# Programmazione Orientata agli Oggetti in Linguaggio Java

## Programmazione Grafica: Componenti

versione 1.0



**G. Mecca – Università della Basilicata – mecca@unibas.it**



## Sommario

- Componenti
- Layout
- Creazione dell'Interfaccia
- Editor Grafico dell'IDE

# Componenti

- Blocchi fondamentali dell'interfaccia
  - finestra
  - pannello
  - componente
- Finestra
  - spazio sullo schermo riservato all'applicazione grafica dal Window manager





# Componenti

- Finestra o "frame" principale
  - ogni applicazione ha una finestra principale (oggetto di tipo javax.swing.JFrame)
  - può averne altre secondarie
- Finestre secondarie
  - dipendono dalla finestra principale
  - spesso utilizzate per la visualizzazione di messaggi (es: messaggi di errore)
  - javax.swing.JDialog

# Componenti

- Finestra di dialogo modale e non modale
  - ⇨una sottofinestra si dice “modale” se impedisce l’accesso alla finestra principale finchè non viene chiusa
- La classe javax.swing.JOptionPane
  - ⇨vari metodi per produrre finestre di dialogo
  - ⇨showMessageDialog(), showInputDialog(), showConfirmDialog()





# Componenti

- Componente
  - ⇨elemento grafico disegnato in una finestra
  - ⇨con una funzione
- Principali componenti
  - ⇨bottoni (javax.swing.JButton)
  - ⇨campi per l’immissione di testo (javax.swing.JTextField)
  - ⇨etichette per la visualizzazione di messaggi (javax.swing.JLabel)

# Componenti

- Inoltre
  - ⇨nel caso di Swing, i componenti grafici non vengono disegnati direttamente sulla finestra
  - ⇨ma in un “pannello”
- Pannello
  - ⇨sfondo vuoto per il disegno di componenti
  - ⇨ogni finestra ha un pannello principale (o “content pane”)
  - ⇨javax.swing.JPanel





# Componenti

- Struttura tipica dell’interfaccia
  - ⇨un frame principale per la finestra dell’applicazione
  - ⇨uno o più pannelli, ciascuno dei quali corrisponde ad uno schermo dell’applicazione
  - ⇨uno o più componenti per pannello
  - ⇨eventuali frame secondari (es: finestre di dialogo)

# Componenti

- Analogo
  - ⇨pensiamo alla GUI come ad una scena teatrale
  - ⇨il frame rappresenta il palcoscenico (infrastruttura)
  - ⇨il pannello rappresenta lo sfondo della scena (schermo); ogni palcoscenico ha uno sfondo “predefinito” (content pane) al quale si possono sovrapporre altri sfondi per scene diverse
  - ⇨i componenti sono gli oggetti di scena (sedie, tavoli, alberi...) che compongono la scena; possono essere aggiunti o rimossi dalla scena





# Componenti

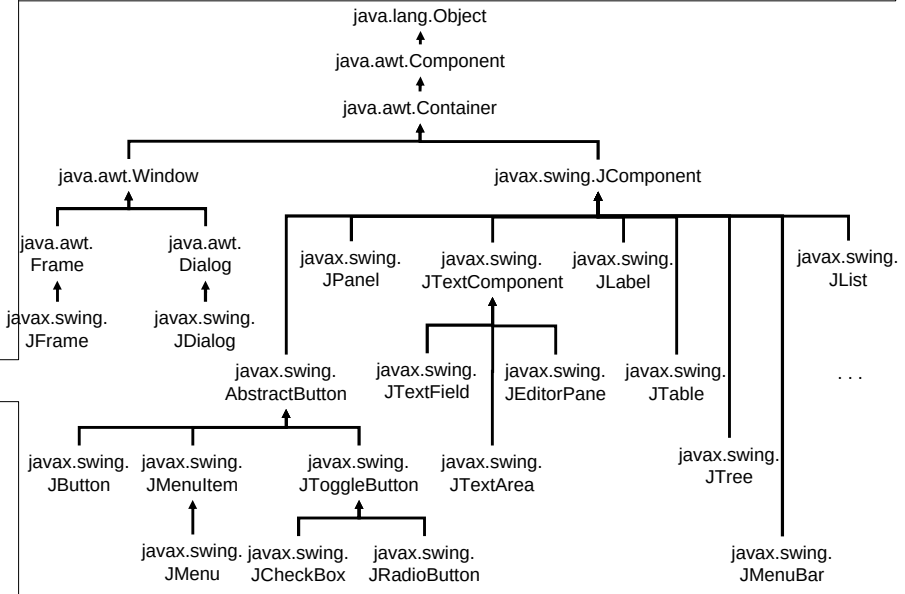
- Attenzione
  - ⇨per la maggior parte dei componenti Swing esiste una controparte AWT
  - ⇨java.awt.Frame, java.awt.Panel, java.awt.Button, java.awt.Label ecc.
  - ⇨ma si tratta di componenti deprecati
  - ⇨non è opportuno mischiare gli uni con gli altri
  - ⇨useremo solo componenti Swing

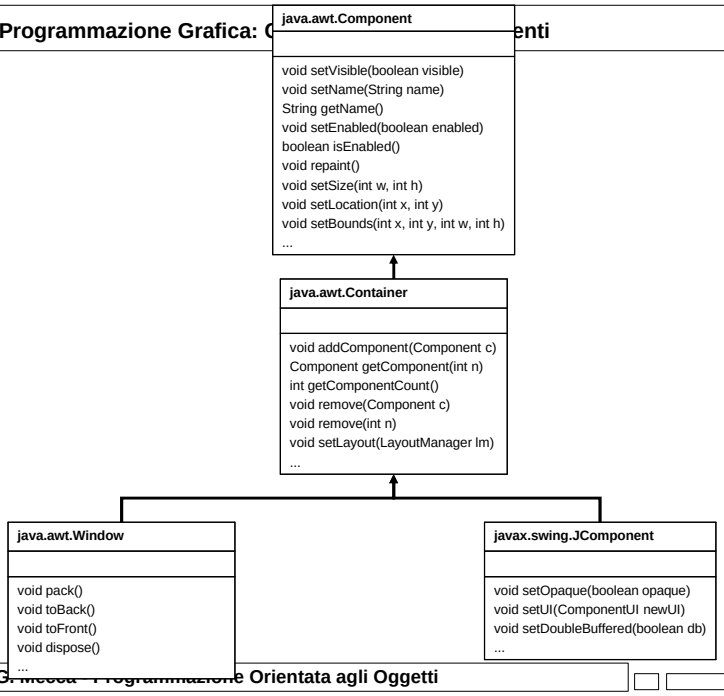
# Componenti

- Una caratteristica fondamentale di Swing
  - ⇨tutti i componenti grafici sono “contenitori”
  - ⇨ovvero possono ospitare al loro interno altri componenti (disegnati internamente)
  - ⇨ovvio per JFrame e JPanel
  - ⇨meno ovvio ma vero anche per JButton
- Gerarchia di ereditarietà complessa
  - ⇨basata sulla classe java.awt.Container













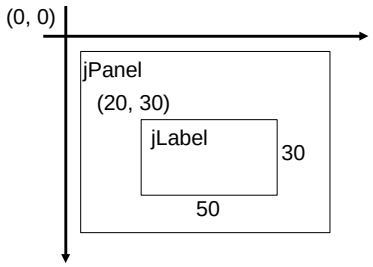

# Layout

- Un problema centrale
  - ⇨la disposizione dei componenti nel pannello
- I approccio: disposizione assoluta
  - ⇨ovvero gestione “manuale” del layout
  - ⇨per ciascun componente specifico la posizione dell’angolo in alto a sinistra
  - ⇨l’altezza e la larghezza in pixel
  - ⇨metodo setBounds(x, y, w, h)

# Layout

```
JPanel jPanel = new JPanel();
JLabel jLabel = new JLabel();
jLabel.setBounds(20, 30, 50, 30);
jPanel.add(jLabel);
```

- Il sistema di coordinate
  - ⇨relativo all'area visibile del contenitore
  - ⇨l'angolo in alto a sinistra ha coordinate 0, 0
  - ⇨l'ascissa cresce da sinistra a destra
  - ⇨l'ordinata cresce dall'alto in basso








# Layout

- Posizionamento assoluto: caratteristiche
  - ⇨consente un controllo preciso sulla posizione
  - ⇨è abbastanza semplice concettualmente
  - ⇨ma un pò macchinoso (richiede di calcolare i pixel)
  - ⇨alcune operazioni non sono banali (es: centrare nel frame un'etichetta di testo)
  - ⇨ma il problema fondamentale è il ridimensionamento del frame

# Layout

- Problema del posizionamento assoluto
  - ⇨il componente non è adeguatamente ridimensionabile
  - ⇨se il componente viene ridimensionato eccessivamente alcuni dei componenti scompaiono
  - ⇨problema: adattamento a diverse risoluzioni
  - ⇨metodo void setResizable(boolean resizable) di java.awt.Frame





# Layout

- Il approccio: gestore di layout
  - ⇨Swing fornisce classi capaci di disporre gli elementi in un pannello secondo strategie
- Idea
  - ⇨il programmatore non specifica la posizione
  - ⇨il gestore di layout dispone i componenti secondo le proprie regole
  - ⇨in caso di ridimensionamento, li ridispone in modo da cercare di mantenerli visibili

# Layout

- java.awt.FlowLayout
  - ⇨layout predefinito dei pannelli
  - ⇨distribuisce gli elementi grafici uno dopo l'altro orizzontalmente
  - ⇨se non c'è più spazio va a capo

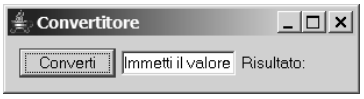






# Layout

- java.awt.BoxLayout
  - ⇨distribuisce gli elementi grafici uno dopo l'altro orizzontalmente oppure verticalmente
  - ⇨parametro di asse: BoxLayout.X_AXIS (orizzontale), BoxLayout.Y_AXIS (verticale)




# Layout

- java.awt.BorderLayout
  - ⇨layout predefinito per i frame
  - ⇨organizza l'area del contenitore in 5 zone
  - ⇨BorderLayout.NORTH, SOUTH, EAST, CENTER, WEST






# Layout

- java.awt.GridLayout
  - ⇨organizza l'area del contenitore in una matrice di N righe x M colonne
  - ⇨ciascun elemento della matrice può ospitare un componente
  - ⇨tutti gli elementi hanno la stessa dimensione

# Layout

- Layout complessi
  - ⇨CardLayout: il contenitore può contenere componenti diversi in momenti diversi
  - ⇨SpringLayout: “layout a molle”; definisce vincoli tra i bordi dei componenti e quello del contenitore
  - ⇨GridBagLayout: il più sofisticato in assoluto; variante del GridLayout in cui un componente può occupare varie celle





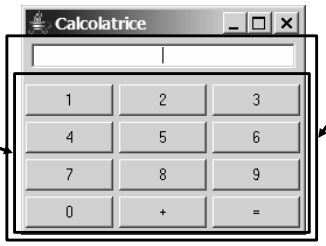
# Layout

- Gestori di layout: caratteristiche
  - ⇨hanno il grande vantaggio di garantire nella maggior parte dei casi un corretto ridimensionamento
  - ⇨sono però più complessi da usare
  - ⇨lo sviluppatore non ha grande controllo sulla posizione dei componenti
  - ⇨i layout avanzati sono potenti ma molto complessi

# Layout

- Combinare i layout
  - ⇨un espediente frequente
  - ⇨utilizzare diversi pannelli con layout diversi per ottenere la disposizione voluta

pannello secondario con GridLayout contiene i bottoni




pannello principale con BorderLayout contiene:
- JTextField
- pannello secondario





# Creazione dell'Interfaccia

- Creazione dell'interfaccia
  - ⇨alcuni passi standard
- I Passo: creazione del frame principale
  - ⇨normalmente il componente fondamentale dell'interfaccia estende JFrame
    es: class Principale extends JFrame {...}
  - ⇨o lavora in associazione con JFrame
    es: class Principale {
    private JFrame frame;

# Creazione dell'Interfaccia

- Passi successivi
  - inizializzazione dei componenti
  - normalmente in un metodo chiamato inizializza()
- II passo: creazione dei menu
  - creazione di un oggetto di tipo JMenuBar
  - aggiunta della JMenuBar al frame
  - aggiunta di vari JMenu alla JMenuBar
  - aggiunta di varie JMenuItem ai JMenu

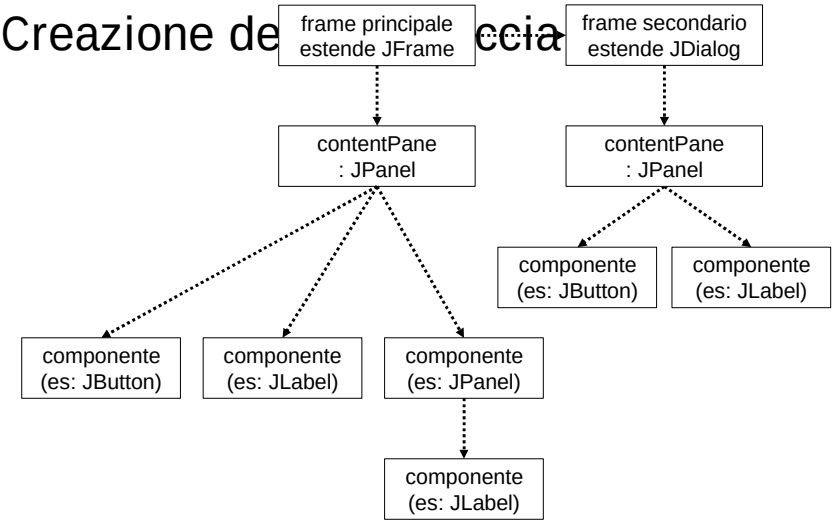




# Creazione dell'Interfaccia

- III passo: creazione del pannello
  - due possibili approccio
- I possibilità: pannello predefinito del frame
  - utilizzare direttamente il pannello di contenuto del frame ("content pane")
  - JPanel pannello = (JPanel)frame.getContentPane();
  - NOTA: il cast è necessario perchè getContentPane() restituisce un riferimento a java.awt.Container

# Creazione dell'Interfaccia



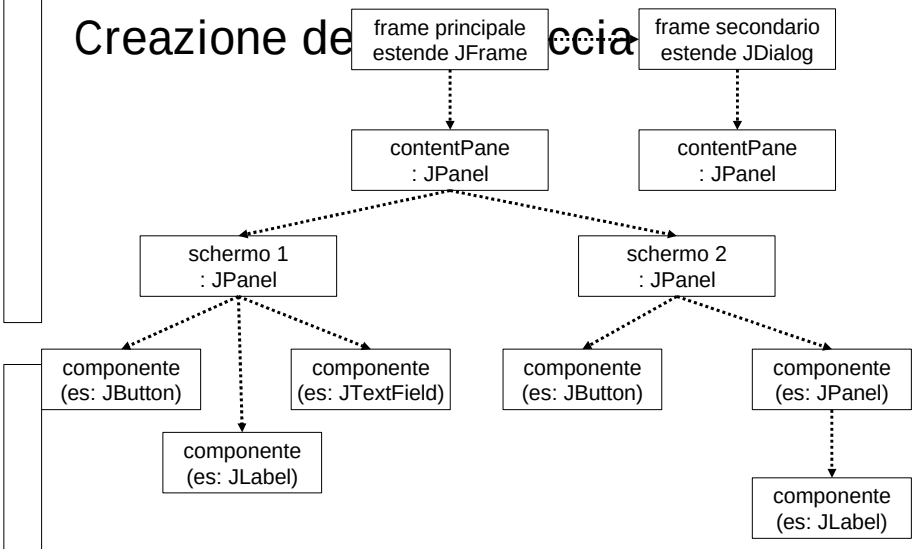





# Creazione dell'Interfaccia

- II possibilità: pannello personalizzato
  - ⇨creare un nuovo JPanel e aggiungerlo al pannello di contenuto
  - ⇨utile se successivamente è necessario cambiare il contenuto del pannello
  - ⇨es: per produrre uno schermo diverso nello stesso frame
  - ⇨JPanel pannello = new JPanel();
    frame.getContentPane().add(pannello);

# Creazione de ccia








# Creazione dell'Interfaccia

- IV passo
  - ⇨scelta della strategia di posizionamento (assoluto/gestore di layout)
  - ⇨nel caso di posizionamento assoluto è necessario rimuovere il gestore di layout predefinito
  - ⇨eseguendo il metodo setLayout(null)
  - ⇨da quel momento in poi tutti i componenti devono essere disposti in modo assoluto

# Creazione dell'Interfaccia

- V passo
  - ⇨creazione dei diversi componenti che compongono l'interfaccia (es: JButton, JLabel, JTextField...)
  - ⇨e aggiunta al pannello di ognuno, gestendone il posizionamento
  - ⇨eventualmente utilizzando pannelli aggiuntivi





# Creazione dell'Interfaccia

- VI passo
  - ⇨visualizzazione del frame
  - ⇨chiamata al metodo setVisible(true)
  - ⇨visualizza il frame nell'angolo in alto a sinistra dello schermo
  - ⇨riposizionabile con setLocation(x, y)
- Ma...
  - ⇨un frame per impostazione predefinita ha dimensione 0x0 pixel

# Creazione dell'Interfaccia

- Stabilire la dimensione del frame
  - I possibilità: utilizzare il metodo setSize(<larghezza>, <altezza>) – approccio tipico con posizionamento assoluto
  - II possibilità: chiamata al metodo pack() per aggiustare la dimensione al minimo indispensabile – approccio tipico con gestore di layout





# Creazione dell'Interfaccia

- Un dettaglio importante
  - è necessario stabilire come gestire l'evento di chiusura del JFrame
  - per impostazione predefinita, chiudendo la finestra questa viene nascosta
- Un'impostazione più naturale
  - uscire dall'applicazione

# Creazione dell'Interfaccia

- Strategia di chiusura del frame
  - ⇨metodo setDefaultCloseOperation()
  - ⇨costanti di JFrame (ereditate da javax.swing.WindowConstants)
  - ⇨HIDE_ON_CLOSE (valore predefinito)
  - ⇨EXIT_ON_CLOSE (valore "normale")
  - ⇨DO_NOTHING_ON_CLOSE
  - ⇨DISPOSE_ON_CLOSE





# Un Metodo Tipico di Inizializzazione

```
public class Principale extends javax.swing.JFrame {

   public void inizializza() {
      this.setDefaultCloseOperation(EXIT_ON_CLOSE);
      this.setTitle("Frame Principale");

      // altre operazioni (aggiunta di pannelli, componenti....)

      this.setLocation(300, 300);
      this.pack();
      // oppure: this.setSize(200, 200);
      this.setVisible(true);
   }
}
```

# Creazione dell'Interfaccia

**ATTENZIONE** al codice del main

- VII passo
  - ⇨creazione del frame nel main
  - ⇨attenzione al blocco di codice necessario

```
public static void main(String[] args){
   javax.swing.SwingUtilities.invokeLater(new Runnable() {
      public void run() {
         Principale principale = new Principale();
         principale.inizializza();
      }
   }
   );
}
```

definizione di classe interna anonima che implementa java.lang.Runnable (riguarda l'utilizzo dei thread)





# Editor Grafico dell'IDE

- Il processo di sviluppo dell'interfaccia
  - ⇨è abbastanza macchinoso
- Un utile supporto
  - ⇨editor grafico dell'interfaccia fornito dall'IDE ("GUI Editor")
  - ⇨consente di disporre "graficamente" i componenti dell'interfaccia
  - ⇨e di definirne rapidamente le proprietà

# Editor Grafico dell'IDE

- Funzionamento tipico dell'editor
  - ⇨all'avvio fornisce uno spazio vuoto corrispondente al frame da creare
  - ⇨e una "palette" di componenti (rappresentati da icone) da cui scegliere quelli da disporre
  - ⇨è possibile trascinare l'icona dei componenti nello spazio del frame
  - ⇨e poi stabilirne le proprietà attraverso una finestra apposita





# Editor Grafico dell'IDE

- Funzionamento tipico dell'editor (cont.)
  - ⇨le scelte dell'utente vengono salvate in un file di metainformazioni
  - ⇨l'IDE genera automaticamente il codice Java per la disposizione degli elementi
- Attenzione
  - ⇨non è possibile intervenire direttamente sul codice generato perchè le modifiche andrebbero perdute

# Editor Grafico dell'IDE

>> convertitore con NetBeans (interfaccia)

- Vantaggio
  - ⇨il ciclo di sviluppo si accorcia decisamente
- Svantaggio
  - ⇨poco controllo sul codice generato
- Un esempio
  - ⇨l'editor grafico di NetBeans
  - ⇨sviluppiamo l'interfaccia di un convertitore di voti da 30mi a 110mi





# Editor Grafico dell'IDE

- Struttura della classe generata
  - ⇨estende JFrame (ma è possibile anche generare JPanel)
  - ⇨tutti i componenti utilizzati sono proprietà della classe
  - ⇨i componenti vengono aggiunti nel metodo di inizializzazione
  - ⇨il main contiene il codice per la gestione dei thread e visualizza il frame

# Editor Grafico dell'IDE

- Gestione degli eventi
  - ⇨in questa forma semplificata, per ogni evento aggiunto ad un componente, l'IDE genera un metodo
  - ⇨attraverso il quale è possibile specificare le operazioni da svolgere in corrispondenza dell'evento
- Un evento importante
  - ⇨evento di "azione" (ActionEvent)

>> convertitore con NetBeans (logica applicativa)





# Editor Grafico dell'IDE

- Nota
  - ⇨l'applicazione creata viola tutte le regole architetturali viste fino a questo punto
  - ⇨perchè interfaccia, controllo e logica applicativa sono tutte concentrate in un'unica classe
  - ⇨si tratta però di un idioma abbastanza utilizzato nella programmazione Swing (>>)

# Riassumendo

- Componenti
- Layout
- Creazione dell'Interfaccia
- Editor Grafico dell'IDE