItemEvent, perché più specifico.

JCheckBox (casella di opzione)

- L' ItemListener dichiara il metodo:

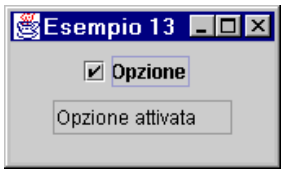
```
public void itemStateChanged(ItemEvent e)
```

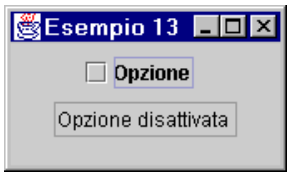
  che deve essere implementato dalla classe che realizza l'ascoltatore degli eventi

- In caso di più caselle gestite dallo stesso listener, il metodo **e.getItemSelectable()** restituisce un riferimento all'oggetto sorgente dell'evento

Esempio

- Un'applicazione comprendente una checkbox e un campo di testo (non modificabile), che riflette lo stato della checkbox





- Alla checkbox è associato un ItemListener, che intercetta gli eventi di selezione / deselezione implementando il metodo itemStateChanged()

Esempio

```
class Es13Panel extends JPanel implements ItemListener
{
 JTextField txt; JCheckBox ck1;
 public Es13Panel()
 {
   super();
   txt = new JTextField(10); txt.setEditable(false);
   ck1 = new JCheckBox("Opzione");
   ck1.addItemListener(this);
   add(ck1); add(txt);
 }
 public void itemStateChanged(ItemEvent e)
 {
  if (ck1.isSelected())
     txt.setText("Opzione attivata");
  else txt.setText("Opzione disattivata");
 }
}
```


Esempio con più caselle

- Un'applicazione con due checkbox e un campo di testo che ne riflette lo stato

- Lo stesso ItemListener è associato a entrambe le checkbox: usa e.getItemSelectable() per dedurre quale casella è stata modificata

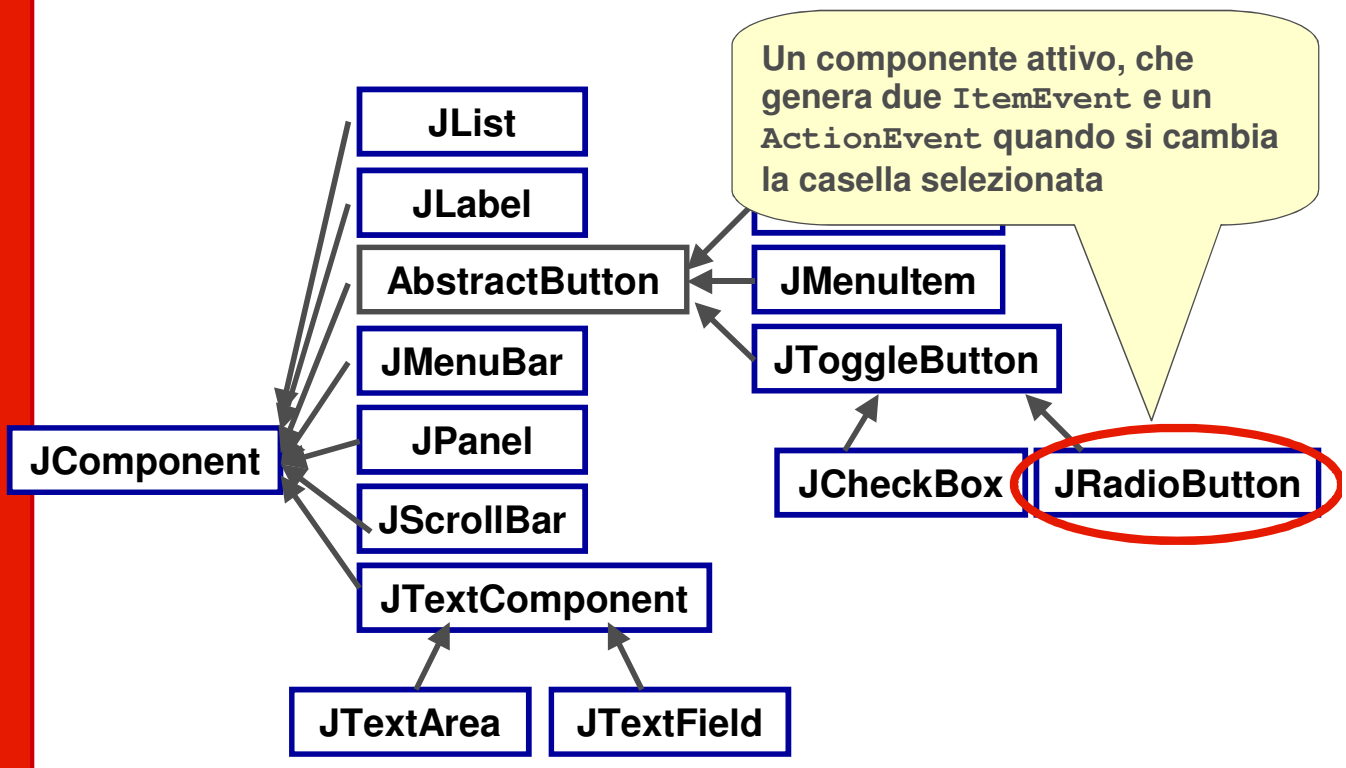
Esempio

```
class Es14Panel extends JPanel
   implements ItemListener
{
 JTextField txt1, txt2;
 JCheckBox c1, c2;
 public Es14Panel()
 {
  super();
  txt1 = new JTextField(15); txt1.setEditable(false);
  txt2 = new JTextField(15); txt2.setEditable(false);
  c1 = new JCheckBox("Mele");
  c1.addItemListener(this);
  c2 = new JCheckBox("Pere");
  c2.addItemListener(this);
  add(c1);   add(c2);
  add(txt1); add(txt2);
 }
```


Esempio

```
...
public void itemStateChanged(ItemEvent e)
{
  Object source = e.getItemSelectable();
  if (source==c1)
    txt1.setText("Sono cambiate le mele");
  else
    txt1.setText("Sono cambiate le pere");
  // ora si controlla lo stato globale
  String frase = (c1.isSelected() ? "Mele " : "")
               + (c2.isSelected() ? "Pere" : "");
  txt2.setText(frase);
 }
}
```

Componenti Swing: JRadioButton

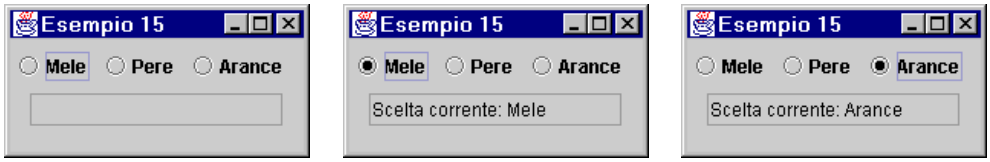


Il radio button

- Il **JRadioButton** è una casella di opzione che fa parte di un gruppo: in ogni istante può essere attiva una sola casella del gruppo
- Quando si cambia la casella selezionata, si generano tre eventi
  - un ItemEvent per la casella deselezionata, uno per la casella selezionata,
  - un ActionEvent da parte della casella selezionata (pulsante premuto)
- In pratica:
  - si creano i JRadioButton che servono
  - si crea un oggetto ButtonGroup e si aggiungono i JRadioButton al gruppo

Esempio

- Un'applicazione comprendente un gruppo di tre radiobutton, con un campo di testo che ne riflette lo stato



- Solitamente conviene gestire l'ActionEvent (più che l'ItemEvent)
- Ogni cambio di selezione ne genera infatti uno solo (a fronte di due ItemEvent), il che semplifica la gestione

Esempio

```
class Es15Panel extends JPanel
   implements ActionListener
{
 JTextField   txt; JRadioButton b1, b2, b3;
 ButtonGroup  grp;
 public Es15Panel()
 {
  super();
  txt = new JTextField(15); txt.setEditable(false);
  b1  = new JRadioButton("Mele");
  b2  = new JRadioButton("Pere");
  b3  = new JRadioButton("Arance");
  grp = new ButtonGroup();
  grp.add(b1);  grp.add(b2);  grp.add(b3);
  b1.addActionListener(this); add(b1);
  b2.addActionListener(this); add(b2);
  b3.addActionListener(this); add(b3);
  add(txt);
 }
```

Esempio

```
  ...

 public void actionPerformed(ActionEvent e)
 {
  String scelta = e.getActionCommand();
  txt.setText("Scelta corrente: " + scelta);
 }
}
```


La gestione del layout

- Quando si aggiungono componenti a un contenitore (in particolare: a un pannello), la loro posizione è decisa dal Gestore di Layout (**Layour Manager**)
- Il gestore predefinito per un pannello è FlowLayout, che dispone i componenti in fila (da sinistra a destra e dall'alto in basso)
- Semplice, ma non sempre esteticamente efficace
- Esistono comunque altri gestori alternativi, più o meno complessi.

Layout manager

Oltre a **FlowLayout**, vi sono:

- **BorderLayout**, che dispone i componenti lungo i bordi (nord, sud, ovest, est) o al centro
- **GridLayout**, che dispone i componenti in una griglia m × n
- **GridBagLayout**, che dispone i componenti in una griglia m × n *flessibile*
  - righe e colonne a dimensione variabile
  - molto flessibile e potente, ma difficile da usare

....

Layout manager

... e inoltre:

- **BoxLayout,** che dispone i componenti o in orizzontale o in verticale, in un'unica casella (layout predefinito per il componente Box)
- **Nessun layout manager**
  - si specifica la posizione assoluta (x,y) del componente
  - sconsigliato perché dipendente dalla piattaforma

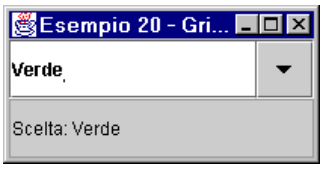
Per cambiare Layout Manager:

```
setLayout(new GridLayout(4,5))
```

Lo stesso pannello con...

… FlowLayout…

… GridLayout …
(griglia 2 x 1)



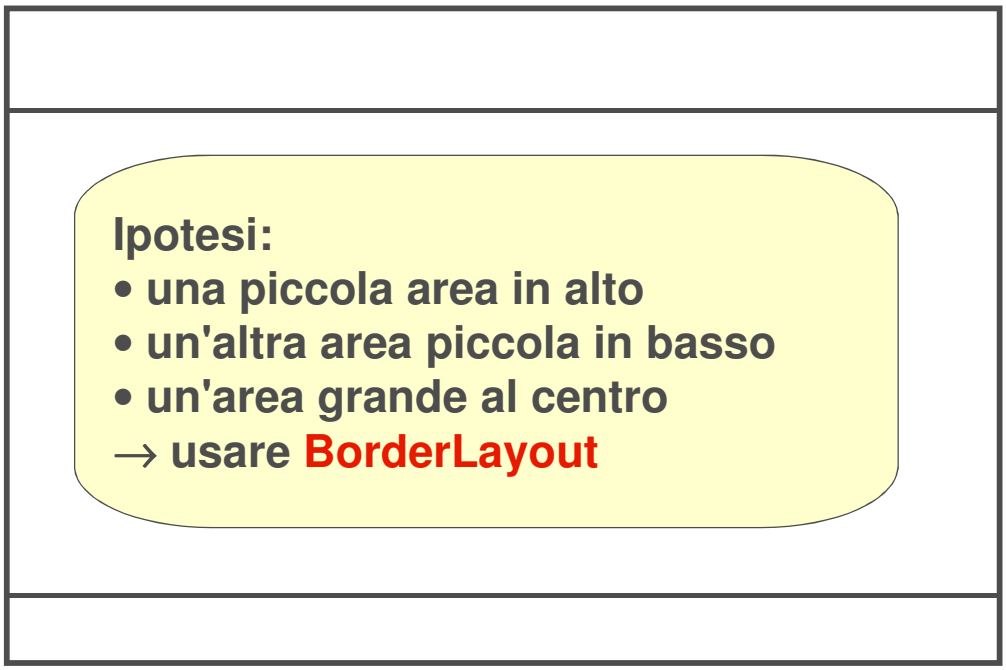
… BorderLayout …
(nord e sud)

… e senza alcun layout.
(posizioni a piacere)

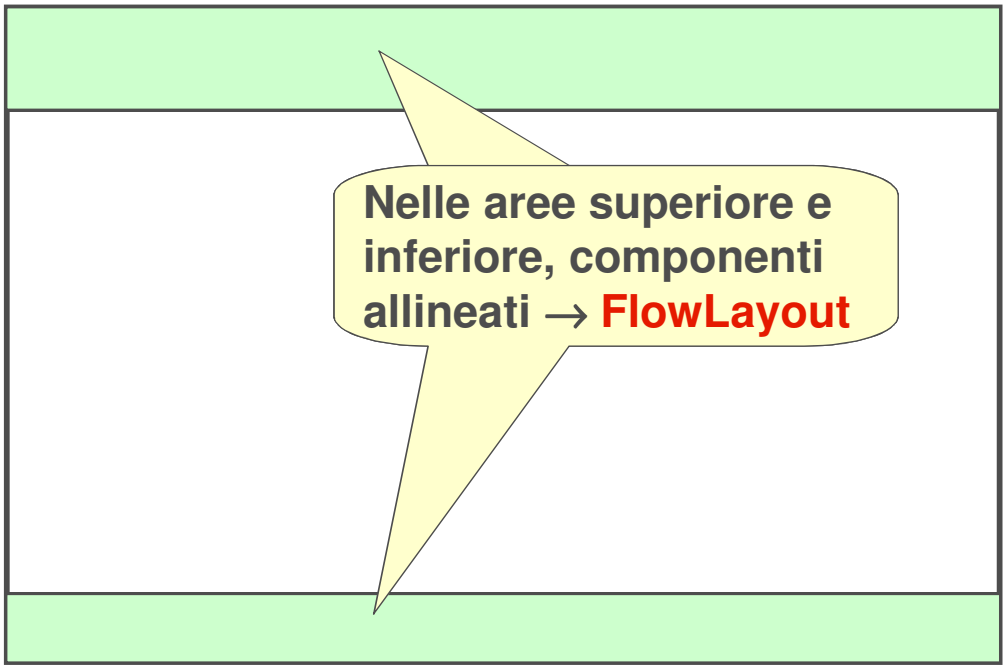
Progettare un'interfaccia

- Spesso, per creare un'interfaccia grafica completa, efficace e gradevole non basta un singolo gestore di layout
- Approccio tipico:
  1. suddividere l'area in zone, corrispondenti ad altrettanti pannelli
  2. applicare a ogni zona il layout manager più opportuno

Esempio



Esempio



Esempio