Interfacce grafiche

- Una GUI (Graphic User Interface) contiene vari **componenti**: bottoni, etichette, immagini...
  - Alcuni componenti ne contengono altri (ad esempio le finestre), e sono detti **contenitori**.
  - Inoltre: **gestori di eventi** (regolano le interazioni con l'utente) e **di layout** (regolano le posizioni).
- Le interfacce grafiche sono gestite dai package delle JFC (Java Foundation Classes), in specie AWT (**`java.awt`**) e Swing (**`javax.swing`**).

Interfacce grafiche

- AWT (Abstract Windowing Toolkit): un insieme di classi (cuore delle JFC) che fornisce gli strumenti per le GUI di applet e applicazioni
- Swing estende AWT (senza rimpiazzarlo)
  - In particolare serve a rendere l'aspetto grafico uniforme in modo indipendente dalla piattaforma
- Oltre ad AWT e Swing, le JFC comprendono:
  - Java 2D e Java 3D
  - Drag and Drop
  - Accessibility (per utenti con difficoltà)

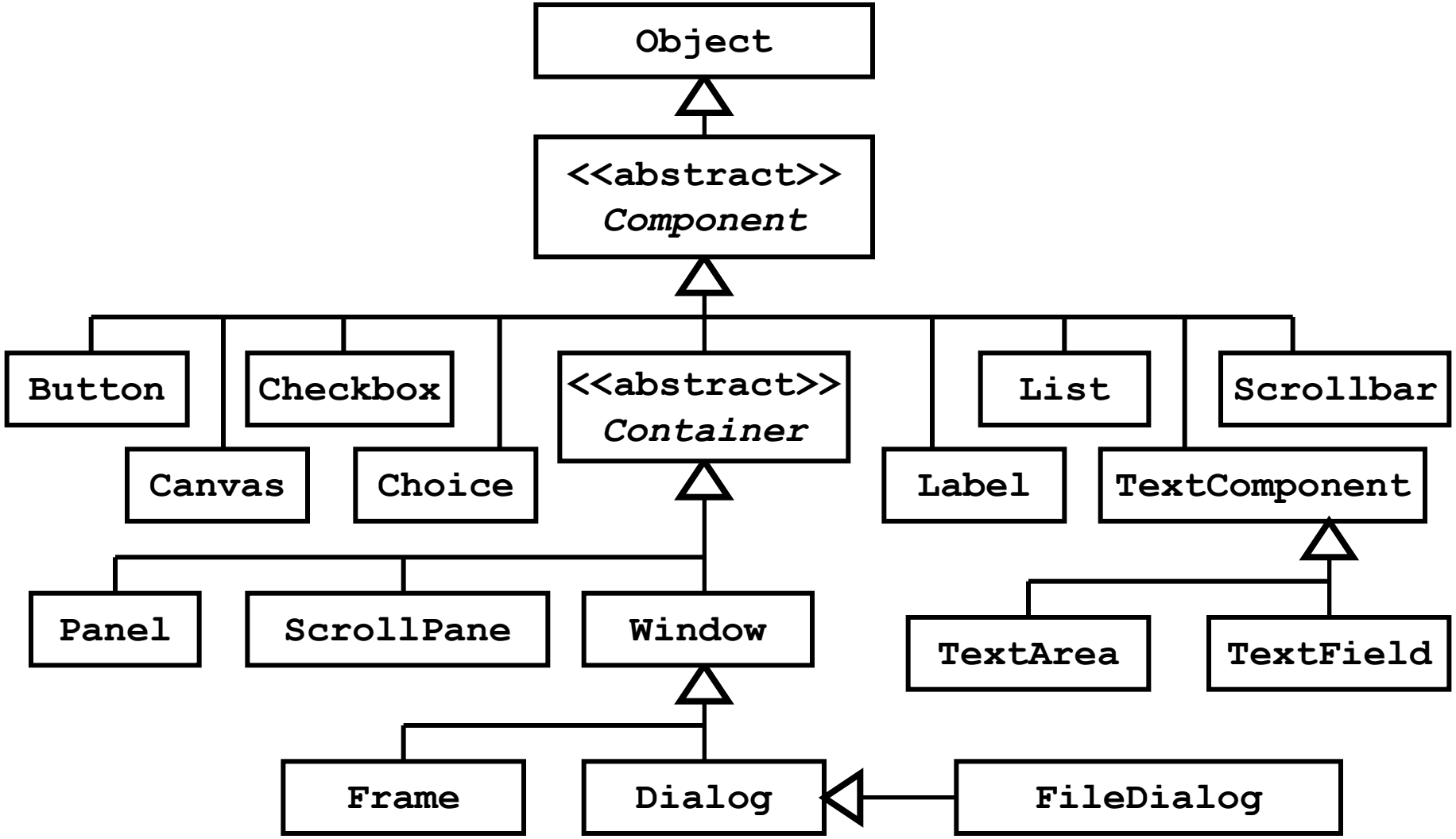
AWT

- AWT supporta il paradigma WYSIWYG (What You See Is What You Get)
- Contiene tante classi, in particolare:
  - La classe astratta **`Component`** è la base di AWT.
  - La classe astratta **`MenuComponent`** serve a gestire i menu (in modo diverso dagli altri componenti)
  - L'interfaccia **`LayoutManager`** gestisce il layout.

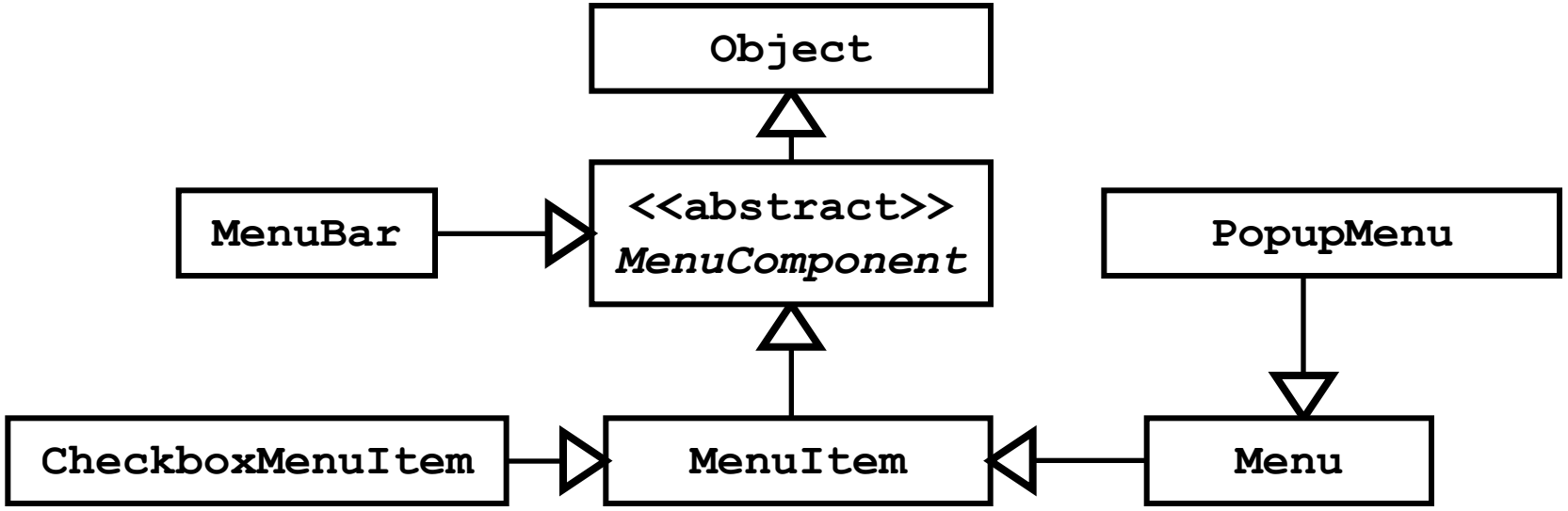
AWT

- Tutte le classi di AWT sono di tipo:
  - Grafica: disegnare figure, visualizzare immagini, selezionare carattere, impostare i colori,..
  - Componenti: pulsanti, campi testo, *menù*, barre...
  - Layout Manager: disposizione sullo schermo di componenti dentro a contenitori
  - Event Manager: che fare se si preme un pulsante, si muove il mouse, ... (vedi **`java.awt.event`**)
  - Trattamento immagini: gestione di formati (vedi: **`java.awt.image`**)

Gerarchia dei componenti AWT



Gerarchia dei componenti AWT

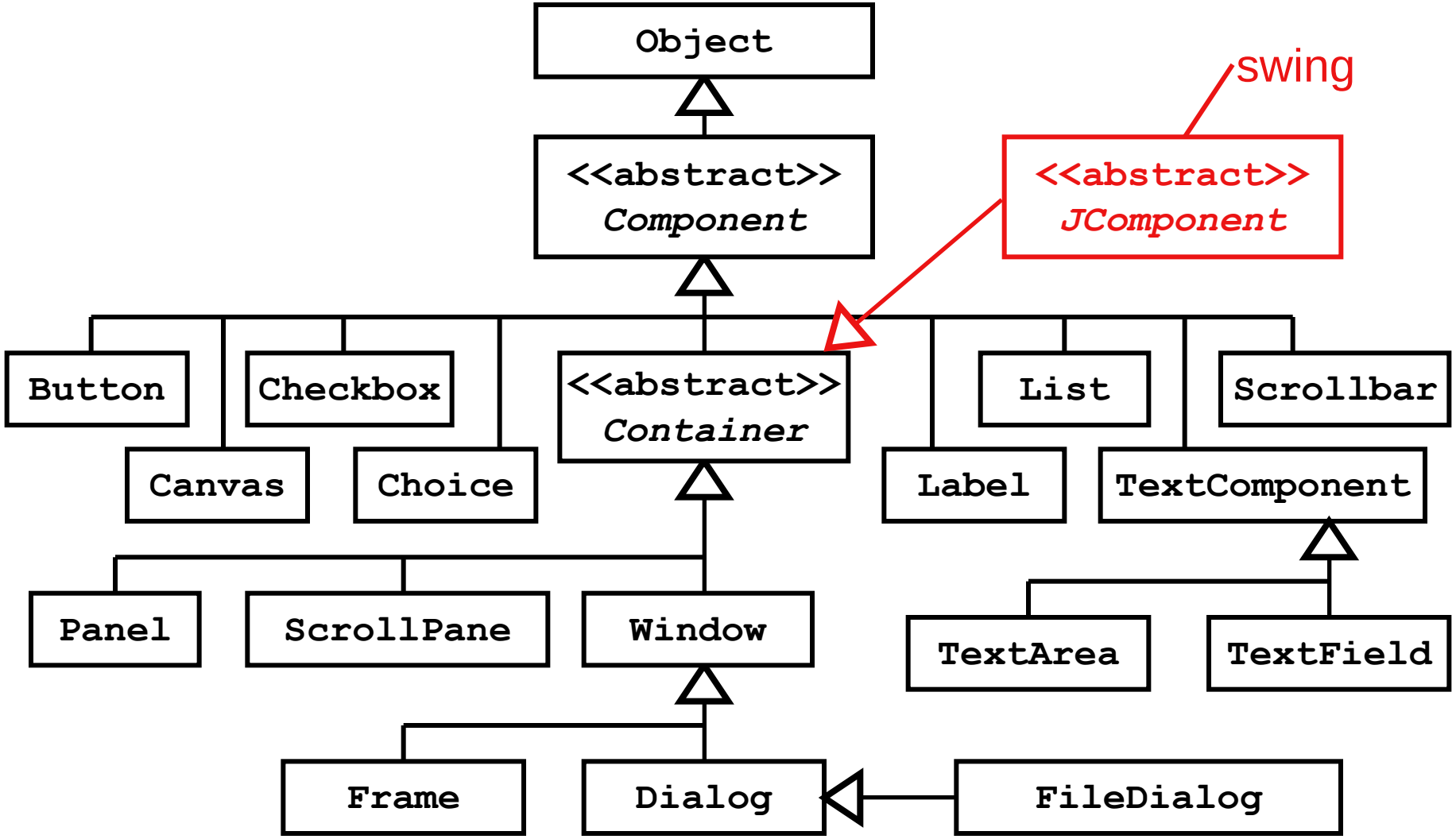


- I menu hanno una gerarchia separata rispetto agli altri componenti perché hanno delle restrizioni (ad esempio meno libertà nei colori)

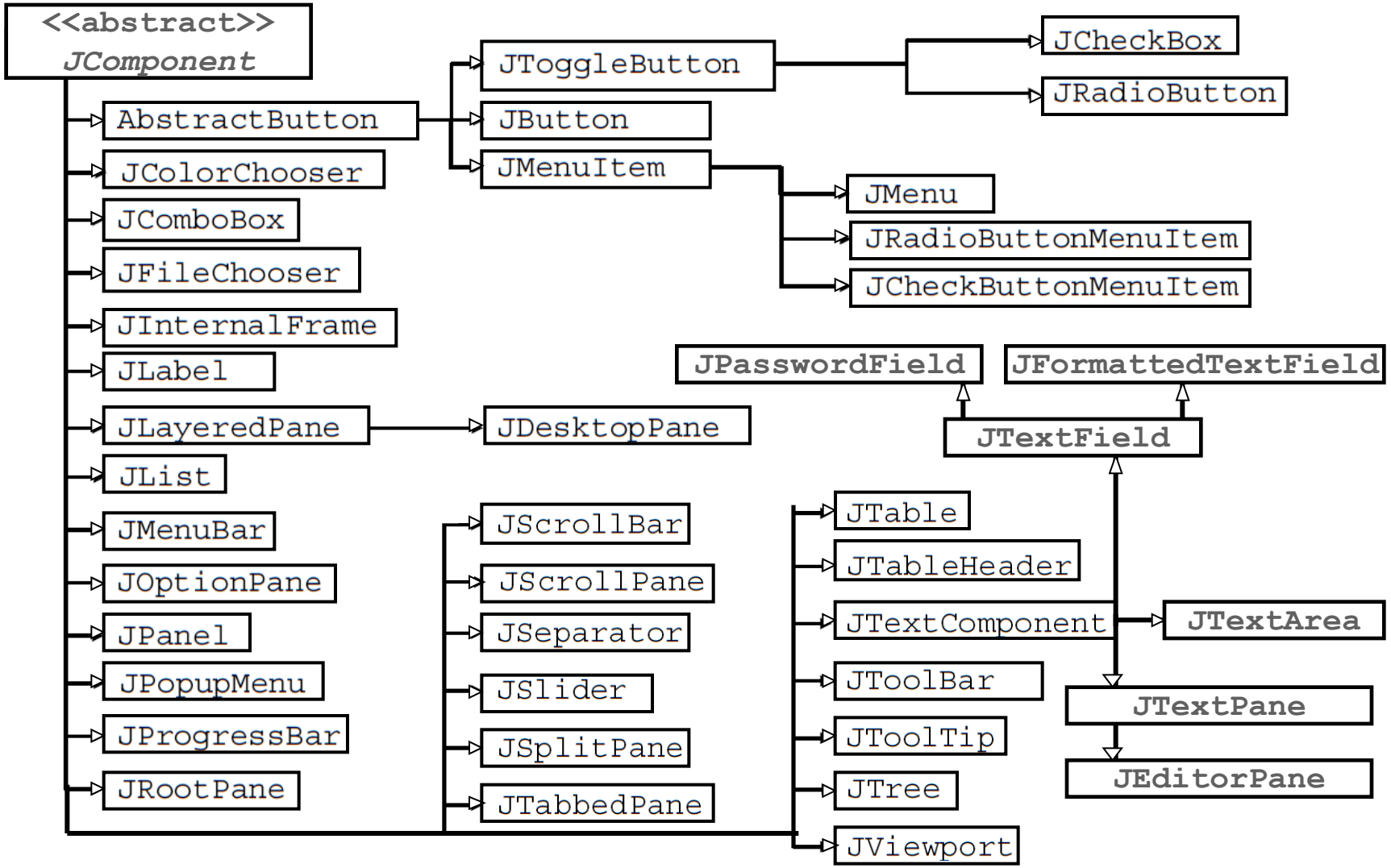
Swing

- Swing contiene componenti grafici alcuni dei quali sono simili a quelli di AWT (ad esempio le etichette), altri sono più complessi come alberi, tabelle, documenti di testo formattato
- Quasi tutti i componenti di Swing derivano da un “genitore” comune: **`JComponent`**, che estende la classe **`Container`** di AWT
- Tipicamente per ogni componente di AWT esiste un corrispondente componente di Swing con prefisso “**`J`**” (non vale il viceversa)

Gerarchia di Swing



Gerarchia di Swing



Swing

- Swing è quasi interamente indipendente dalla piattaforma (a differenza di AWT).
- Qual è il segreto? È scritto in Java!
  - Non dipende quindi da componenti legati al sistema operativo sottostante (peer components)
  - Eccezioni: **`JApplet`**, **`JDialog`**, **`JFrame`**, **`JWindow`** (sono sottoclassi dirette di classi di AWT legate alla piattaforma)

Swing

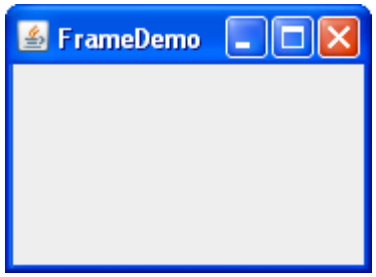
- In Swing tipicamente una GUI ha 3 livelli:
  - Un contenitore principale, che può essere un frame (**`JFrame`**), una finestra di dialogo (**`JDialog`**), un applet (**`JApplet`**).
  - Un pannello (**`JPanel`**), contenuto nel corpo del contenitore (insieme a una toolbar, **`JToolbar`**). Può contenere componenti atomici o altri pannelli.
  - I componenti “atomici” contenuti nel pannello: etichette (**`JLabel`**), bottoni (**`JButton`**), campi di testo (**`JTextField`**)...

Frame

- Per creare un frame, usiamo **`JFrame`**

```
JFrame fr = new JFrame("FrameDemo");
fr.setSize(300,100);
fr.setVisible(true);
```

un costruttore che usa una **`String`**



- Potevamo anche estendere la classe **`JFrame`**

```
import javax.swing.JFrame
public class MioFrame extends JFrame
{ public MioFrame()
   { super();
     this.setTitle("FrameDemo");
     this.setSize(300,100);
     this.setVisible(true);
   }
}
```

un costruttore senza parametri

Frame

- Varianti / raffinamenti:

```
import javax.swing.JFrame
public class MioFrame extends JFrame
{ public MioFrame()
   { super();
     this.setTitle("FrameDemo");
     this.pack();
     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     this.getContentPane().add
     (emptylabel, BorderLayout.CENTER);
     this.setVisible(true);
   }
   public static void main(String[] args)
   { new MioFrame();
   }
}
```

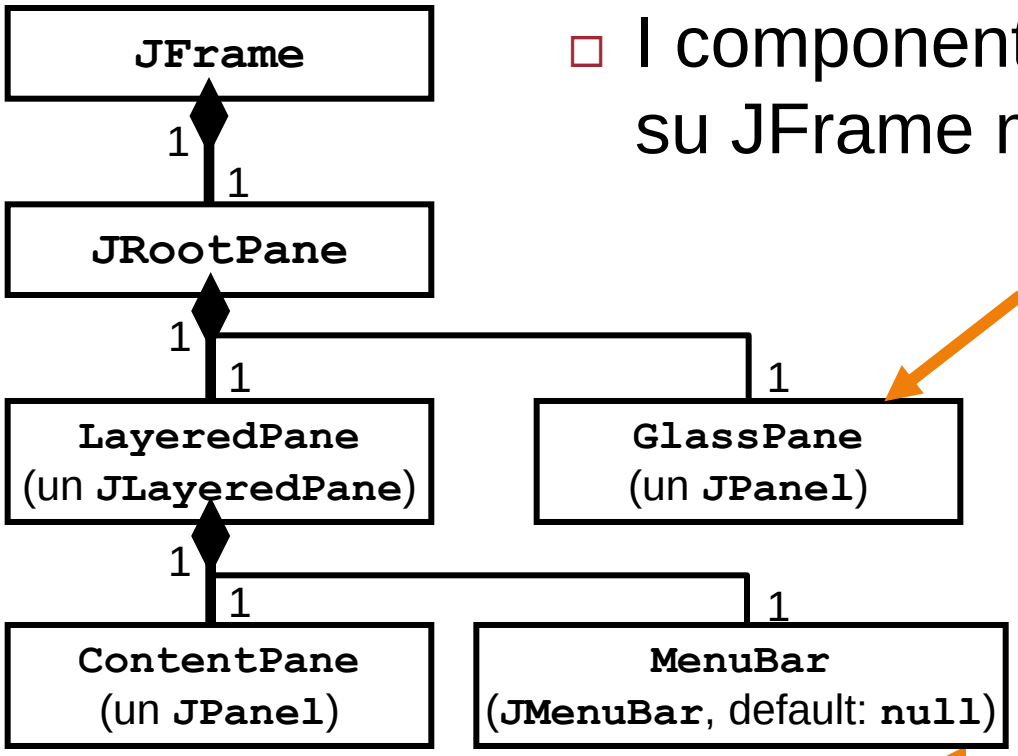
lascia dimensionare al layout manager secondo ciò che è meglio

decide il da farsi in caso che il frame venga chiuso

aggiunge oggetti al pannello (un'etichetta vuota)

il main esegue solo la creazione di un'istanza (crea una GUI in quanto estende **`JFrame`**)

Cosa contiene un **`JFrame`**



- I componenti non vanno piazzati su JFrame ma sui suoi figli.

Un "pannello di vetro" sovrapposto, invisibile di default, utile per visualizzare il cursore e intercettare eventi del mouse in qualunque punto del frame

Per avere un menù va creato

```
JMenuBar menu = new JMenuBar();
setJMenuBar(menu);
```

Qui va il contenuto vero e proprio

Entrambi sono a livello **`FRAME_CONTENT_LAYER`** (= -30000)

Altri contenitori

- La **`JRootPane`** è più che altro un elemento tecnico della gerarchia.
  - Ha un suo layout di tipo **`RootLayout`** che imposta come stanno **`glassPane`**, **`contentPane`**, **`menuBar`** (non serve a molto altro)
  - Poi implementa l'interfaccia **`RootPaneContainer`** che dichiara metodi per la gestione di componenti interni e di pannelli interni: può essere utile implementare la stessa interfaccia in altri contenitori definiti dall'utente con struttura simile

Applet

- Le applet sono programmi Java che vengono eseguiti all'interno di un browser per il web.
- Estendono la classe **`JApplet`** ridefinendo alcuni metodi che regolano il comportamento.
- Tra di essi non c'è un metodo **main**!
  - Ciò che la applet esegue, dipende dal browser.

Applet

- I metodi da riscrivere sono:
  - **`init()`**: invocato una sola volta quando la applet viene caricata (inizializza i dati)
  - **`start()`**: invocato quando si carica la applet o si fa refresh della pagina (dà il via ai thread)
  - **`stop()`**: quando si chiude il browser o si nasconde la finestra (blocca i thread per risparmiare risorse)
  - **`destroy()`**: quando si chiude il browser (libera le risorse di sistema allocate, opposto a **`init`**)
  - **`paint()`**: disegna (o ridisegna) quello che deve apparire nella finestra dell'applet, viene invocato ogni volta che lo stato dell'applet viene aggiornato

Applet

- Nel codice HTML si usa **<applet> </applet>** per inserirne una e **<param />** per i parametri

nel sorgente HTML

```
<applet code="H.class" width="300" height="100">
<param name="messaggio" value="Ciao!!" />
<param name="colore" value="Red" />
<param name="numero" value="4" />
</applet>
```

nel metodo **init()**

```
String messaggio = getParameter("messaggio");
int numero = getParameter("numero");
```

Componenti

- I componenti sono sottoclassi di **`JComponent`** ognuno coi suoi metodi per cose specifiche.
- Tutti hanno (eredità di **`JComponent`**):
  - **`setBorder(Border b)`**
  - **`setToolTipText(String istruzioni)`**
- I **bottoni push** (**`JButton`**) hanno:
  - costruttori con una **`String`** oppure un **`Icon`**
  - **`getText()`**, **`setText()`**
  - **`setEnabled()`**

Componenti

- I **bottoni check** (**`JCheckButton`**) e i **bottoni radio** (**`JRadioButton`**) danno un valore on/off.
  - I check ammettono scelte multiple, i radio una sola.
- Le **etichette** (**`JLabel`**) sono aree che visualizzano stringhe o icone:
  - costruttori con una **`String`** oppure un **`Icon`**
  - **`getText()`**, **`setText()`**
- **Aree/campi di testo** (**`JTextArea`**/**`JTextField`**) contengono testo (nei campi può editare)
  - costruttori con **`String`** o **`int`** (n. caratteri)
  - **`getText()`**, **`setText()`**
  - **`getSelectedText()`**,**`setEditable()`**

Componenti

- I componenti non vengono inseriti direttamente nel frame, ma in un pannello usato come contenitore intermedio.
- Il contenitore intermedio immediatamente dentro il frame è il **content pane**, che è reperibile invocando **`getContentPane()`**.
  - È l'unico "figlio" predefinito del frame
- I componenti di **`JComponent`** ereditano da **`Container`**, possono a loro volta contenere o essere contenuti da altri componenti.

Componenti

- Quindi per inserire dei componenti dentro un pannello e a sua volta inserire quest'ultimo dentro una cornice, scriveremo ad esempio:

```
JFrame mioFrame = new JFrame();
JButton aButton = new JButton();
JLabel aLabel = new JLabel();
JPanel pannello = new JPanel();
pannello.add(aButton);
pannello.add(aLabel);
mioFrame.getContentPane().add(pannello);
```

Esercizio

- Provare a creare la seguente GUI

- Estendere la classe **`JFrame`** e mettere tutto nel costruttore. Conviene usare un metodo apposito per inizializzare bottone ed etichetta, anche questo da mettere nel costruttore.

Esercizio

```
import javax.swing.*;

public class App extends JFrame
{ private JButton bottone;
  private JLabel etichetta;

  public App()
  { super();
    this.setDefaultCloseOperation
              (JFrame.EXIT_ON_CLOSE);
    this.setTitle("Cornice");
    this.inizializzaGUI();
    this.pack();
    this.setVisible(true);
  }
```

Esercizio

```
  private void inizializzaGUI()
  { JPanel intermedio = new JPanel();
    bottone = new JButton("Bottone 1");
    etichetta = new JLabel("Etichetta 2");
    intermedio.add(bottone);
    intermedio.add(etichetta);
    this.getContentPane().add(intermedio);
  }

  public static void main(String[] args)
  { new App();
  }
}
```

Esercizio

- Creare la seguente GUI

- Come si fa ad allineare i componenti?
  - Quando si fa **`add()`** si specifica solo di aggiungere il componente, ma non dove!

Gestori di layout

- I gestori di layout sono oggetti che dicono come disporre i contenitori nel pannello.
- Essendo oggetti, vanno creati a partire dalla loro classe e inseriti nel contenitore. Ad es.:

```
JPanel intermedio = new JPanel();
intermedio.setLayout(new GridLayout(3,4));
```

  definisce (e impone di usare) un layout manager che usa una griglia.
- Attenzione: si trovano in **`java.awt`** !

Gestori di layout

- Se non specificato, il default è **`FlowLayout`** (mette componenti da sinistra a destra finché può, poi passa alla riga sotto). Altri possibili:
  - **`GridLayout`**, mette in una griglia n×m
  - **`GridBagLayout`**, come la griglia, ma consente ai componenti di occupare più righe / colonne
  - **`BoxLayout`**, mette tutto in orizzontale o verticale
  - **`BorderLayout`**, usa 5 zone (nord, sud, ovest, est, centro) e va detto dove va ogni componente
  - .... (si possono anche innestare)

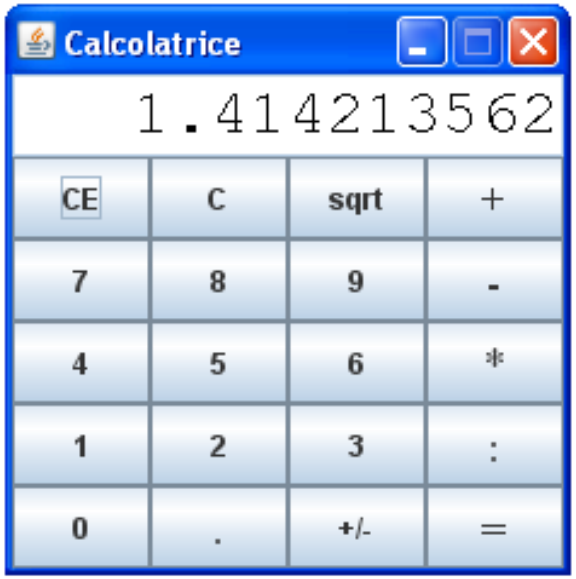
Gestori di layout

- Esempio: realizzare questa interfaccia


```
private void inizializzaGUI()
{ JPanel intermedio = new JPanel();
  intermedio.setLayout(new GridLayout(2,3));
  JButton[] b = new JButton[6];
  b[0]= new JButton("Usa"); b[1]= new JButton("Salta");
  b[2]= new JButton("Spara"); b[3]= new JButton("Sinistra");
  b[4]= new JButton("Abbassati"); b[5]= new JButton("Destra");
  for (int i = 0; i<bottone.length; i++)
    { intermedio.add(bottone[i]); }
  this.getContentPane().add(intermedio);
}
```

Gestori di layout

- Esercizio: realizzare una interfaccia per calcolatrice (classico esercizio Java)
- Si possono abbellire i vari elementi usando diversi font e colori (java.awt definisce le classi **`Font`** e **`Color`**)



notazione RGB o diversi colori predefiniti

Look and Feel

- Swing consente di modificare il “look and feel” (L&F), cioè come appaiono i componenti (look) e cosa fanno (feel).
- L’architettura di Swing lo consente perché ogni componente è descritto da due classi: il **`JComponent`** e una **`ComponentUI`**.
  - Ad es., oltre al **`JButton`** (estende **`JComponent`**) c’è un **`ButtonUI`** (estende **`ComponentUI`**).
  - La sottoclasse **`ComponentUI`** si chiama “delegato UI”, o “delegato L&F” o anche solo “UI”.

Look and Feel

- In aggiunta, la classe **`UIManager`** (sottoclasse diretta di **`Object`** del package **`javax.swing`**) si occupa di gestire il settaggio del L&F.
  - Possiede metodi statici get e set per farlo.
- Esempio di uso:

ricava (e setta) il L&F “di sistema”

```
try
{ UIManager.setLookAndFeel(
  UIManager.getSystemLookAndFeelClassName() ); }
catch (ClassNotFoundException e) { ... }
catch (InstantiationException e) { ... }
catch (IllegalAccessException e) { ... }
catch (UnsupportedLookAndFeelException e) { ... }
```

Look and Feel

- **`setLookAndFeel`** deve annunciare con **`throws`** tutte le seguenti eccezioni controllate:
  - **`ClassNotFoundException`** : non c'è il L&F
  - **`InstantiationException`** : il L&F chiede di creare un'istanza di classi astratte (o simile)
  - **`IllegalAccessException`** : il L&F chiede di usare qualcosa di non accessibile (privato)
  - **`UnsupportedLookAndFeelException`** : il valore di **`isSupportedLookAndFeel()`** del L&F è false

Look and Feel

- Variamo il look and feel usandone uno chiamato “Motif”

```
import javax.swing.JFrame
public class MioFrame extends JFrame
{ public MioFrame() { ... }

public static void main(String[] args)
{ String lnF = "com.sun.java.swing.plaf.motif.MotifLookAndFeel";
  try
  { UIManager.setLookAndFeel(lnF); }
  catch (ClassNotFoundException e) { System.err.println(lnF+
        " non trovato! Sicuro che esiste? Userò il default..");}
  catch (UnsupportedLookAndFeelException e) { System.err.
         println(lnF+" non funziona! Userò il default..");}
  catch (Exception e) {System.err.println("Altri problemi con:"
         +lnF+", userò il default.."); e.printStackTrace();}
  new MioFrame();
 }
}
```

provare anche con altri

meglio settare subito il L&F
o la JVM metterà un default

queste eccezioni non sono gestite granché bene
ma non è un problema: è sensato usare il default