

JAVA E LA GRAFICA

L'architettura Java è *graphics-ready*

- **Package `java.awt`**
 - il primo package grafico (Java 1.0)
 - indipendente dalla piattaforma... o quasi!
- **Package `javax.swing`**
 - il nuovo package grafico (Java 2; versione preliminare da Java 1.1.6)
 - scritto esso stesso in Java, realmente indipendente dalla piattaforma

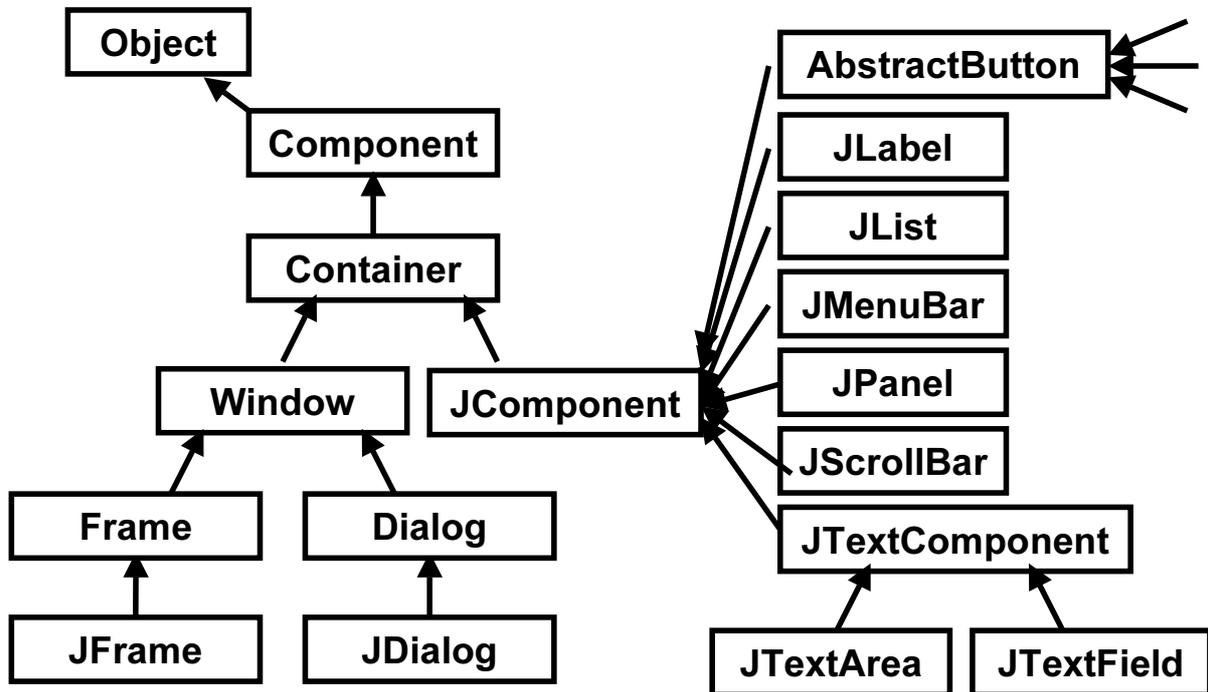
Swing - 1

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, etc etc
- **Programmazione “event-driven”:**
 - non più algoritmi stile input/elaborazione/output...
 - ... ma *reazione agli eventi* che l'utente, in modo interattivo, genera sui componenti grafici
- **Concetti di evento e di ascoltatore degli eventi**

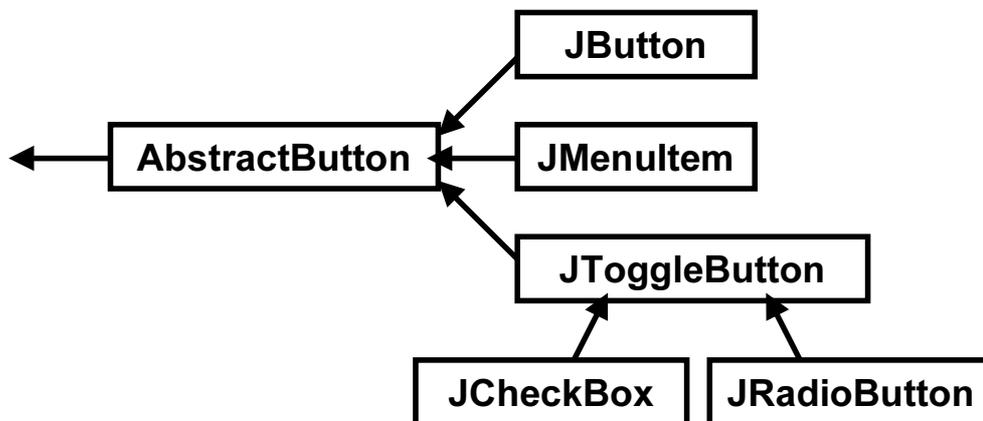
Swing - 2

SWING: GERARCHIA DI CLASSI



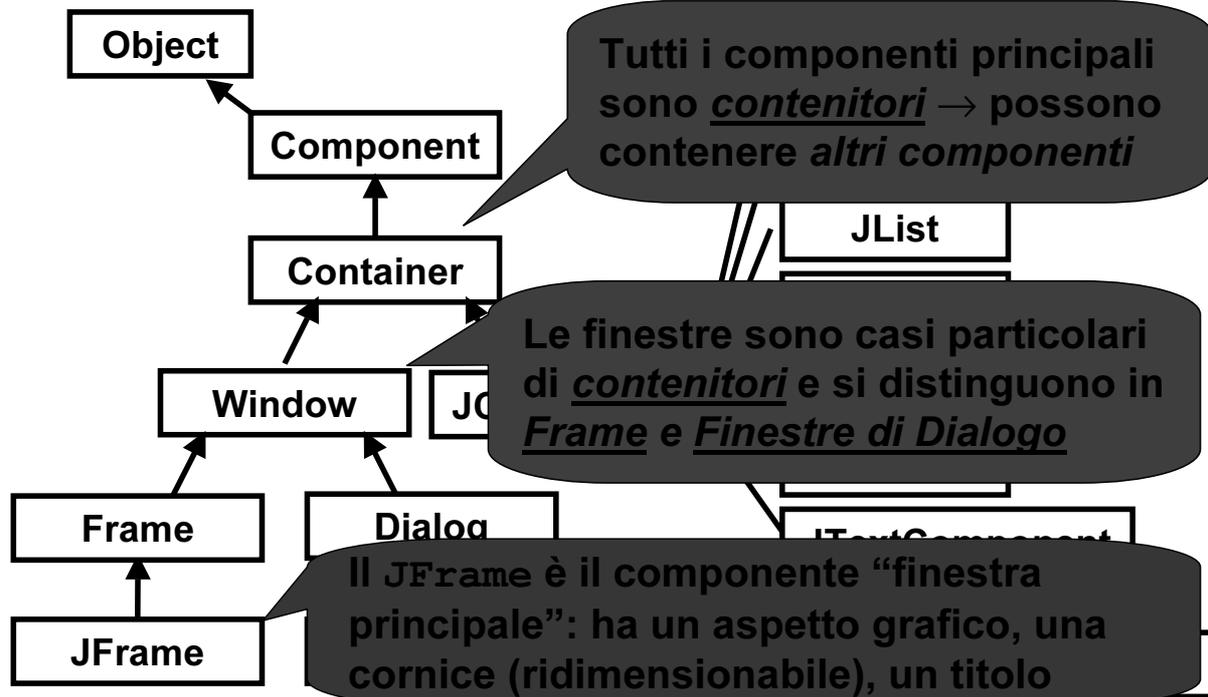
Swing - 3

SWING: GERARCHIA DI CLASSI



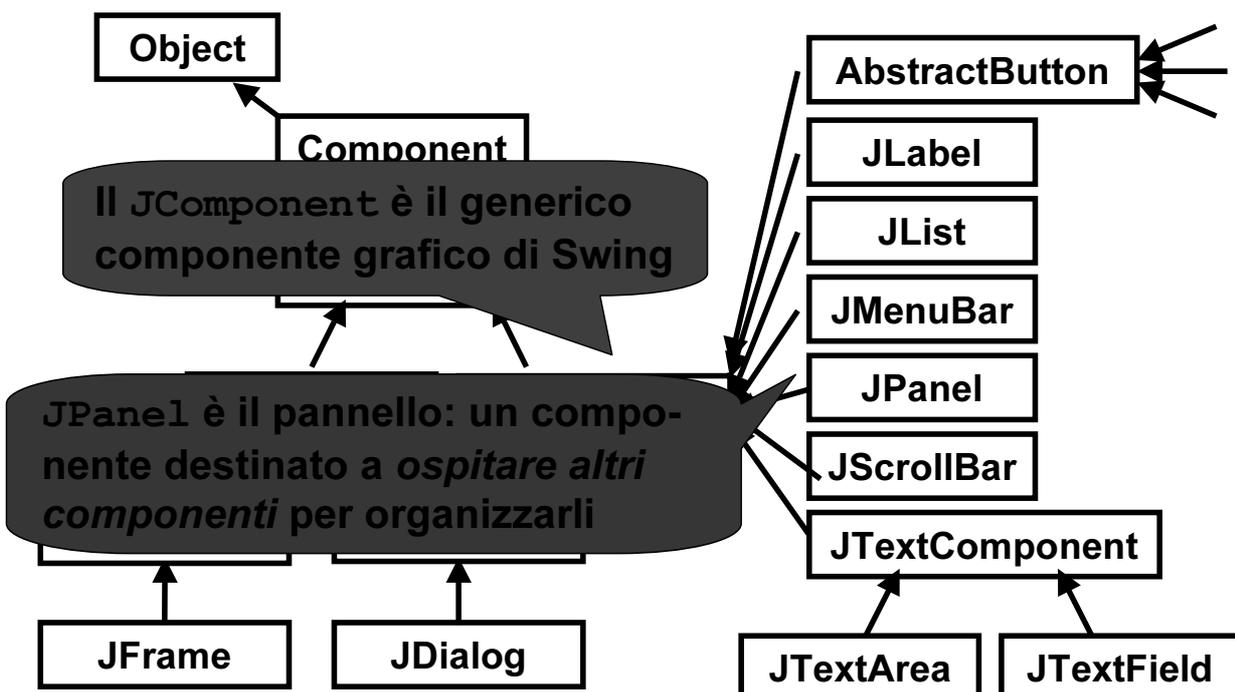
Swing - 4

SWING: GERARCHIA DI CLASSI



Swing - 5

SWING: GERARCHIA DI CLASSI



Swing - 6

SWING: UN ESEMPIO

- La più semplice applicazione grafica consiste in una classe il cui main *crea un JFrame e lo rende visibile col metodo 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, col titolo specificato

Swing - 7

SWING: UN ESEMPIO

- La più semplice applicazione grafica consiste in una classe il cui main *crea un JFrame e lo rende visibile col metodo show()*. I comandi standard sono già attivi (la chiusura per default *nasconde il frame senza chiuderlo realmente*)

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

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



Per chiuderla, CTRL+C dalla console

Swing - 8

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()`, che ha come parametro un opportuno oggetto di classe `Dimension`:

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

Larghezza (x), Altezza (y)

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

Swing - 9

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 - 10

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.setBounds(200,100, 300,150)
        f.show();
    }
}
```

Posizione iniziale = (200,100)
Larghezza = 300, Altezza = 150

Swing - 11

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);
        setBounds(200,100, 300,150);
    }
}
```

Swing - 12

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();
    }
}
```

Posizione iniziale = (200,100)
Larghezza = 300, Altezza = 150

Swing - 13

STRUTTURA DEL FRAME

- In Swing *non si possono aggiungere nuovi componenti* direttamente al `JFrame`
- Dentro a ogni `JFrame` c'è un `Container`, recuperabile col metodo `getContentPane()`: è a lui che vanno aggiunti i nuovi componenti
- Tipicamente, si aggiunge un pannello (un `JPanel` o una nostra versione più specifica), tramite il metodo `add()`
 - sul pannello si può disegnare (forme, immagini...)
 - ...o aggiungere pulsanti, etichette, icone, etc

Swing - 14

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!

DISEGNARE SU UN PANNELLO

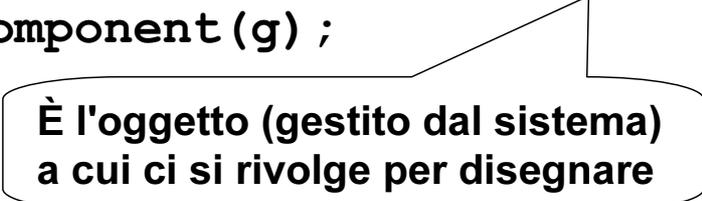
Per disegnare su un pannello occorre:

- definire una propria classe (`MyPanel`) che estenda il `JPanel` originale
- in tale classe, *ridefinire* `paintComponent()`, che è il metodo (ereditato da `JComponent`) che si occupa di disegnare il componente
 - **ATTENZIONE:** il nuovo `paintComponent()` da noi definito deve sempre richiamare il metodo `paintComponent()` originale, tramite `super`

DISEGNARE SU UN PANNELLO

Il nostro pannello personalizzato:

```
public class MyPanel extends JPanel {  
    // nessun costruttore, va bene il default  
    public void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        ...  
    }  
}
```



È l'oggetto (gestito dal sistema) a cui ci si rivolge per disegnare

Qui aggiungeremo le nostre istruzioni di disegno

Swing - 17

DISEGNARE SU UN PANNELLO

Quali metodi per disegnare?

- `drawImage()`, `drawLine()`, `drawRect()`,
`drawRoundRect()`, `draw3DRect()`,
`drawOval()`, `drawArc()`, `drawString()`,
`drawPolygon()`, `drawPolyLine()`
- `fillRect()`, `fillRoundRect()`,
`fill3DRect()`, `fillOval()`, `fillArc()`,
`fillPolygon()`, `fillPolyLine()`
- `getColor()`, `getFont()`, `setColor()`,
`setFont()`, `copyArea()`, `clearRect()`

Swing - 18

ESEMPIO: DISEGNO DI FIGURE

Il pannello personalizzato con il disegno:

```
public class MyPanel extends JPanel {  
    public void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        g.setColor(Color.red);  
        g.fillRect(20,20, 100,80);  
        g.setColor(Color.blue);  
        g.drawRect(30,30, 80,60);  
        g.setColor(Color.black);  
        g.drawString("ciao", 50,60);  
    }  
}
```

Swing - 19

ESEMPIO: DISEGNO DI FIGURE

Il pannello personalizzato con il disegno:

```
public class M  
    public void p  
    super.paintC  
    g.setColor(Co  
    g.fillRect(20,20, 100,80);  
    g.setColor(Color.blue);  
    g.drawRect(30,30, 80,60);  
    g.setColor(Color.black);  
    g.drawString("ciao", 50,60);  
    }  
}
```

Colori possibili: white, gray,
lightGray, darkGray, red, green,
blue, yellow, magenta, cyan, pink,
orange, black

Swing - 20

ESEMPIO: DISEGNO DI FIGURE

Il main che lo crea e lo inserisce nel frame:

```
import java.awt.*; import javax.swing.*;
public class EsSwing4 {
    public static void main(String[] v){
        MyFrame f = new MyFrame("Esempio 4");
        Container c = f.getContentPane();
        MyPanel panel = new MyPanel();
        c.add(panel);
        f.show();
    }
}
```

Potremmo usare anche un JFrame standard: il MyFrame ha il vantaggio di essere già di dimensioni opportune

Swing - 21

ESEMPIO: DISEGNO DI FIGURE

Il main che lo crea e lo inserisce nel frame:

```
import java.awt.*; import javax.swing.*;
public class EsSwing4 {
    public static void main(String[] v){
        MyFrame f = new MyFrame("Esempio 4");
        Container c = f.getContentPane();
        MyPanel panel = new MyPanel();
        c.add(panel);
        f.show();
    }
}
```



Potremmo usare anche un JFrame standard: il MyFrame ha il vantaggio di essere già di dimensioni opportune

Swing - 22

ESEMPIO: DISEGNO DI FIGURE

Per cambiare font:

- si crea un oggetto `Font` appropriato
- lo si imposta come font predefinito usando il metodo `setFont()`

```
Font f1 =  
    new Font("Times", Font.BOLD, 20);  
g.setFont(f1);
```

Il nome del font

Dimensione
in punti

Stile: `Font.PLAIN`, `Font.BOLD`, `Font.ITALIC`
(corrispondono a 0,1,2,3: `BOLD` e `ITALIC` si sommano)

Swing - 23

ESEMPIO: DISEGNO DI FIGURE

Recuperare le proprietà di un font

- Il font corrente si recupera con `getFont()`
- Dato un `Font`, le sue proprietà si recuperano con `getName()`, `getStyle()`, `getSize()`
- e si verificano con i predicati `isPlain()`, `isBold()`, `isItalic()`

```
Font f1 = g.getFont();  
  
int size = f1.getSize();  
int style = f1.getStyle();  
String name = f1.getName();
```

Swing - 24

ESERCIZIO: GRAFICO DI F(X)

Per disegnare il grafico di una funzione occorre

- creare un'apposita classe `FunctionPanel` che estenda `JPanel`, ridefinendo il metodo `paintComponent()` come appropriato
 - sfondo bianco, cornice nera
 - assi cartesiani rossi, con estremi indicati
 - funzione disegnata in blu
- creare, nel main, un oggetto di tipo `FunctionPanel`

Swing - 25

ESERCIZIO: GRAFICO DI F(X)

Il solito main:

```
import java.awt.*; import javax.swing.*;
public class EsSwing5 {
    public static void main(String[] v) {
        JFrame f = new JFrame("Grafico f(x)");
        Container c = f.getContentPane();
        FunctionPanel p = new FunctionPanel();
        c.add(p);
        f.setBounds(100,100,500,400);
        f.show();
    }
}
```

Swing - 26

ESERCIZIO: GRAFICO DI F(X)

Il pannello apposito:

```
class FunctionPanel extends JPanel {  
    int xMin=-7, xMax=7, yMin=-1, yMax=1;  
    int larghezza=500, altezza=400;  
    float fattoreScalaX, fattoreScalaY;  
  
    public void paintComponent(Graphics g) {  
        super.paintComponent(g);  
        setBackground(Color.white);  
        fattoreScalaX=larghezza/((float)xMax-xMin);  
        fattoreScalaY=altezza/((float)yMax-yMin);  
        ...  
    }  
}
```

Swing - 27

ESERCIZIO: GRAFICO DI F(X)

```
...  
// cornice  
g.setColor(Color.black);  
g.drawRect(0,0, larghezza-1, altezza-1);  
// assi cartesiani  
g.setColor(Color.red);  
g.drawLine(0, altezza/2, larghezza-1, altezza/2);  
g.drawLine(larghezza/2, 0, larghezza/2, altezza-1);  
// scrittura valori estremi  
g.drawString(""+xMin, 5, altezza/2-5);  
g.drawString(""+xMax, larghezza-10, altezza/2-5);  
g.drawString(""+yMax, larghezza/2+5, 15);  
g.drawString(""+yMin, larghezza/2+5, altezza-5);  
...
```

Swing - 28

ESERCIZIO: GRAFICO DI F(X)

```
...
// grafico della funzione
g.setColor(Color.blue);
setPixel(g,xMin,f(xMin));
for (int ix=1; ix<larghezza; ix++){
    float x = xMin+((float)ix)/fattoreScalaX;
    setPixel(g,x,f(x));
}
}

static float f(float x){
    return (float)Math.sin(x);
}
...
```

La funzione da
graficare (statica)

Swing - 29

ESERCIZIO: GRAFICO DI F(X)

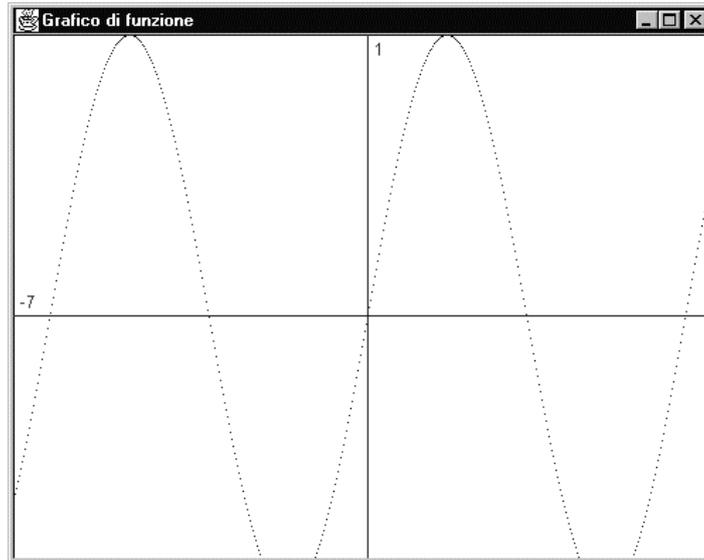
```
void setPixel(Graphics g, float x, float y){
    if (x<xMin || x>xMax || y<yMin || y>yMax )
        return;

    int ix = Math.round((x-xMin)*fattoreScalaX);
    int iy = altezza-Math.round(
        (y-yMin)*fattoreScalaY);
    g.drawLine(ix,iy,ix,iy); // singolo punto
}

}
```

Swing - 30

ESERCIZIO: GRAFICO DI F(X)



Swing - 31

DISEGNARE IMMAGINI

Come si disegna un'immagine?

- 1) ci si procura un apposito oggetto Image
- 2) si crea un oggetto MediaTracker che segua il caricamento dell'immagine, e gli si affida l'immagine da caricare
 - necessario perché `drawImage ()` ritorna al chiamante subito dopo aver *iniziato* il caricamento dell'immagine, senza attendere di averla caricata
 - senza MediaTracker, l'immagine *può non essere visualizzata* prima della fine del programma
- 3) si disegna l'immagine con `drawImage ()`

Swing - 32

DISEGNARE IMMAGINI

E come ci si procura l'oggetto Image?

1) si recupera il "toolkit di default":

```
Toolkit tk = Toolkit.getDefaultToolkit();
```

2) si chiede al toolkit di recuperare l'immagine:

```
Image img = tk.getImage("new.gif");
```

Sono supportati i formati GIF e JPEG

Si può anche fornire un URL:

```
URL url = ...;
```

```
Image img = tk.getImage(url);
```

Swing - 33

DISEGNARE IMMAGINI

E il MediaTracker?

1) Nel costruttore del pannello, si crea un oggetto **MediaTracker**, precisandogli su quale componente avverrà il disegno...

```
MediaTracker mt = new MediaTracker(this);
```

Di solito il parametro è **this** (il pannello stesso)

2) ...si aggiunge l'immagine al **MediaTracker**...

```
mt.addImage(img, 1);
```

Il parametro è un intero, a nostra scelta, che identifica univocamente l'immagine

Swing - 34

DISEGNARE IMMAGINI

E il MediaTracker?

3) ..e gli si dice di attendere il caricamento di tale immagine, usando l'ID assegnato

```
try { mt.waitForID(1); }  
catch (InterruptedException e) {}
```

Occorre un blocco `try/catch` perché l'attesa potrebbe essere interrotta da un'eccezione.

Se si devono attendere molte immagini:

```
try { mt.waitForAll(); }  
catch (InterruptedException e) {}
```

Swing - 35

DISEGNARE IMMAGINI: ESEMPIO

```
public class ImgPanel extends JPanel {  
    Image img1;  
    public ImgPanel() {  
        Toolkit tk = Toolkit.getDefaultToolkit();  
        img1 = tk.getImage("new.gif");  
        MediaTracker mt = new MediaTracker(this);  
        mt.addImage(img1, 1);  
        // aggiunta di eventuali altre immagini  
        try { mt.waitForAll(); }  
        catch (InterruptedException e) {}  
    }  
    ...  
}
```

Swing - 36

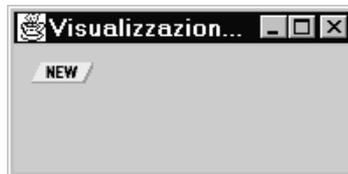
DISEGNARE IMMAGINI: ESEMPIO

...

```
public void paintComponent(Graphics g) {  
    super.paintComponent(g);  
    g.drawImage(img1, 30, 30, null);  
}
```

Le coordinate (x,y) della posizione in cui disegnare l'immagine (angolo superiore sinistro)

Un oggetto cui notificare l'avvenuto caricamento (solitamente null)



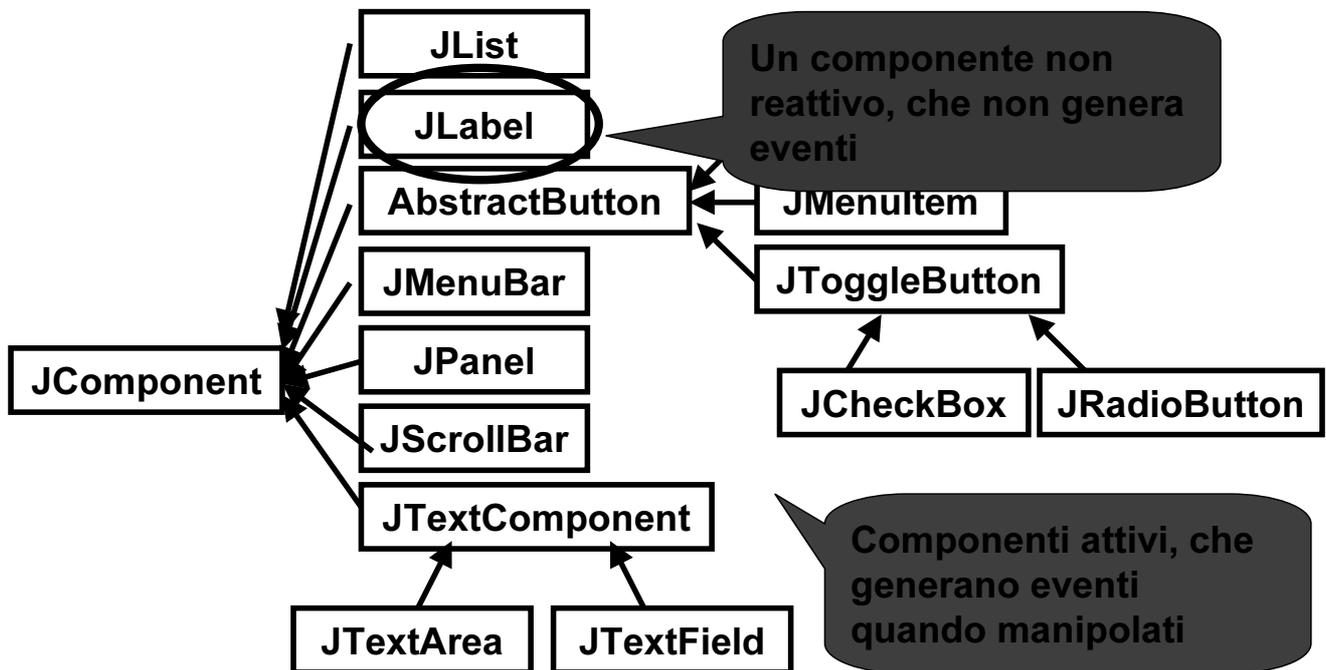
Swing - 37

OLTRE IL SOLO DISEGNO

- Finora, la grafica considerata consisteva nel *puro disegno* di forme e immagini
- È grafica "passiva": non consente all'utente alcuna interazione
 - si può solo guardare il disegno...!!
- La costruzione di interfacce grafiche richiede invece interattività
 - l'utente deve poter premere bottoni, scrivere testo, scegliere elementi da liste, etc etc
- Componenti *attivi*, che generano eventi

Swing - 38

SWING: GERARCHIA DI CLASSI



Swing - 39

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

Swing - 40

ESEMPIO: USO DI JLabel

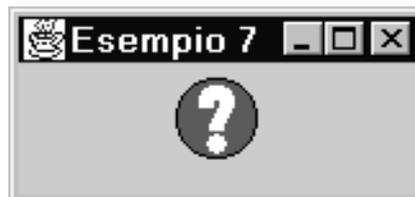
```
public class Es7Panel extends JPanel {  
    public Es7Panel() {  
        super();  
        JLabel lb1 = new JLabel("Etichetta");  
        add(lb1);  
    }  
}
```



Swing - 41

VARIANTE: JLabel CON ICONA

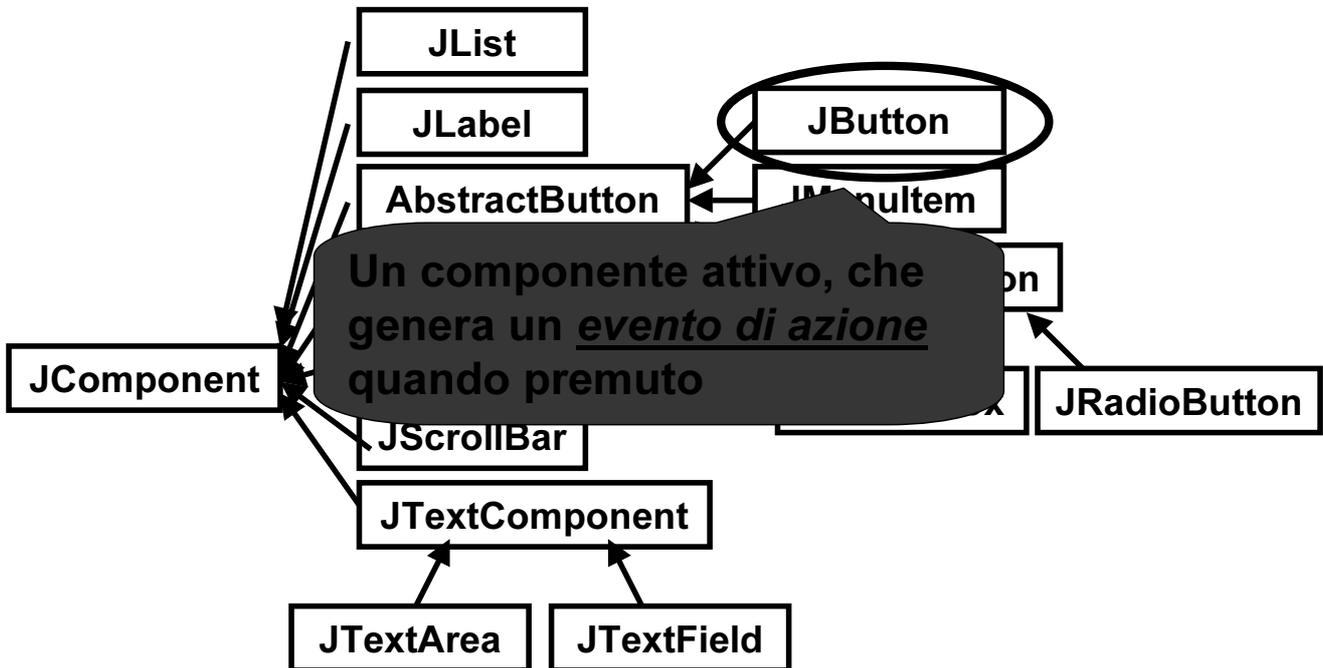
```
public class Es7Panel extends JPanel {  
    public Es7Panel() {  
        super();  
        JLabel lb2 = new JLabel( new ImageIcon("image.gif") );  
        add(lb2);  
    }  
}
```



Si evita MediaTracker e relative complicazioni.

Swing - 42

SWING: GERARCHIA DI CLASSI



Swing - 43

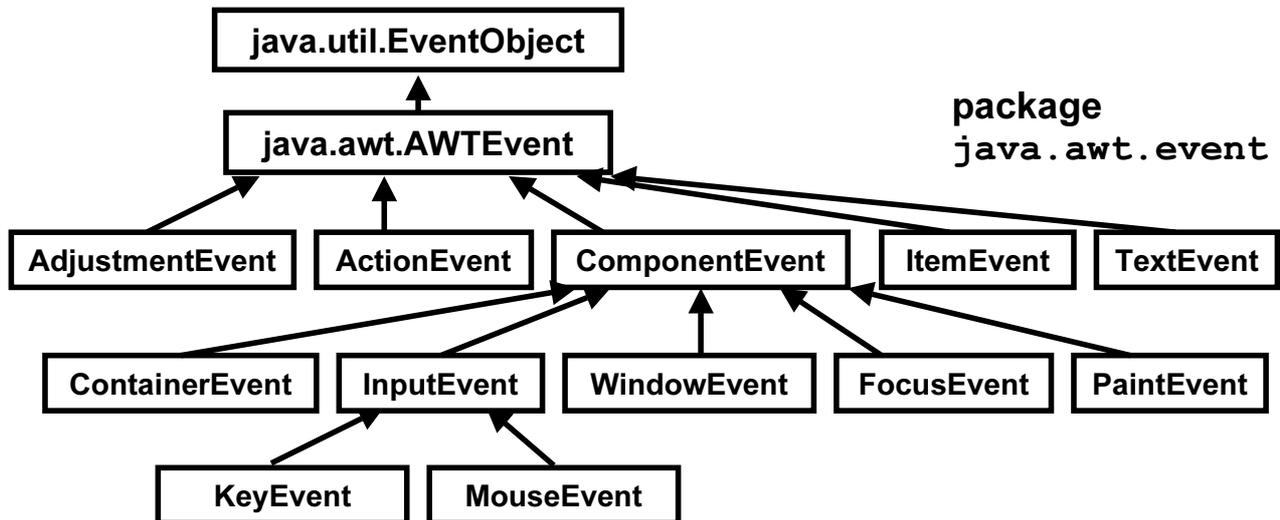
EVENTI

- 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 la casella è stata selezionata / deselezionata

Swing - 44

EVENTI IN JAVA

In Java, un *evento* è un oggetto, istanza di (una sottoclasse di) `java.util.EventObject`



Swing - 45

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

Premi qui



Event Listener

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

Swing - 46

GESTIONE DEGLI EVENTI

- Quando si interagisce con un componente "attivo" si genera un evento, che è un oggetto `Event` della (sotto)classe 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), è interattiva e reattiva

Swing - 47

IL PULSANTE `JButton`

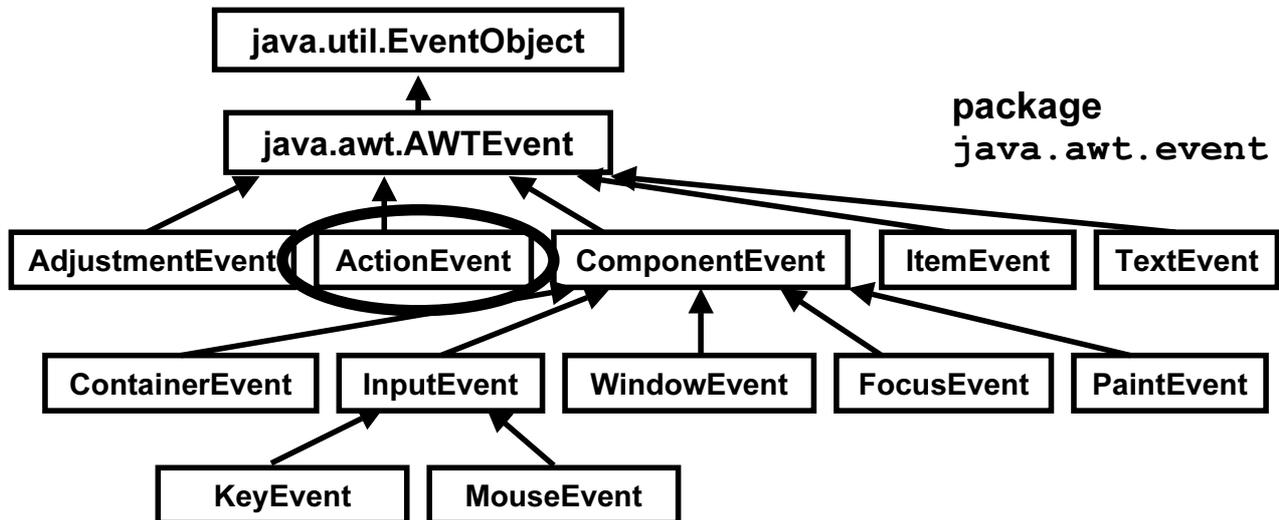
- Quando viene premuto, un bottone genera un evento di classe `ActionEvent`
- Questo evento viene inviato dal sistema allo specifico *ascoltatore degli eventi*, di classe `ActionListener`, registrato per quel bottone
 - può essere un oggetto di un'altra classe...
 - .. o anche il pannello stesso (`this`)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

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

Swing - 48

IL PULSANTE JButton

Un bottone premuto genera un `ActionEvent`



Swing - 49

ESEMPIO: USO DI JButton

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



Swing - 50

ESEMPIO: USO DI JButton

Architettura dell'applicazione

- Un pannello che contiene etichetta e pulsante
→ il costruttore del pannello crea l'etichetta e il pulsante
- Il pannello fa da *ascoltatore degli eventi* per il pulsante → il costruttore del pannello imposta il pannello stesso come *ascoltatore degli eventi* del pulsante

Swing - 51

ESEMPIO: USO DI JButton

Architettura dell'applicazione

- Un pannello che contiene etichetta e pulsante
→ il costruttore del pannello crea l'etichetta e il pulsante

- Il p
pu
sta
ev
- ```
public Es8Panel() {
 super();
 l = new JLabel("Tizio");
 add(l);
 JButton b = new JButton("Tizio/Caio");
 add(b);

}
```
- er il  
o-  
gli

Swing - 52

```

public Es8Panel() {
 super();
 l = new JLabel("Tizio");
 add(l);
 JButton b = new JButton("Tizio/Caio");
 add(b);
 b.addActionListener(this);
}

```

Ar

•

e n

lsante  
netta

- Il pannello fa da *ascoltatore degli eventi* per il pulsante → il costruttore del pannello imposta il pannello stesso come *ascoltatore degli eventi* del pulsante

Swing - 53

## ESEMPIO: USO DI JButton

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



Swing - 54

## ESEMPIO: USO DI JButton

### Eventi da gestire:

- l'evento `ActionPerformed` è generato dal pulsante quando viene premuto. Corrisponde al metodo `ActionPerformed()` dell'interfaccia `ActionListener`.

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

Il metodo `getText()` restituisce il testo attualmente contenuto nell'etichetta, mentre `setText()` cambia il testo dell'etichetta.

- l'ascoltatore dell'evento, che implementa il metodo `ActionPerformed()`, deve recuperare il testo dell'etichetta e cambiarlo.



Swing - 55

## ESEMPIO: USO DI JButton

```
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`

Etichetta del pulsante

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

Swing - 56

## ESEMPIO: USO DI JButton

...

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



Swing - 57

## ESEMPIO: USO DI JButton

Il solito main:

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

Necessario importare 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();
 }
}
```

Swing - 58

# UNA VARIANTE

## 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

Swing - 59

# UNA VARIANTE

```
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`

Swing - 60

# UNA VARIANTE

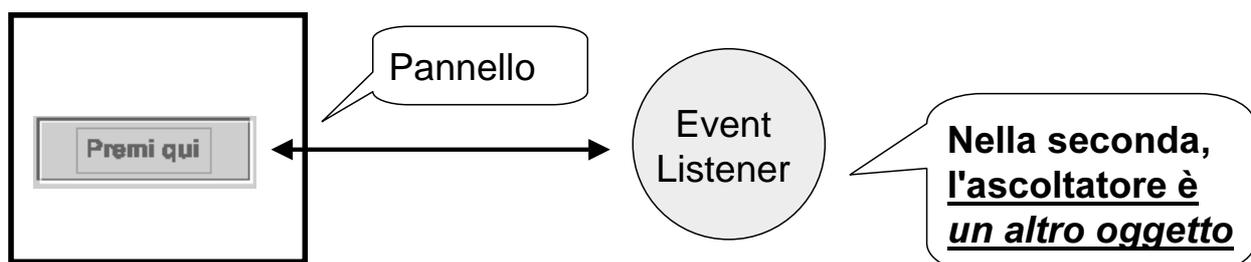
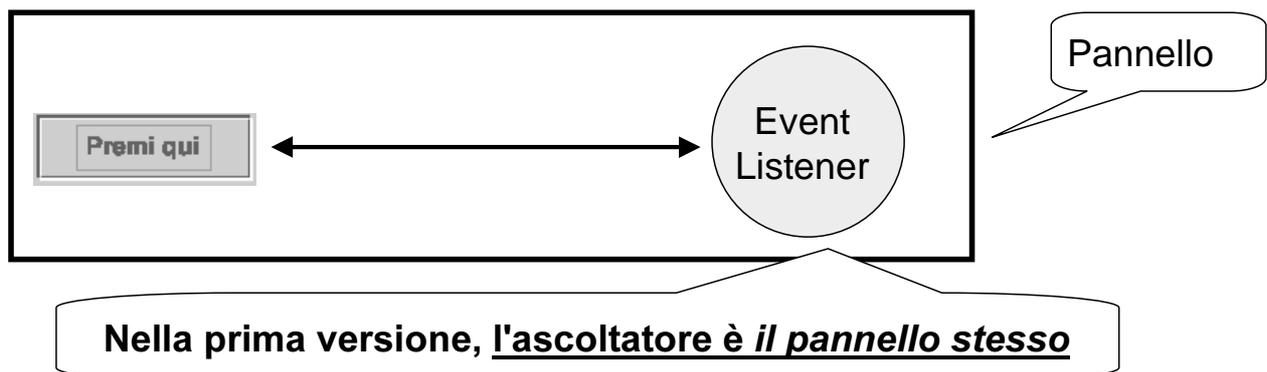
## L'ascoltatore degli eventi:

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

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

Swing - 61

# CONFRONTO



Swing - 62

# UN ESEMPIO CON DUE PULSANTI

## Scopo dell'applicazione

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

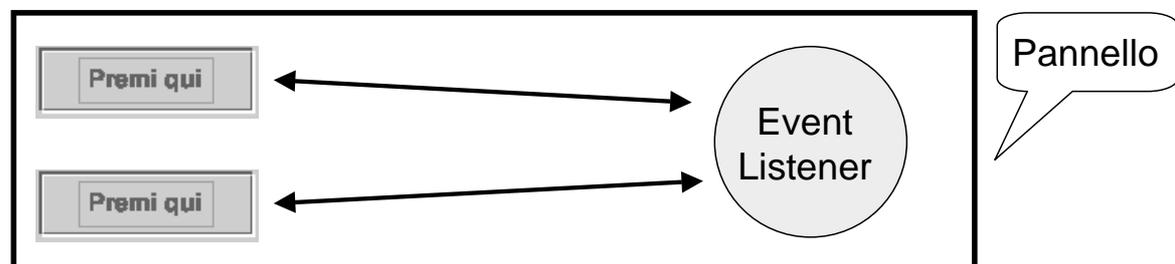
## Architettura dell'applicazione

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

Swing - 63

# UN 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

Swing - 64

# UN 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

Swing - 65

# UN 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);
}
```

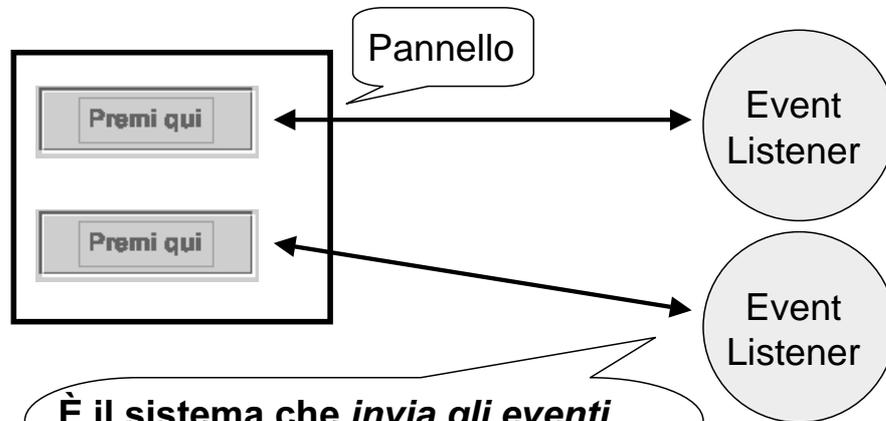
Occorre *controllare l'identità dell'oggetto* che ha generato l'evento



Swing - 66

# UN ESEMPIO CON DUE PULSANTI

**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.

Swing - 67

# UN ESEMPIO CON DUE PULSANTI

```
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 oggetti `Es9Listener` e li imposta ognuno come ascoltatore degli eventi per un pulsante, **passando a ognuno il pannello su cui agire e il colore da usare**

# UN ESEMPIO CON DUE PULSANTI

## L'ascoltatore degli eventi:

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

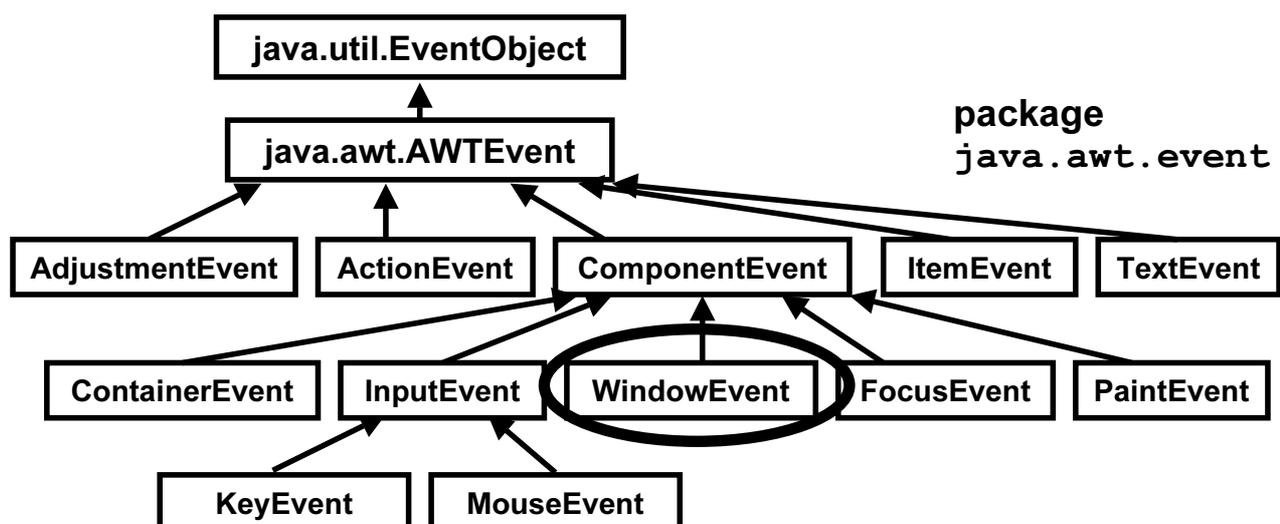
 public void actionPerformed(ActionEvent e){
 pannello.setBackground(colore);
 }
}
```

L'ascoltatore deve *ricevere come parametri* sia il pannello su cui agire sia il colore da impostare...

... che gli servono per gestire l'evento

Swing - 69

# GLI EVENTI DI FINESTRA



Le operazioni sulle finestre (finestra chiusa, aperta, minimizzata, ingrandita...) generano un `WindowEvent`

Swing - 70

## GLI EVENTI DI FINESTRA

- **Gli eventi di finestra sono gestiti dai metodi dichiarati dall'interfaccia `WindowListener`**

```
public void windowClosed(WindowEvent e);
public void windowClosing(WindowEvent e);
public void windowOpened(WindowEvent e);
public void windowIconified(WindowEvent e);
public void windowDeiconified(WindowEvent e);
public void windowActivated(WindowEvent e);
public void windowDeactivated(WindowEvent e);
```

- **Il comportamento predefinito di questi metodi va *già bene*, tranne `windowClosing()`, che *non fa uscire l'applicazione*: *nasconde solo la finestra***

Swing - 71

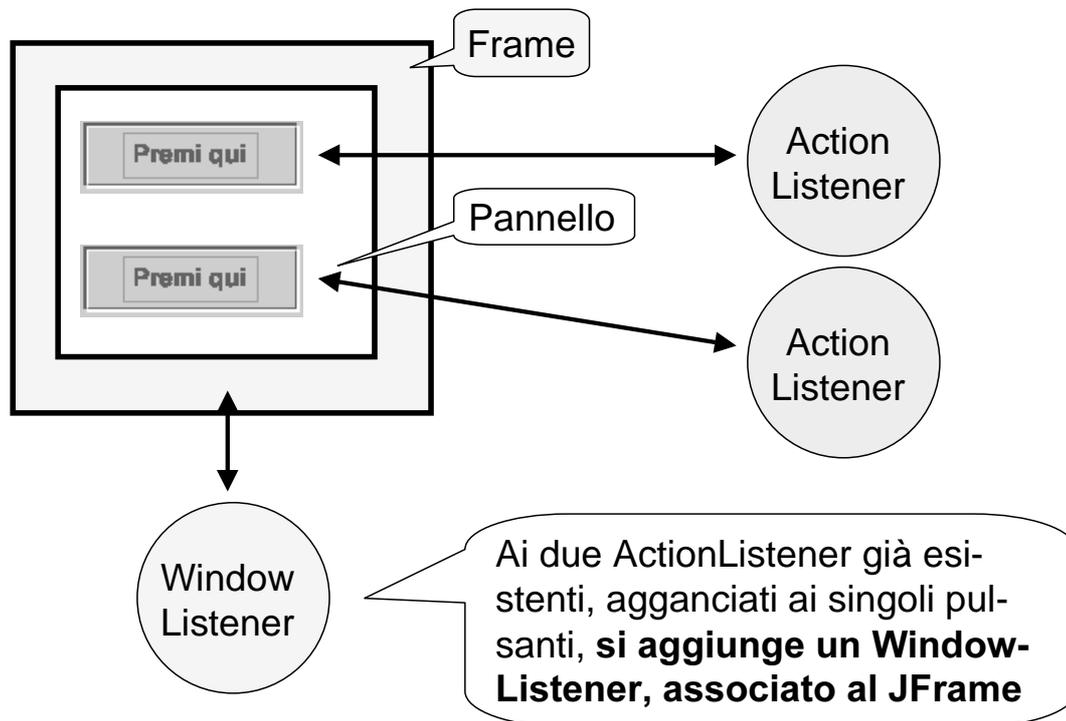
## GLI EVENTI DI FINESTRA

- **Per far sì che chiudendo la finestra del frame l'applicazione venga chiusa, il frame deve implementare l'interfaccia `WindowListener`, e ridefinire `windowClosing` in modo che invochi `System.exit()`**
- **Gli altri metodi devono essere *formalmente implementati*, ma, non dovendo svolgere compiti precisi, possono essere definiti semplicemente con un *corpo vuoto*:**

```
public void WindowOpened(WindowEvent e) { }
```

Swing - 72

## ADATTARE L'ESEMPIO



Swing - 73

## ADATTARE L' ESEMPIO

```
public class EsSwing9 {
 public static void main(String[] v) {
 JFrame f = new JFrame("Esempio 9");
 Container c = f.getContentPane();
 Es9Panel p = new Es9Panel();
 c.add(p);
 f.addWindowListener(new Terminator());
 f.pack();
 f.show();
 }
}
```

La nostra classe che implementa l'interfaccia WindowListener

Swing - 74

# ADATTARE L' ESEMPIO

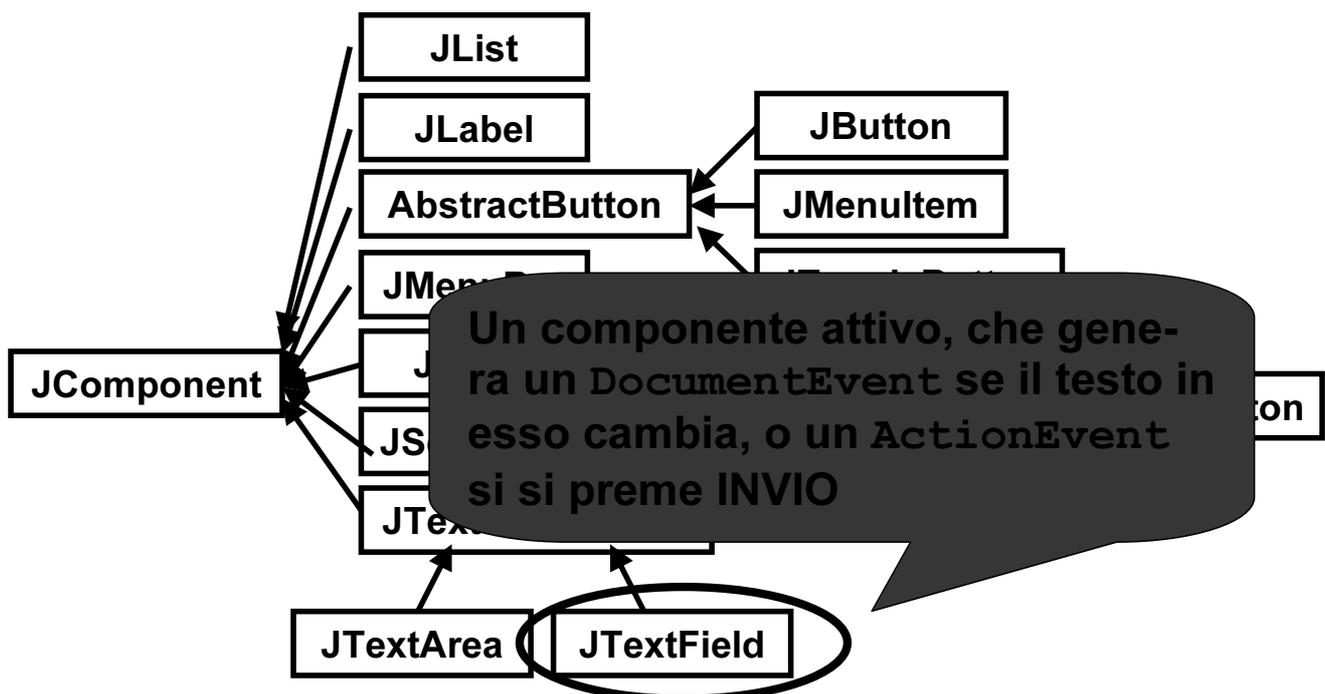
```
class Terminator implements WindowListener {
 public void windowClosed(WindowEvent e) {}
 public void windowClosing(WindowEvent e) {
 System.exit(0);
 }
 public void windowOpened(WindowEvent e) {}
 public void windowIconified(WindowEvent e) {}
 public void windowDeiconified(WindowEvent e) {}
 public void windowActivated(WindowEvent e) {}
 public void windowDeactivated(WindowEvent e) {}
}
```



ora, chiudendo la finestra si esce dall'applicazione

Swing - 75

# SWING: GERARCHIA DI CLASSI



Swing - 76

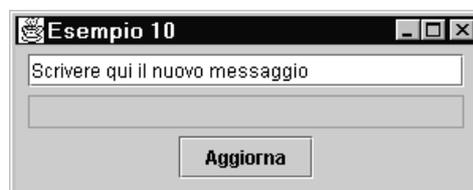
# IL CAMPO DI TESTO

- Il `JTextField` è un componente "campo di testo", 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***
- ***Se però è sufficiente registrare i cambiamenti solo quando si preme INVIO, basta gestire semplicemente il solito `ActionEvent`***

Swing - 77

## ESEMPIO

- Un'applicazione comprendente 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***
- ***L'unico evento è ancora il pulsante premuto: ancora non usiamo il `DocumentEvent`***

Swing - 78

# ESEMPIO

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();
 }
}
```

Swing - 79

# ESEMPIO

```
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);
 }
 ...
```

Larghezza preferita (caratteri)

Il secondo campo di testo non  
è modificabile dall'utente

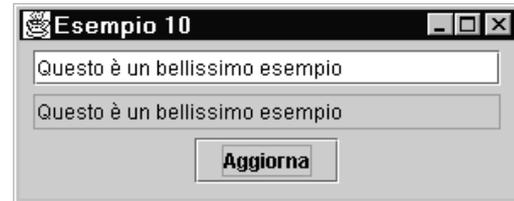
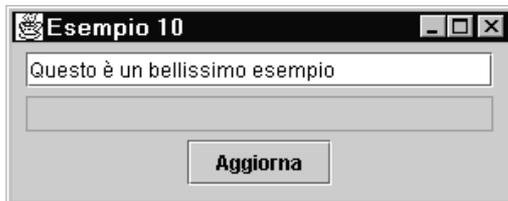
Swing - 80

# ESEMPIO

## La gestione dell'evento "pulsante premuto":

...

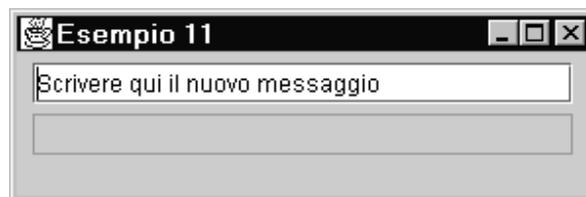
```
public void actionPerformed(ActionEvent e) {
 txt2.setText(txt1.getText());
}
}
```



Swing - 81

## UNA VARIANTE

- **Niente più pulsante, solo i due campi di testo**



- **Sfruttiamo la pressione del tasto INVIO come pulsante, quindi intercettiamo l'ActionEvent (*ancora non usiamo il DocumentEvent*)**
- **Quando si preme INVIO, il testo del secondo campo (non modificabile dall'utente) viene cambiato, e reso uguale a quello scritto nel primo**

Swing - 82

# ESEMPIO

```
class Es11Panel extends JPanel
 implements ActionListener {

 JTextField txt1, txt2;

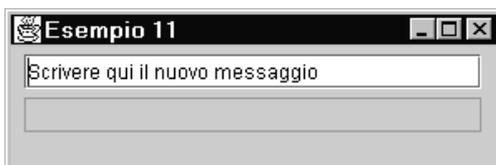
 public Es11Panel() {
 super();
 txt1 = new JTextField("Scrivere qui il testo", 25);
 txt2 = new JTextField(25); txt2.setEditable(false);
 txt1.addActionListener(this);
 add(txt1);
 add(txt2);
 }
 ...
}
```

Mettiamo un **ActionListener** in ascolto sul campo di testo **txt1**, pronto a intercettare gli eventi di azione (cioè la pressione di INVIO)

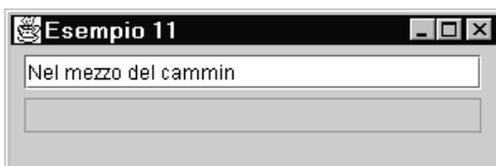
Swing - 83

# ESEMPIO

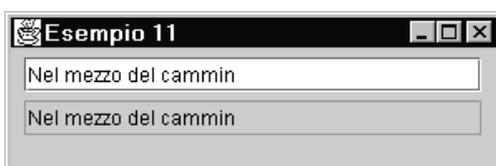
## La gestione dell'evento rimane inalterata.



La situazione iniziale...



La situazione quando si comincia a scrivere...

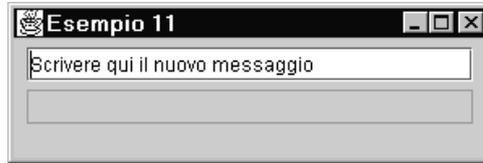


... e la situazione dopo aver premuto INVIO.

Swing - 84

## UN'ULTERIORE VARIANTE

- Sfruttiamo il concetto di DOCUMENTO che sta dietro a ogni campo di testo



- A ogni modifica del contenuto, il documento di cui il campo di testo fa parte genera un Document-Event per segnalare l'avvenuto cambiamento
- Tale evento dev'essere gestito da un opportuno DocumentListener

Swing - 85

## UN'ULTERIORE VARIANTE

- 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 in realtà è inutile, in quanto cosa sia accaduto è già implicito nel metodo chiamato; esso esiste solo per uniformità

Swing - 86

# UN'ULTERIORE VARIANTE

## Nel nostro caso:

- l'azione da svolgere in caso di inserimento o rimozione di caratteri è *identica*, quindi i due metodi  
`void insertUpdate(DocumentEvent e);`  
`void removeUpdate(DocumentEvent e);`  
saranno *identici* (purtroppo vanno comunque implementati entrambi)
- Il metodo `changedUpdate(DocumentEvent e)` è pure inutile, dato che `JTextField` non lo chiama, ma va comunque formalmente implementato.

Swing - 87

## IL CODICE DEL NUOVO ESEMPIO

```
import javax.swing.event.*;
... (il solito main) ...
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

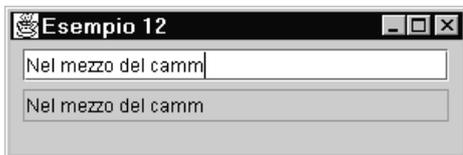
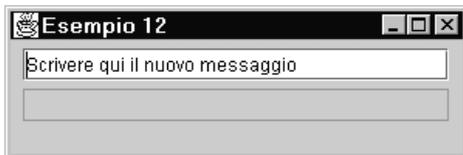
Swing - 88

# 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) {}
```

implementazione formale



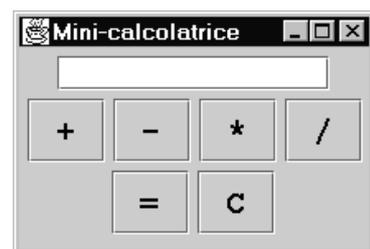
Ora, a ogni carattere inserito o cancellato, *l'aggiornamento è istantaneo e automatico*

Swing - 89

# UNA MINI-CALCOLATRICE

## 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

Swing - 90

**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...



**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

Swing - 91

## UNA MINI-CALCOLATRICE

**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();
 }
}
```

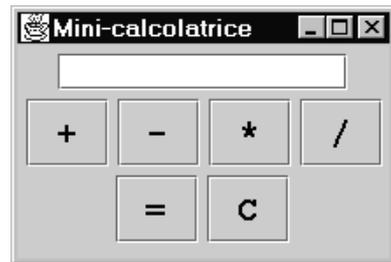
Swing - 92

# UNA MINI-CALCOLATRICE

## 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



Swing - 93

# UNA MINI-CALCOLATRICE

## 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");
 ...
 }
}
```

Swing - 94

# UNA MINI-CALCOLATRICE

## 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 (è il vero calcolatore)

Swing - 95

# UNA MINI-CALCOLATRICE

## 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

Swing - 96

# UNA MINI-CALCOLATRICE

## Il listener / calcolatore:

```
...
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;
}
}
```

Recupera il nome del pulsante premuto

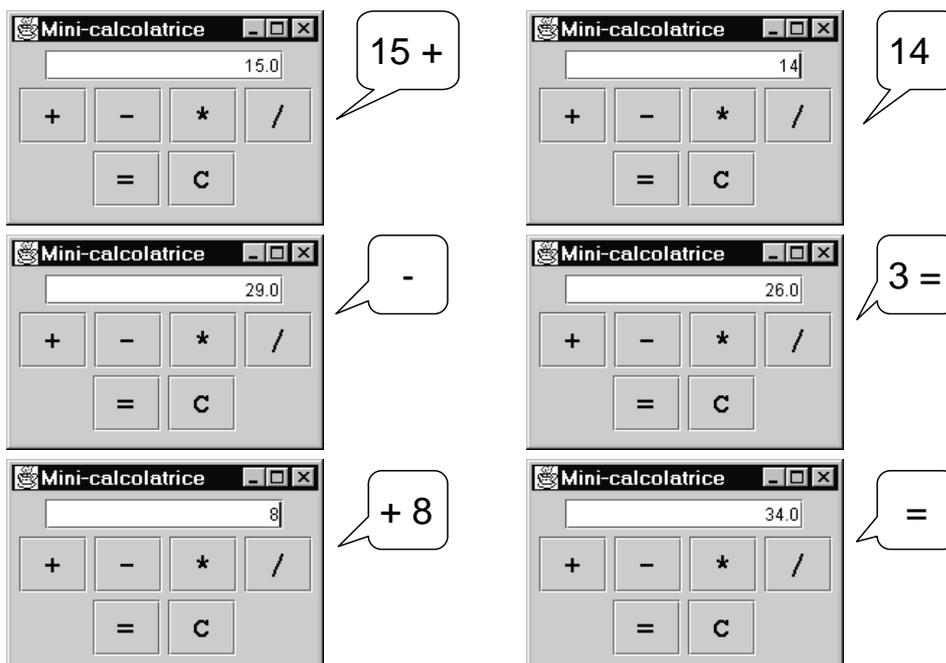
L'operazione attuale è quella da eseguire la prossima volta

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

Swing - 97

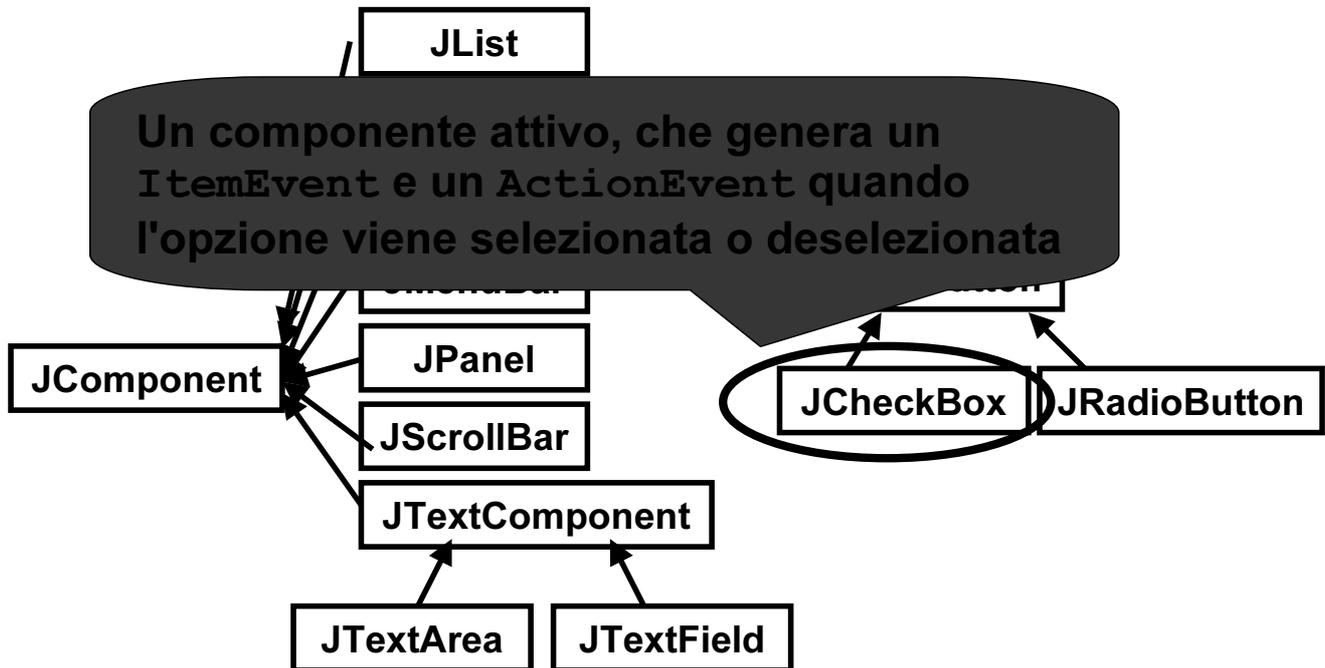
# UNA MINI-CALCOLATRICE

## Esempio di uso:



Swing - 98

# SWING: GERARCHIA DI CLASSI



Swing - 99

## IL CHECKBOX (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.

Swing - 100

# IL CHECKBOX (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.

Swing - 101

## 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 / deselegione implementando il metodo `itemStateChanged()`

Swing - 102

# ESEMPIO

```
class Es13Panel extends JPanel
 implements ItemListener {
 JTextField txt; JCheckBox ck1;

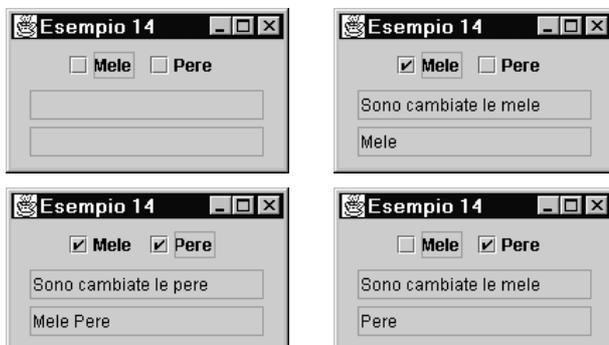
 public Es13Panel() {
 super();
 txt = new JTextField(10); txt.setEditable(false);
 ck1 = new JCheckBox("Opzione");
 ck1.addItemListener(this);
 add(ck1); add(txt);
 }

 public void itemStateChanged(ItemEvent e) {
 if (ck1.isSelected()) txt.setText("Opzione attivata");
 else txt.setText("Opzione disattivata");
 }
}
```

Swing - 103

## 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

Swing - 104

## 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);
 }
 ...
}
```

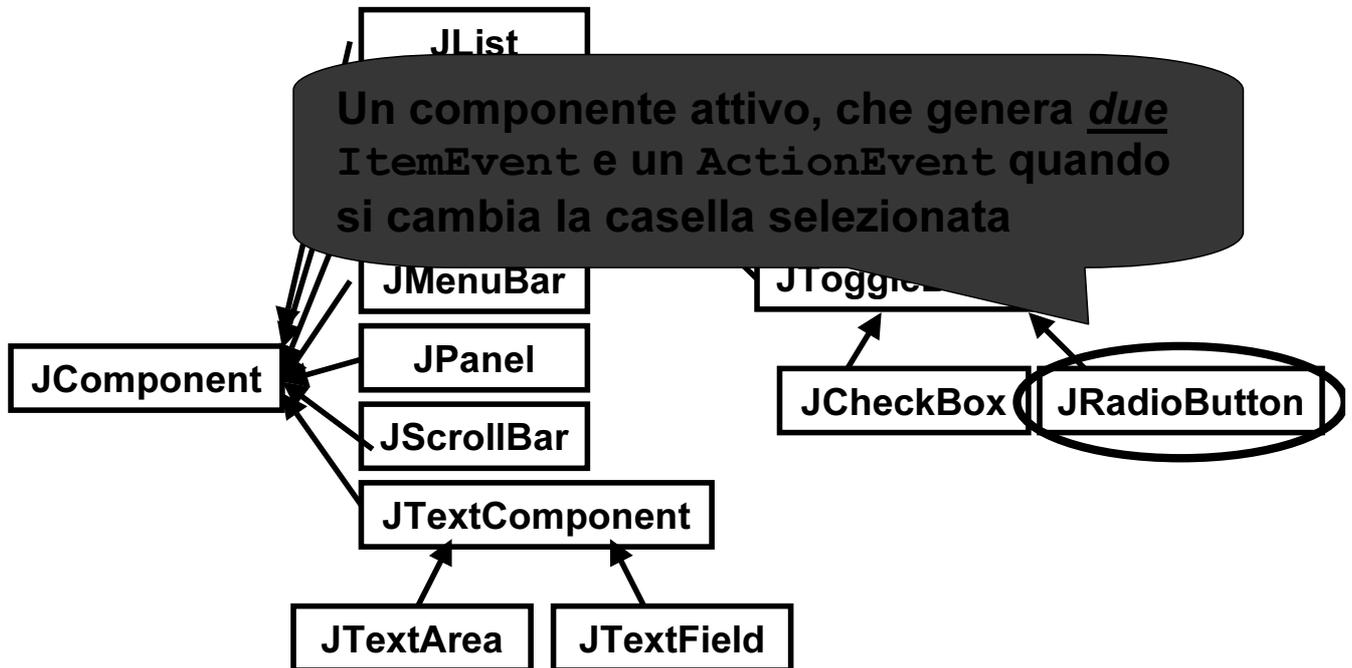
Swing - 105

## 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);
}
}
```

Swing - 106

# SWING: GERARCHIA DI CLASSI



Swing - 107

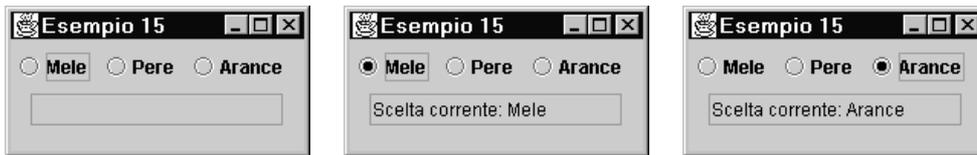
## IL RADIOBUTTON

- 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, e 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

Swing - 108

## 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) perché ogni cambio di selezione ne genera uno solo (a fronte di *due* ItemEvent), il che semplifica la gestione.

Swing - 109

## 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);
 }
}
```

Swing - 110

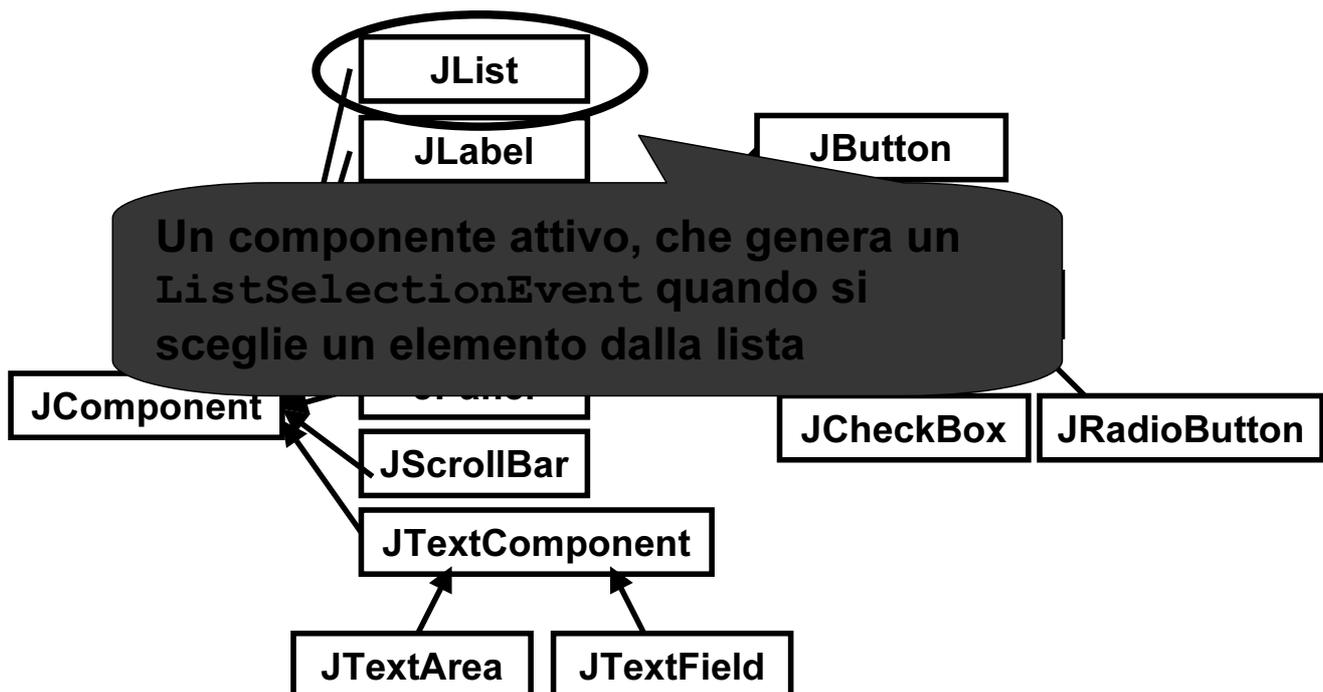
# ESEMPIO

```
...

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

Swing - 111

## SWING: GERARCHIA DI CLASSI



Swing - 112

## LA LISTA JList

- Una `JList` è una *lista di valori* fra cui si può sceglierne uno o più
- Quando si sceglie una voce si genera un evento `ListSelectionEvent`, gestito da un `ListSelectionListener`
- Il listener deve implementare il metodo `void valueChanged(ListSelectionEvent)`
- Per recuperare la/e voce/i scelta/e si usano `getSelectedValue()` e `getSelectedValues()`

Swing - 113

## ESEMPIO

- Un'applicazione con una lista e un campo di testo che riflette la selezione corrente



- Per intercettare le selezioni occorre gestire il `ListSelectionEvent`
- Di norma, `JList` *non* mostra una barra di scorrimento verticale: se la si vuole, va aggiunta a parte

Swing - 114

# ESEMPIO

## Il codice:

```
class Es16Panel extends JPanel
 implements ListSelectionListener {
 JTextField txt; JList list;
 public Es16Panel() {
 super();
 txt = new JTextField(15); txt.setEditable(false);
 String voci[] = {"Rosso", "Giallo", "Verde", "Blu"};
 list = new JList(voci);
 list.addListSelectionListener(this);
 add(list); add(txt);
 }
 ...
}
```



Swing - 115

# ESEMPIO

## Il codice:

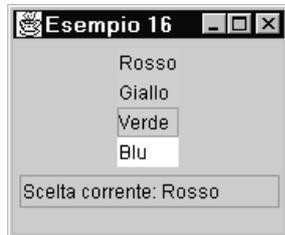
```
...
public void valueChanged(ListSelectionEvent e) {
 String scelta = (String) list.getSelectedValue();
 txt.setText("Scelta corrente: " + scelta);
}
}
```

Restituisce la prima voce selezionata (come **Object**, quindi occorre un cast)

Swing - 116

## VARIANTE

Con gli usuali tasti **SHIFT** e **CTRL**, sono possibili anche *selezioni multiple*:

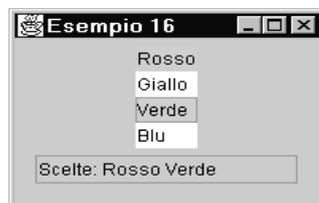


- con **SHIFT** si selezionano tutte le voci comprese fra due estremi, con **CTRL** si selezionano voci sparse
- `getSelectedValue()` restituisce solo la prima, per averle tutte occorre `getSelectedValues()`

Swing - 117

## VARIANTE

Per gestire le selezioni multiple basta cambiare l'implementazione di `valueChanged()`:



```
public void valueChanged(ListSelectionEvent e) {
 Object[] scelte = list.getSelectedValues();
 StringBuffer s = new StringBuffer();
 for (int i=0; i<scelte.length; i++)
 s.append((String) scelte[i] + " ");
 txt.setText("Scelte: " + s);
}
```

Swing - 118

## ULTERIORE VARIANTE

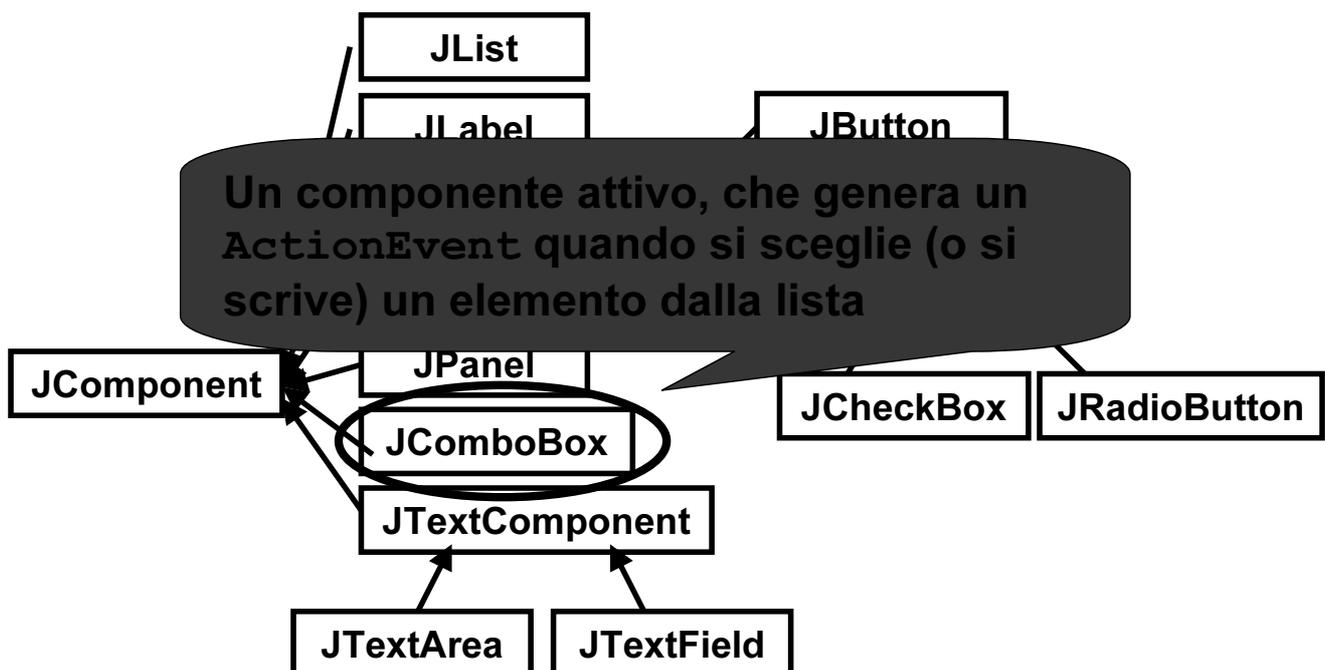
Per aggiungere una barra di scorrimento, si sfrutta un `JScrollPane`, e si fissa un numero massimo di elementi visualizzabili per la lista:

```
public Es18Panel() {
 ...
 list = new JList(voci);
 JScrollPane pane = new JScrollPane(list);
 list.setVisibleRowCount(3);
 list.addListSelectionListener(this);
 add(pane); // invece che add(list)
 add(txt);
}
```



Swing - 119

## SWING: GERARCHIA DI CLASSI



Swing - 120

# LA CASELLA COMBINATA

- Una `JComboBox` è una *lista di valori a discesa*, in cui si può o sceglierne uno, o scrivere un valore diverso
  - combina il campo di testo con la lista di valori
- Per configurare l'elenco delle voci proposte, si usa il metodo `addItem()`
- Per recuperare la voce scelta o scritta, si usa `getSelectedItem()`
- Quando si sceglie una voce o se ne scrive una nuova, si genera un `ActionEvent`

Swing - 121

## ESEMPIO

- Un'applicazione con una casella combinata e un campo di testo che riflette la selezione



- Ponendo `setEditable(true)`, si può anche scrivere un valore diverso da quelli proposti:



Swing - 122

# ESEMPIO

```
class Es19Panel extends JPanel implements
 ActionListener {
 JTextField txt; JComboBox list;

 public Es19Panel() {
 super();
 txt = new JTextField(15);
 txt.setEditable(false);
 list = new JComboBox();
 list.setEditable(true);
 list.addItem("Rosso"); list.addItem("Giallo");
 list.addItem("Verde"); list.addItem("Blu");
 list.addActionListener(this);
 add(list);
 add(txt);
 }
 ...
}
```

Consente non solo di scegliere una delle voci proposte, ma anche di scrivere una diversa

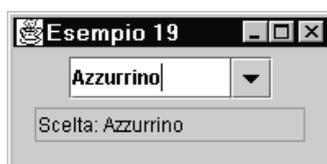
Swing - 123

# ESEMPIO

## La gestione dell'evento:

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

Recupera la voce selezionata o scritta dall'utente (in questo caso, quando si preme INVIO)



Swing - 124

## 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 (Layout 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

Swing - 125

## 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  $m \times n$
- `GridBagLayout`, che dispone i componenti in una griglia  $m \times n$  *flessibile*
  - righe e colonne a dimensione variabile
  - molto flessibile e potente, ma difficile da usare

....

Swing - 126

# 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))
```

Swing - 127

## LO STESSO PANNELLO CON...

... **FlowLayout**...



... **GridLayout** ...  
(griglia 2 x 1)



... **BorderLayout** ...  
(nord e sud)



... e senza alcun layout.  
(posizioni a piacere)



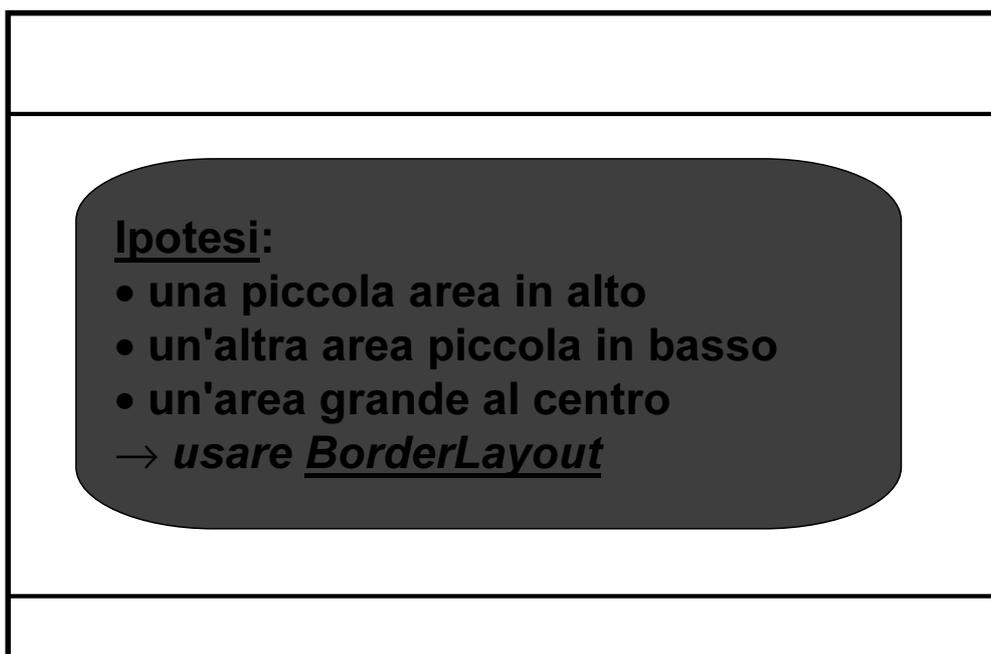
Swing - 128

# 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*

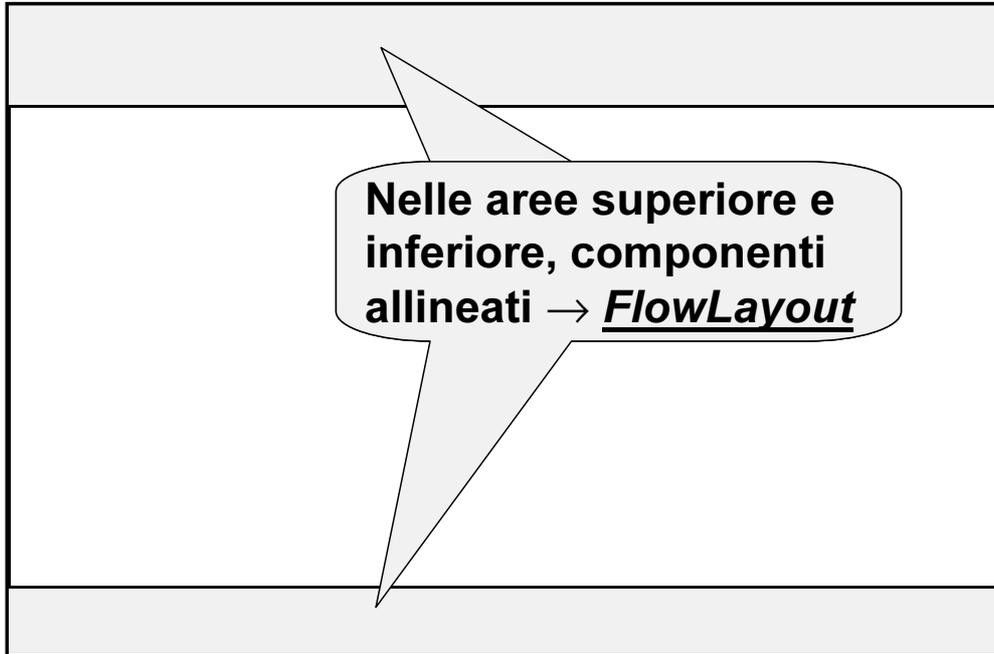
Swing - 129

## ESEMPIO



Swing - 130

## ESEMPIO



Swing - 131

## ESEMPIO



Swing - 132