GUI In Java

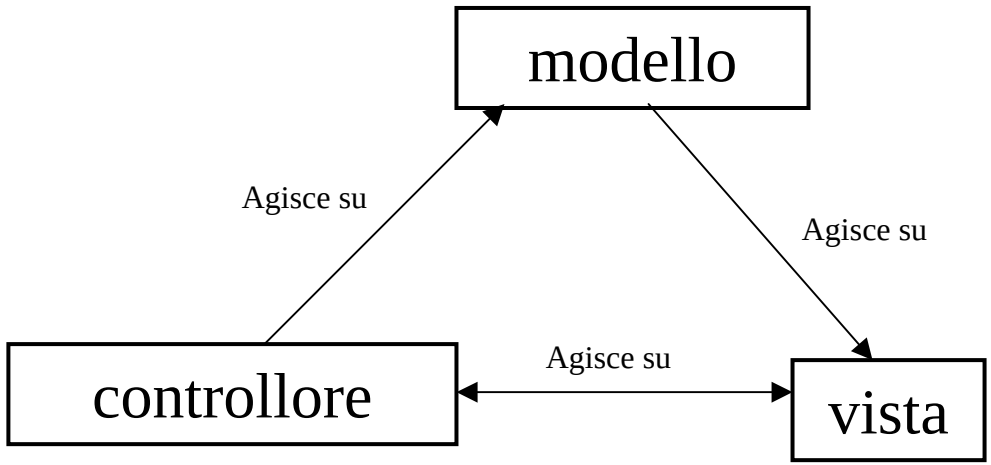
- **Lo sviluppo di GUI in Java e' basato su due package**
- **java.awt**
  - **definito in JDK 1.1**
- **javax.swing**
  - **incluso definitivamente come estensione standard di Java2**
- **Sono parte di Java Foundation Classes (JFC)**
- **Se siete pratici di X-Window ritroverete anche qui i concetti base**
- **Vedremo swing**

Swing

- Estende molte delle classi di AWT
- Swing e’ 100% Java
- Implementa il concetto di pluggable look’n’feel
- Architettura modello-vista-controllore
  - concetto non nuovo ereditato da Smalltalk

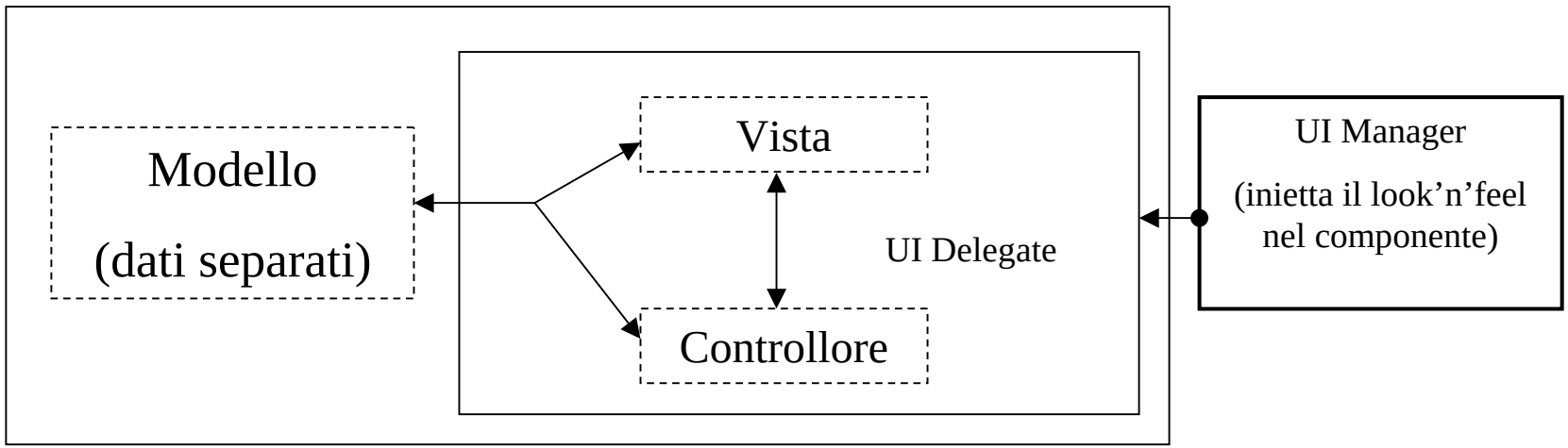
Architettura modello-vista-controllore

- Il modello contiene i dati che definiscono il componente
- La vista crea la rappresentazione grafica/visuale del componente a partire dai dati nel modello
- Il controllore gestisce l'interazione con l'utente e modifica il modello e la vista se necessario/richiesto dall'utente



Architettura documento-vista

- Di solito esiste una interdipendenza strettissima tra vista e controllore
- impossibile separare la classe vista dalla classe controllore
- oggetto composto vista/controllore
- SUN la chiama architettura a modello separato



Finestra

- java.lang Object
- java.awt Component
  - » Un oggetto che puo’ essere visualizzato
- java.awt Container
  - » Un componente che puo’ contenerne altri
- java.awt Window
  - » Una generica finestra senza bordo o barra del titolo
  - » Rimuove la proprieta’ di contenibilita’
- java.awt Frame
  - » Una finestra con bordo e barra del titolo
- javax.swing JFrame
  - » aggiunge caratteristiche avanzate (ha piu’ di 200 metodi)

Componenti e contenitori

- La radice del display di un'applicazione e' sempre un JFrame (o un Frame)
- Un JFrame contiene sempre un JRootPane (tuuta l'area sotto la barra del titolo)
- Il JRootPane puo' contenere la barra dei menu' (opzionale)
- Il JRootPane contiene il layeredPane
  - oggetto di tipo JLayeredPane
  - usato per gestire layers di componenti da mandare in fronte o indietro a gruppi (layers di visualizzazione)

Componenti e contenitori (Cont.)

- Il layeredPane contiene il contentPane
  - oggetto di tipo JInternalPane
- Dentro contentPane si aggiungono altri componenti
- Esiste anche il glassPane
  - usato per i componenti che devono essere sempre di fronte(es. menu popup)

Componenti e contenitori (Cont.)

- Da un oggetto JFrame si ottengono i pannelli (Pane) visti precedentemente con appositi metodi
- JRootPane jrp = myjframe.getRootPane();
- JLayeredPane jlp = myjframe.getLayeredPane();
- Container ct = myjframe.getContentPane();
- Component cp = myjframe.getGlassPane();

Caratteristiche Base dei Componenti

- posizione
  - coordinate x y in pixels rispetto all'angolo in alto a sinistra
- nome
  - una stringa
- dimensione
  - in pixels
- colore di sfondo
- colore di primo piano
- font

Caratteristiche base dei componenti (Cont.)

- cursore
  - icona usata per il cursore quando e’ sopra l’oggetto
- abilitazione
  - se il componente e’ abilitato l’utente puo’ accedervi
- visibilita’
  - se un oggetto e’ visibile allora viene disegnato sullo schermo
- validita’
  - per poter essere visualizzato un oggetto deve essere valido
  - tutti i componenti del suo layout devono essere validi

Accesso alle caratteristiche

- tutti i campi delle caratteristiche dei componenti sono privati
- si possono accedere solo tramite metodi appositi
  - finestra.setName(“La mia finestrina”);
  - System.out.println(finestra.getName());

```
if(finestra.isValid())
{
        System.out.println(“La finestra e’ valida”);
}
else
{
        System.out.println(“La finestra non e’ valida”);
}
```

Gestione di posizione e dimensione di un componente

- void setBounds(int x, int y, int width, int height)
- void setBounds(Rectangle r)
- Rectangle getBounds()
- void setSize(Dimension d)
- Dimension getSize()
- void setLocation(int x, int y)
- void setLocation(Point p)
- Point getLocation()
- ATTENZIONE: Rectangle e Point non sono oggetti grafici, sono oggetti geometrici

Toolkit

- la classe Toolkit e’ una classe astratta che permette di ottenere informazioni sull’ambiente grafico in cui si sta lavorando
  - non esiste costruttore per la classe Toolkit
  - si puo’ ottenere un’istanza di una classe concreta (derivata) interrogando un componente

```
JFrame j = new JFrame(“Per vedere il toolkit”);
Toolkit kit = j.getToolkit();
System.out.println(kit.toString());
Dimension scrSize = kit.getScreenSize();
System.out.println(scrSize.toString());
```

Colori e font di un componente

- void setBackgraoud(Color c)
- Color getBackground()
- void setForeground(Color c)
- Color getForeground()
- void setCursor(Cursor cu)
- Cursor getCursor()
- void setFont(Font f)
- Font getFont()

Colori

- i colori sono definiti con i valori RGB
- Color c = new Color(int R, int G, int B)
- i colori possono essere schiariti o scuriti
  - aumento o diminuisco la luminanza
- c = c.brighter();
- c = c.darker();

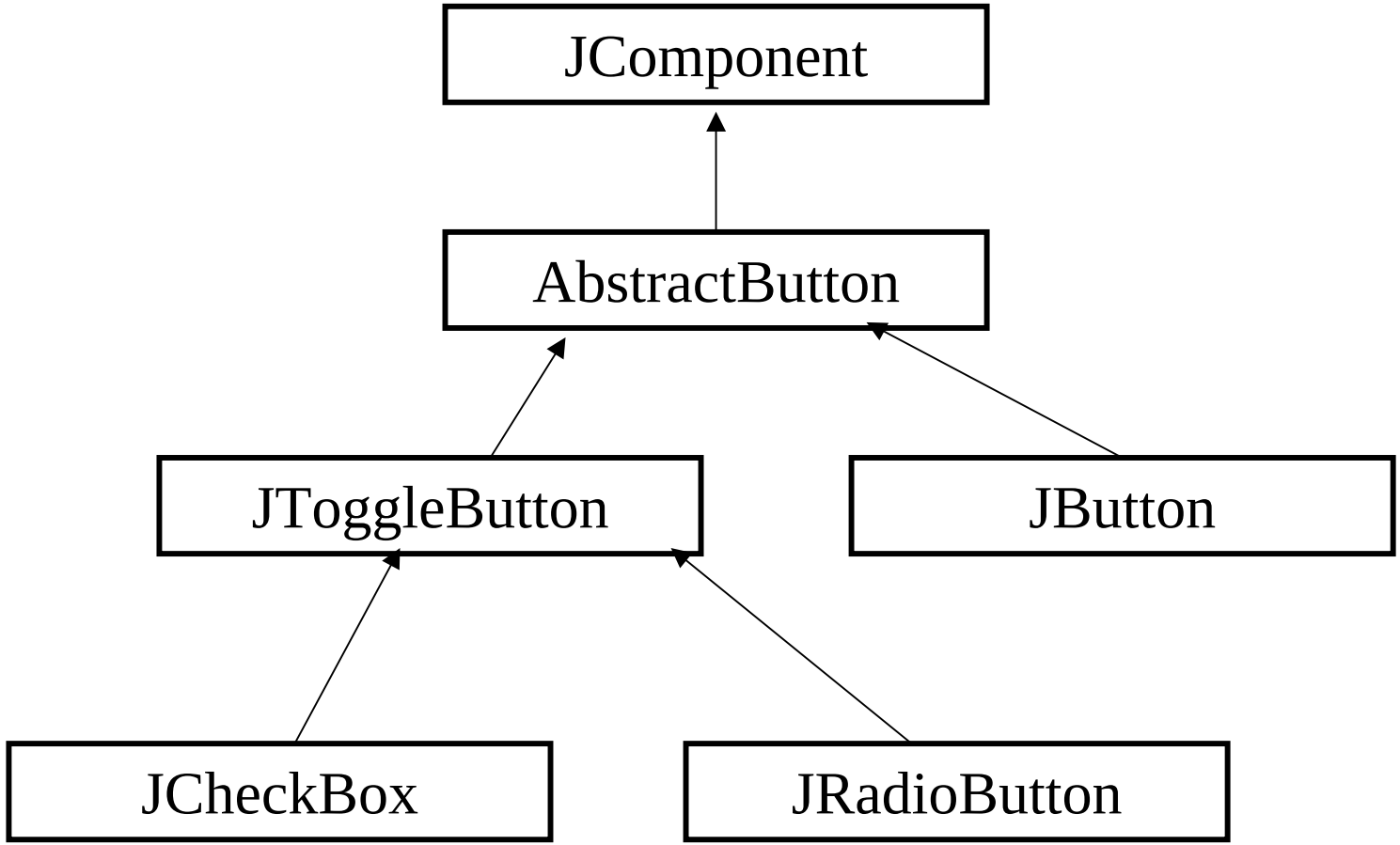
Colori di sistema

- AWT definisce una classe SystemColor
- contiene i colori che sono usati per le componenti grafiche di base nella forma di campi statici di tipo SystemColor
  - window per il colore di sfondo di una finestra
  - windowText per il colore del testo inserito in una finestra
  - menu il colore di sfondo dei menu
  - etc.

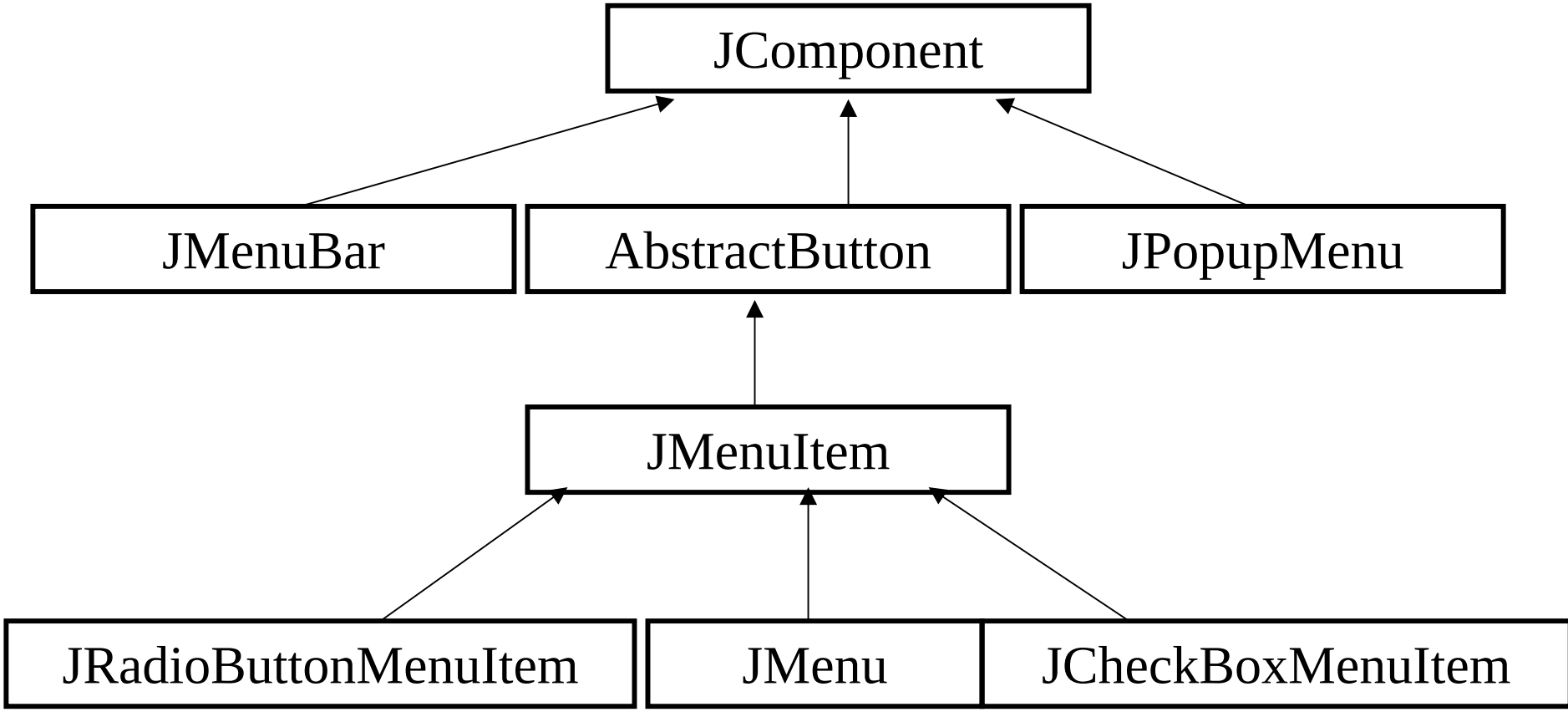
Componenti swing

- la classe base per i componenti swing e' JComponent
- Tutti i JComponent supportano
  - pluggable look'n'feel
  - tooltip
  - capacita' di scrolling tramite click and drag con il mouse
  - opzione slow-motion per il debug

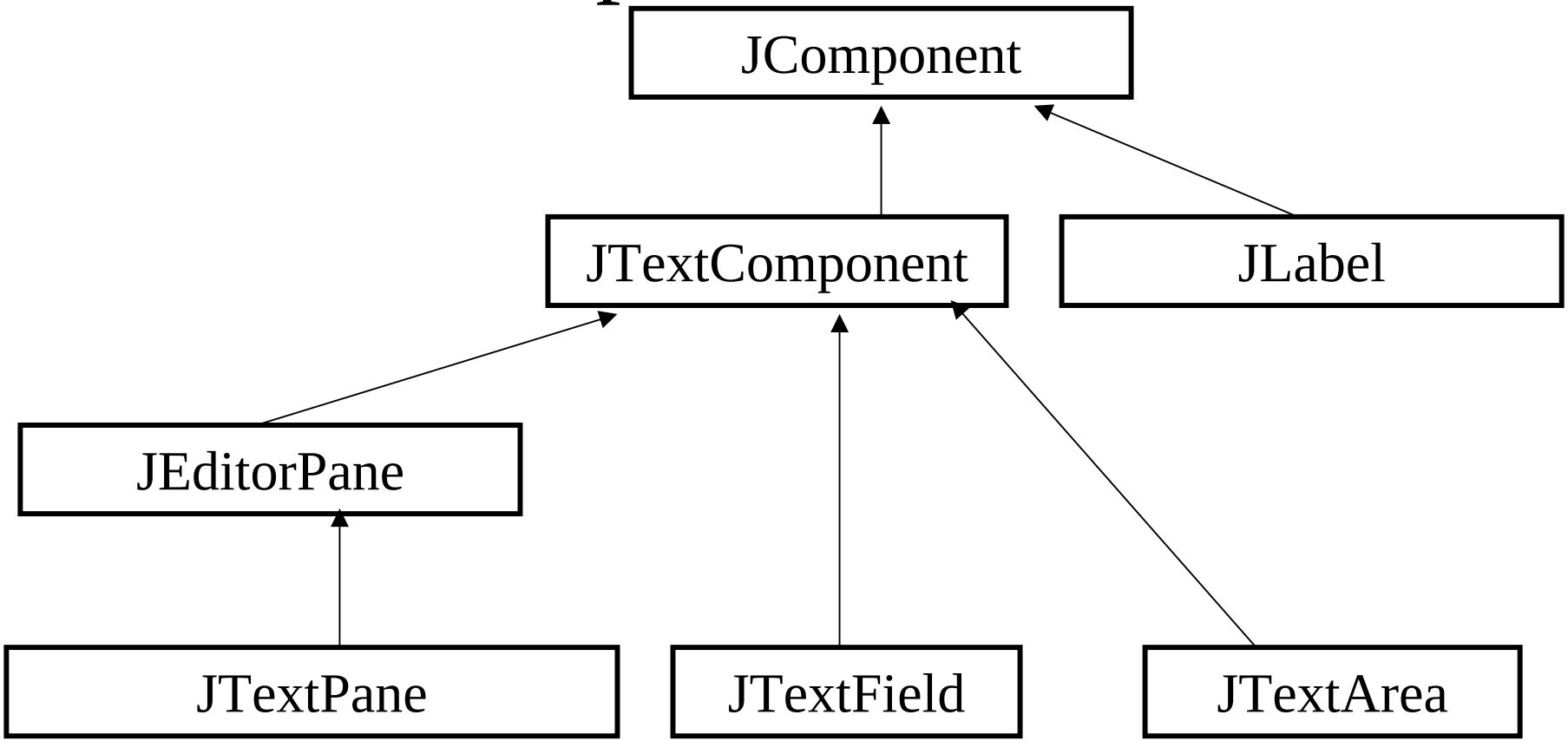
Bottoni


JComponent
AbstractButton
JToggleButton
JButton
JCheckBox
JRadioButton

Menu


JComponent
JMenuBar
AbstractButton
JPopupMenu
JMenuItem
JRadioButtonMenuItem
JMenu
JCheckBoxMenuItem

Componenti ditesto



Altri componenti di uso comune

- JPanel
  - raggruppa altri componenti
- JList
  - Una lista selezionabile
- JTable
  - una tavola selezionabile per celle, righe e colonne
- JTree
  - un albero con nodi collassabili ed espandibili

Disposizione dei componenti

- la metodologia di posizionamento dei componenti in un contenitore e' gestita da un oggetto che implementa l'interfaccia LayoutManager
- Ci sono varie classi che implementano l'interfaccia LayoutManager
- I diversi contenitori di swing hanno un LayoutManager di default

Classi implementanti LayoutManager

- FlowLayout
  - Dispone i componenti in righe successive
  - riempie ogni riga prima di cominciare la successiva
  - e’ il default per oggetti JPanel
- BorderLayout
  - pone i componenti lungo i bordi del contenitore e al centro
  - si usano le costanti BorderLayout.SOUTH BorderLayout.NORTH BorderLayout.EAST BorderLayout.WEST BorderLayout.CENTER
  - e’ il default per il contentPane di ogni JFrame, per oggetti JDialog e JApplet

Classi implementanti LayoutManager (Cont.)

- CardLayout
  - Solo il componente top e’ visibile (Come un mazzo di carte)
  - non e’ il default per nessun componente
- GridLayout
  - Dispone i componenti su righe e colonne di dimensione costante
  - il numero di righe e colonne e’ specificabile
  - non e’ il default per nessun componente

Classi implementanti LayoutManager (Cont.)

- GridBagLayout
  - Come GridLayout ma righe e colonne possono avere dimensione variabile
  - non e' il default per nessun componente
- BoxLayout
  - dispone i componenti in una riga o in una colonna
  - se necessario taglia i componenti per "farceli stare"
  - e' il default per oggetti di tipo Box

Eventi e gestione degli eventi

- Come tutti i sistemi a finestre sia AWT che swing sono basati su eventi
  - Il livello e' quello del toolkit Xt, non c'e' accesso all'event loop
- Si definiscono degli "EventListener" che catturano gli eventi e agiscono di conseguenza

Evento Pressione Bottone


```
Public class ActionListenerEsempio implements ActionListener
{
        …..
        void actionPerformed(ActionEvent theEvent)
        {
                // gestisci l'evento
        }
}
```

ActionListener

- interfaccia con metodo ActionPerformed con parametro l’evento generato
- Ad un oggetto che puo’ generare eventi (es. bottone) deve essere associato un oggetto che implementi opportunamente il metodo ActionPerformed
- L’associazione e’ fatta tramite il metodo addActionListener()

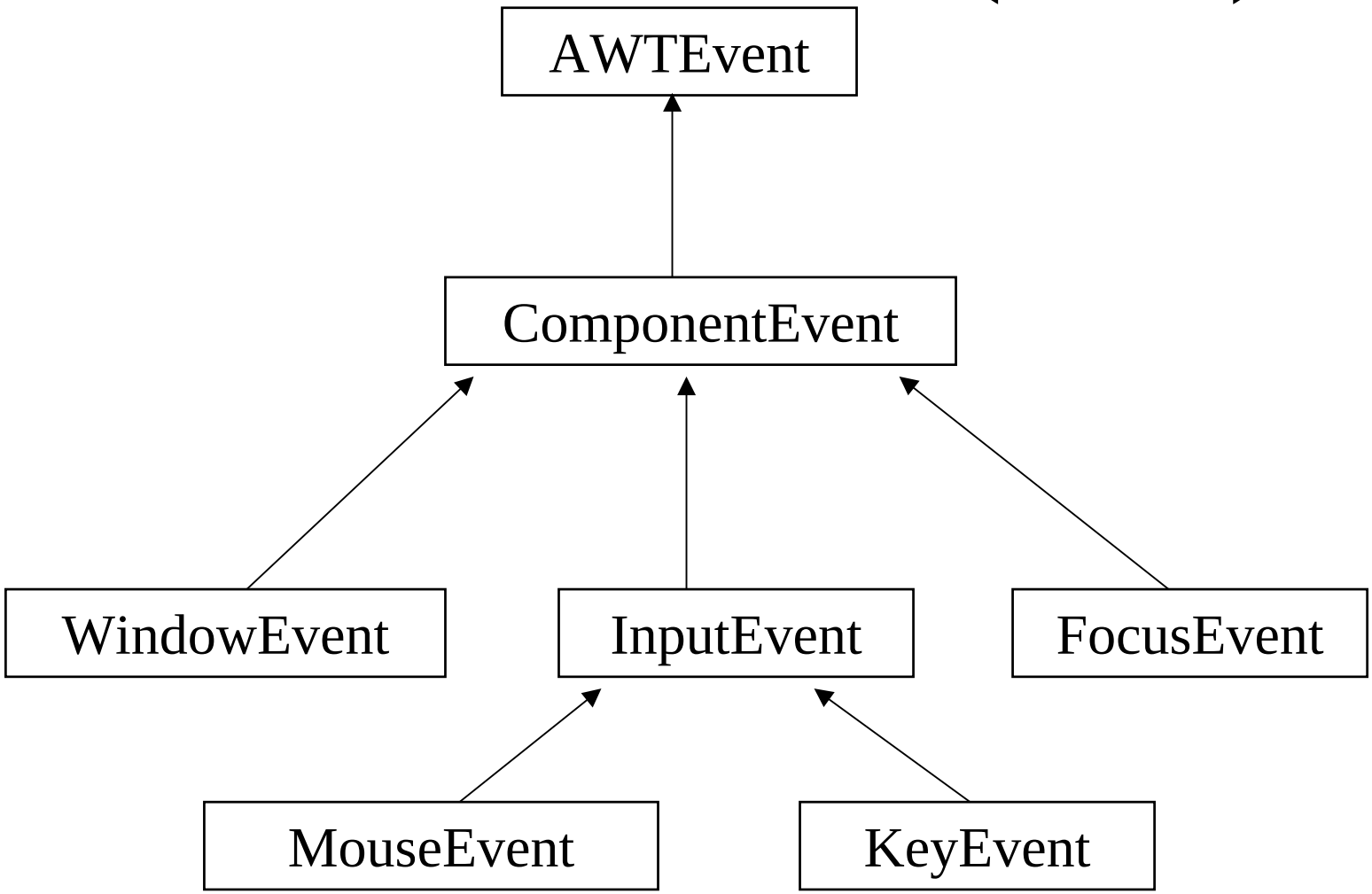
Eventi

- Un Evento contiene informazioni
  - sul tipo di evento
  - sulla sorgente dell'evento
- Eventi low-level
- generalmente gestiti automaticamente dai componenti swing
  - movimenti del mouse
  - keystrokes
  - azioni sulle finestre
- Eventi semantici
- gestione specificata dalla semantica dell'applicazione
  - pressione del bottone OK
  - selezione del menu "new object"

Eventi low-level

- FocusEvent
- MouseEvent
- KeyEvent
- WindowEvent
- altri

Eventi low-level (Cont.)



Event Mask

- X-Windows, anyone?
- Permettono di attivare la produzione di event
- Collegare un Listener attiva automaticamente gli eventi del tipo gestito dal Listener
- Si possono attivare esplicitamente per effettuare gestione diretta nell'oggetto generante (GENERALMENTE SCONSIGLIATO)

Esempio gestione diretta eventi

- Es. gestione diretta evento chiusura finestra
- nel costruttore si aggiunge
  - enableEvents(AWTEvent.WINDOW_EVENT_MASK);
- si definisce un metodo

```
protected void processWindowEvent(WindowEvent we)
{
        // gestione diretta per es. esci se evento chiusura finestra
        if(e.getID() == WindowEvent.WINDOW_CLOSING)
        {
                dispose();
                System.exit(0);
        }
        // FONDAMENTALE!!!!
        super.processWindowEvent(we);
}
```

Low-level event Listener

- esistono interfacce low-level per Listener dedicati agli eventi low-level
- estendono tutte l'interfaccia EventListener
  - WindowListener
  - MouseListener
  - MouseMotionListener
  - KeyListener
  - FocusListener

WindowListener

- definisce i seguenti metodi
  - windowOpened(WindowEvent we)
  - windowClosing(WindowEvent we)
  - windowClosed(WindowEvent we)
  - windowActivated(WindowEvent we)
  - windowDeactivated(WindowEvent we)
  - windowIconified(WindowEvent we)
  - windowDeiconified(WindowEvent we)

Adapter

- Implementare un'interfaccia richiede di implementare TUTTI i metodi da essa definiti
- se mi interessa solo uno dei metodi definiti devo comunque aggiungere anche gli altri vuoti
- per evitare questo esistono le classi Adapter
  - definiscono tutti i metodi dell'interfaccia come metodi vuoti
  - estendo un adapter ed effettuo l'overwrite solo dei metodi che mi interessano

Esempio uso low-level listener

- vogliamo implementare la condizione chiusura finestra == termina applicazione

```
public class OurFrame extends JFrame
{
        public OurFrame()
        {
                ...
                addWindowListener(new OurCloser(this));
        }
        class OurCloser extends WindowAdapter
        {
                JFrame frame = null;
                public OurCloser(JFrame theFrame)
                {
                        super();
                        frame = theFrame;
                }
                public void WindowClosing(WindowEvent we)
                {
                        System.out.println(we.toString();
                        frame.dispose();
                        System.exit(0);
                }
        }
}
```

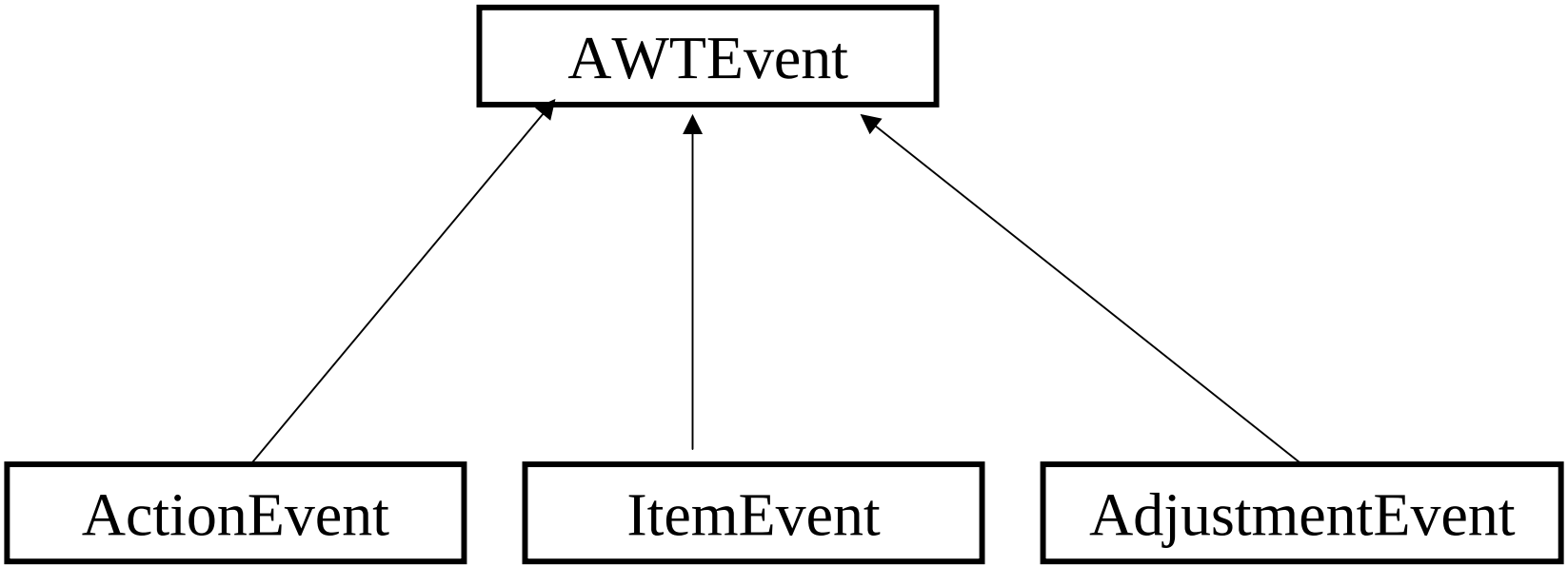
Esempio uso low-level listener (Corretto)

```
public class OurFrame extends JFrame
{
	public OurFrame()
	{
		...
		addWindowListener(new OurCloser(this));
	}
	class OurCloser extends WindowAdapter
	{
		JFrame frame = null;
		public OurCloser(JFrame theFrame)
		{
			super();
			frame = theFrame;
		}
		public void windowClosing(WindowEvent we)
		{
			System.out.println(we.toString();
			frame.dispose();
			System.exit(0);
		}
	}
}
```

Esempio uso low-level listener (Con classe anonima)

```
public class OurFrame extends JFrame
{
	public OurFrame()
	{
		...
		addWindowListener(new WindowAdapter()
		{
			public void windowClosing(WindowEvent we)
			{
				System.out.println(we.toString();
				frame.dispose();
				System.exit(0);
			}
		});
	}
}
```

Eventi Semantici



Chi Genera Eventi Semantici?

- ActionEvent
  - JButton, JToggleButton, JCheckBox
  - JMenuItem, JMenu, JCheckBoxMenuItem, JRadioButtonMenuItem
  - JTextField
- ItemEvent
  - JButton, JToggleButton, JCheckBox
  - JMenuItem, JMenu, JCheckBoxMenuItem, JRadioButtonMenuItem
- AdjustmentEvent
  - JScrollBar

Interfacce Listener per Eventi Semantici

- ActionListener
  - void actionPerformed(ActionEvent ae)
- ItemListener
  - void itemStateChanged(ItemEvent ie)
- AdjustmentListener
  - void adjustmentValueChanged(AdjustmentEvent ae)
- Una sola funzione per interfaccia, quindi non ci sono Adapters