APPLICAZIONI GRAFICHE in JAVA

La classe AWT e SWING

La classe AWT

L'interfaccia utente è lo strumento attraverso il quale un'applicazione comunica con l'utente

Per sviluppare un programma contenent una interfaccia utente del tipo grafico abbiamo bisogno di un ambiente di alto livello chiamato TOOLKIT che ha il compito di semplificare l'utilizzo di **OGGETTI** ed **EVENTI**:

IL TOOLKIT e' un API (Application Programmi Interface) che mette a disposizione oggetti di tipo finestra (icone, pulsanti, menu) chiamati WIDGET.
Un ***widget*** e' caratterizzato dai seguenti elementi:

1. La ***finestra*** su cui e' posizionato e che lo contiene;
2. Lo ***stato***;
3. Un insieme di ***metodi*** che possono modificare la finestra e' lo stato;
4. Un insieme di ***eventi*** ad esso associati che possono essere intercettati direttamente;
5. Un ***sistema di collegamento*** tra eventi e widgetm che permette al programmatore la gestione personale delle risposte all'evento;

I widget sono organizzati in classi: *creare un widget significa istanziare una classe in un oggetto, mentre invece per personalizzare il comportamento dobbiamo riscrivere i metodi*

I widget si possono classificare in due categorie fondamentali:

- Widget **CONTENITORI** che servono per ragruppare altri widget all'interno di un contesto grafico;
- Widget **COMPONENTI** che servono per presentare o ricevere informazioni dall'utente;

In JAVA, finestre ed elementi di un'interfaccia grafica, vengono solitamente chiamati ***COMPONENTI***.

JAVA mette a disposizione due librerie di classi necessarie per la gestione delle interfaccie grafiche:

- Java AWT (**Java Abstract Widget Toolkit**);
- Java SWING;

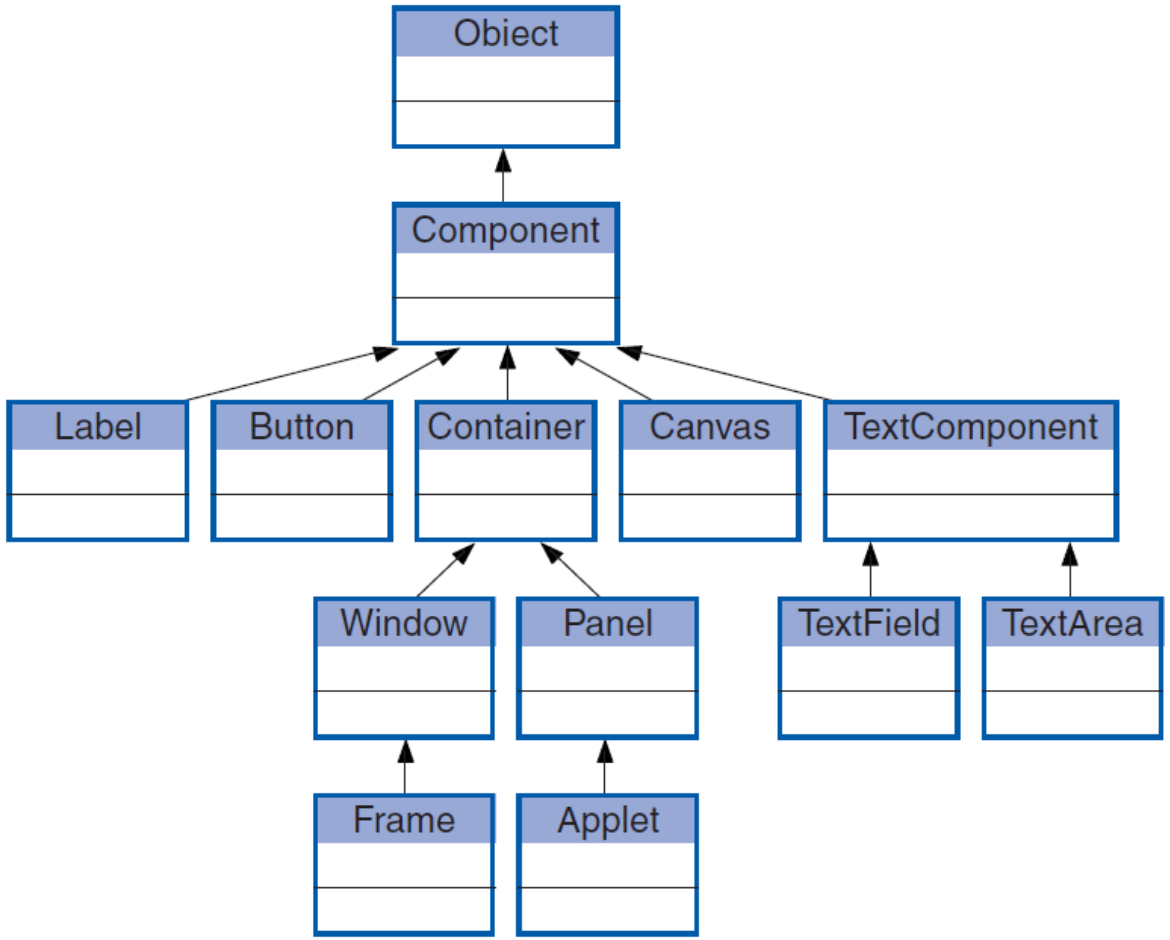
Un progetto di un interfaccia grafica viene fatta seguendo i seguenti passi principali:

- Importazione dei package necessari;
- Definizione di una **finestra** top level definizione del **contenuto**, cioe' dei componenti di interfaccia;
- Gestione degli **eventi** nei componenti;

Dentro a ogni JFrame c'è un ***container*** (in cui posso aggiungere pannelli, oggetti **JPanel**);

Un **contenitore** è un oggetto che può contenere le *componenti*. Usando opportuni metodi, si possono *aggiungere o togliere le componenti dal contenitore*. *Il compito del contenitore è quello di posizionare e dimensionare le componenti all'interno del contenitore stesso* (layout delle componenti).
Esistono vari modi con cui questa operazione viene eseguita e dipende dal tipo di gestore del layout (in inglese **Layout Manager**) che viene assegnato a quel particolare contenitore.

Le classi di Java che realizzano le componenti e i contenitori, sono organizzate in una **gerarchia delle componenti** che ha come padre la classe Component



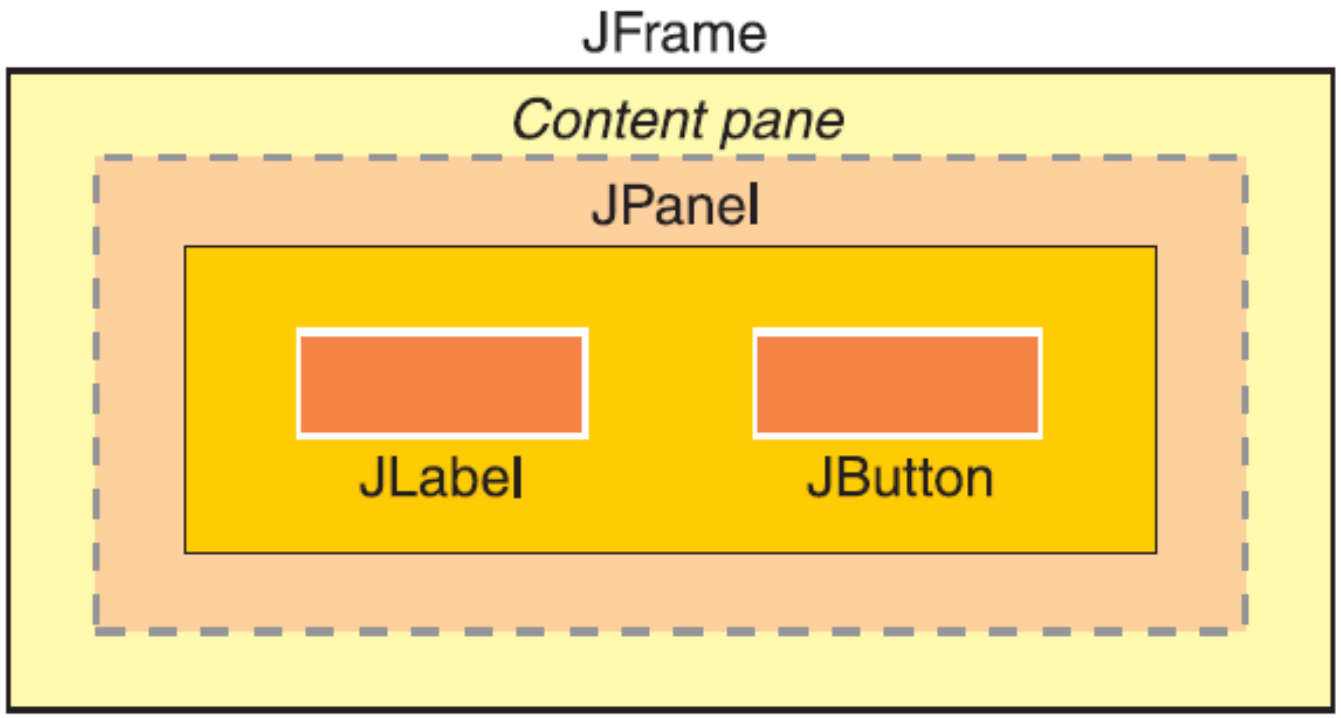
Si noti che la classe **Container** è una sottoclasse astratta derivata dalla classe Component e rappresenta i contenitori. Le sue sottoclassi implementano concretamente i contenitori che possono essere utilizzati per costruire un'interfaccia grafica.

Nella libreria Swing i nomi delle componenti e dei contenitori iniziano con la lettera J e si possono classificare in:

- **contenitori principali (top-level container):** finestra principale (JFrame), finestra di dialogo(JDialog) e finestra di applet (JApplet);
- **contenitori intermedi** (intermediate container, indicati anche con il termine pane), cioè contenitori di altre componenti: pannello (JPanel), pannello con barre di scorrimento (JScrollPane) o scheda con etichette (JTabbedPane);
- **componenti atomiche:** etichetta (JLabel), pulsante (JButton), casella di testo (JTextField), area di testo (JTextArea), combo box (JComboBox), tabelle (JTable).

I pannelli (pane) hanno lo scopo di organizzare le componenti atomiche nella finestra, semplificandone il posizionamento.

Lo schema seguente rappresenta ***l'organizzazione delle componenti*** **Swing** per una generica finestra ***contenente un pannello,*** al cui interno sono collocati **un'etichetta** e un **pulsante**



La costruzione delle interfacce utente, tramite le componenti e i contenitori **AWT** e **Swing**, può essere fatta in modo visuale, usando l'ambiente di programmazione *NetBeans*, oppure in modo testualescrivendo direttamente il codice Java

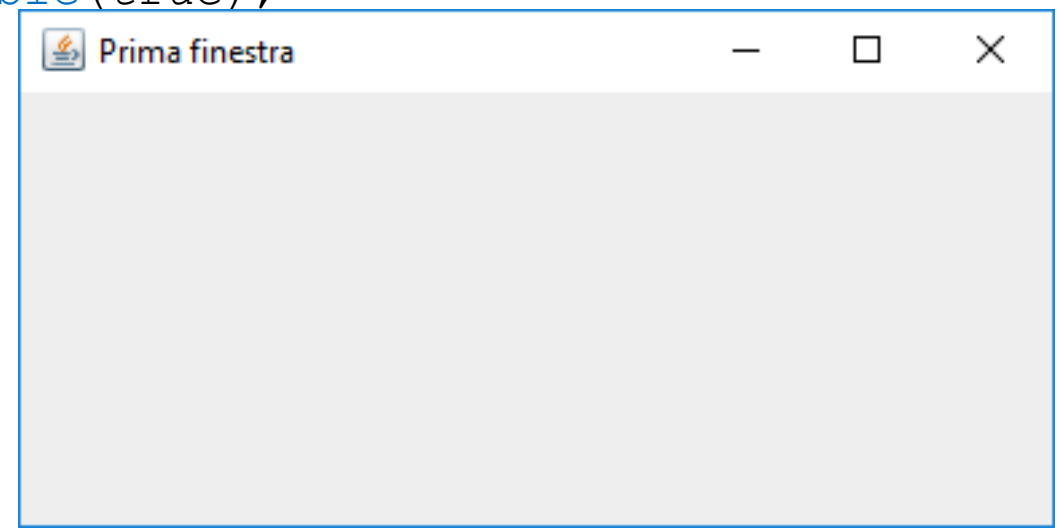
Creazione di una finestra con classe JWindow

```
import javax.swing.*;

public class EsempioSwing_n_1{
       public static void main(String[] args) {

       //istanza dell'oggetto di classe Jwindow
       JWindow finestra_1 = new Jwindow();

       //definizione delle dimensioni della finestra 1
       finestra_1.setSize(400,200);

       //la finestra diventa visibile chiamando il metodo setVisible
       finestra_1.setVisible(true);
   } //main
} //class
```

Per chiudere la finestra nell'esempio precedente avrei bisogno di alcuni pulsanti, l'esempio che segue crea una finestra piu' completa, che sara' un contenitore principale utilizzando la classe JFrame (*JFrame di Swing*)

Creazione finestra con classe JFrame

```
import javax.swing.*;

public class EsempioSwing_n_2{
	public static void main(String[] args) {

		//istanza dell'oggetto di classe Jwindow
		Jframe finestra_1 = new JFrame("Prima Finestra");

		//definizione delle dimensioni della finestra 1
		finestra_1.setSize(400,200);

		//la finestra diventa visibile chiamando il metodo setVisible
		finestra_1.setVisible(true);
	} //main
} //class
```



I comandi standard delle finestre sono già attivi

ATTENZIONE: la chiusura non distrugge il Frame ma lo nasconde soltanto. Per chiuderlo effettivamente ci vuole **Ctrl+C**

Creazione finestra con classe JFrame (esempio Migliorato)

La finestra dell'esempio precedente viene visualizzata nell'angolo superiore sinistro dello schermo Ricordando che tutto lo schermo è 800X600 o 1024x768 etc..., per impostare la posizione di un qualunque contenitore si usa setLocation() che riceve come parametri le coordinate dello schermo.

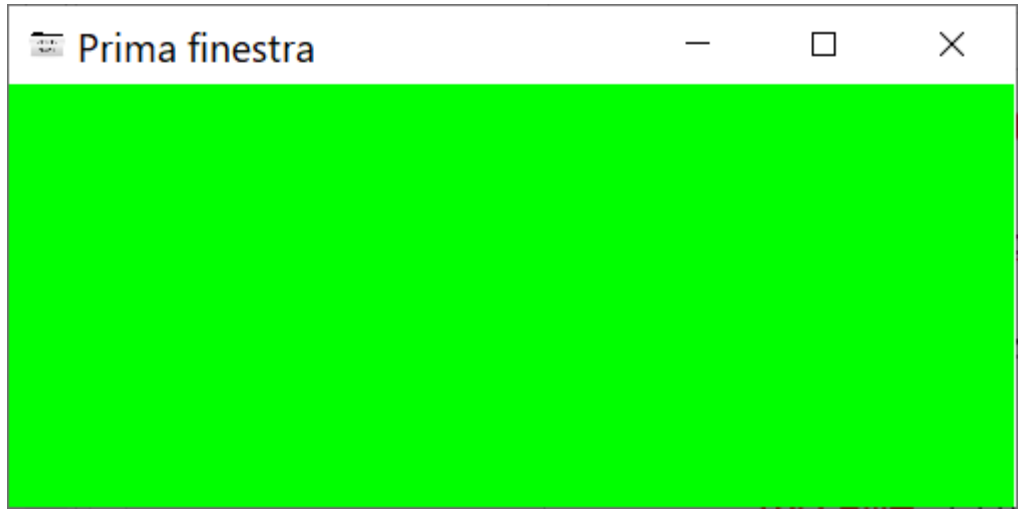
```
import javax.swing.*;

public class EsempioSwing_n_3{
   public static void main(String[] args) {

        JFrame finestra_1 = new JFrame("Prima finestra"); //istanza l'oggetto JFrame
        finestra_1.setSize(400,200); //set delle dimensioni della finestra_1

        //finestra_1.setLocation(200, 200); //(0,0) = angolo superiore sinistro
        //Posizione e dimensioni si possono anche fissare insieme, col metodo setBounds()
        finestra_1.setBounds(200,200, 400, 200);
        finestra_1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Esce dall'app
        //la finestra diventa visibile chiamando il metodo setVisible
        finestra_1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Esce dall'APP
        finestra_1.setVisible(true);
    } //main
} //class
```

INSERIRE UN'IMMAGINE COME ICONA DELL'APP

```
import javax.swing.*;

public class EsempioSwing_n_3{
   public static void main(String[] args) {

      JFrame finestra_1 = new JFrame("Prima finestra"); //istanza l'oggetto JFrame

       . . .

      //INSERIRE UN'IMMAGINE COME ICONA DELL'APP contenuta nella root del progetto
      ImageIcon miaIcona = new ImageIcon("nomeImmagine.png");
      finestra_1.setIconImage(miaIcona.getImage());


      finestra_1.setVisible(true);
   } //main
} //class
```

Cambiare lo sfondo al JFrame

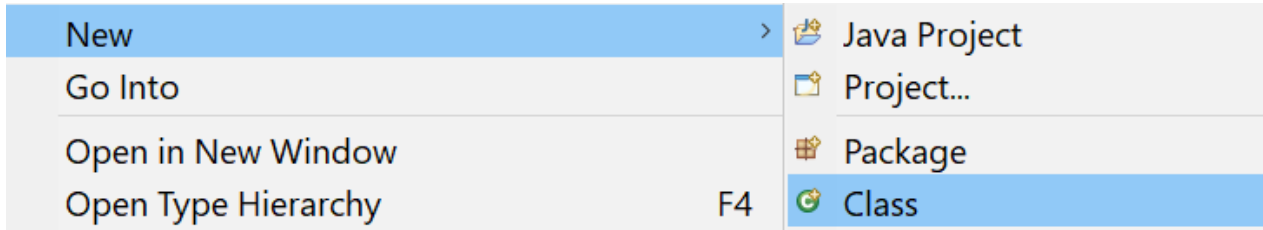
```
import javax.swing.*;

public class EsempioSwing_n_3{
   public static void main(String[] args) {

      JFrame finestra_1 = new JFrame("Prima finestra"); //istanza l'oggetto JFrame


       . . .

      //ottenuto il Panel del JFRAME cambiamo il colore allo sfondo (pannello)
      finestra_1.getContentPane().setBackground(new Color(0,255,0));
      //finestra_1.getContentPane().setBackground(Color.green));


      finestra_1.setVisible(true);
   } //main
} //class
```

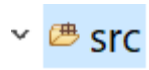


Personalizzare il JFrame (1/2)

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

- Cliccare su src col tasto destr;
- File/new Class:
- dare un nome alla classe ***MioJFrame***
- ***Estendere la classe JFrame***

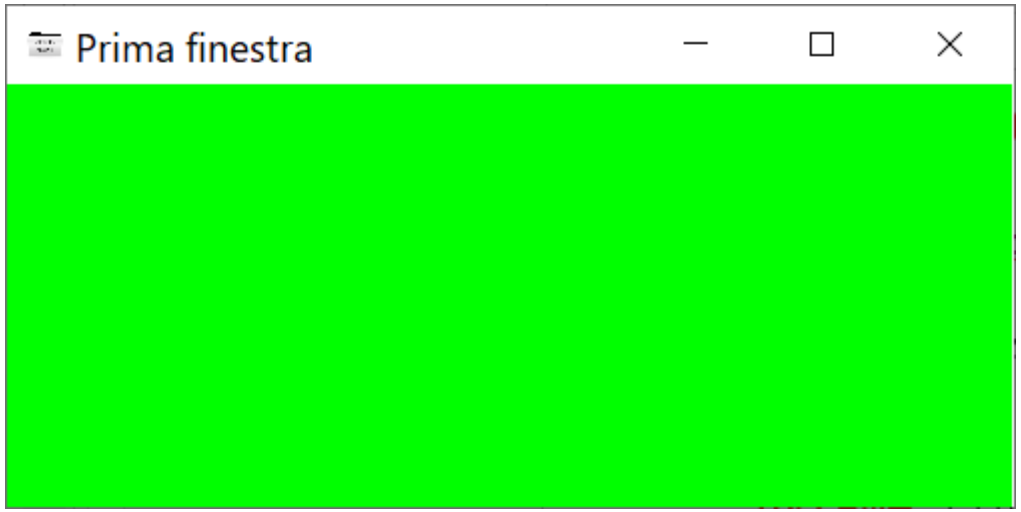





```java
import java.awt.Color;
import javax.swing.*;

public class MioJFrame extends JFrame {

    MioJFrame(){
        this.setSize(400,200); //definizione delle dimensioni della finestra
        this.setLocation(200, 200); //(0,0) = coordinate dall'angolo superiore sinistro
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Esce dall'applicazione
        ImageIcon miaIcona = new ImageIcon("nomeImmagine.png");
        this.setIconImage(miaIcona.getImage());
        this.getContentPane().setBackground(Color.green);
        this.setVisible(true); //la finestra diventa visibile chiamando il metodo setVisible
    } //costr

} //class
```

Personalizzare il JFrame (2/2)

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

public class Main{
    public static void main(String[] args) {

        MioJFrame finestra = new MioJFrame (); //istanza l'oggetto MioJFrame
        finestra.setVisible(true);

    } //main
} //class
```



Come utilizzare il JFrame di Swing: preparazione componenti dell'interfaccia

In Swing non si possono aggiungere nuovi **componenti** direttamente al **JFrame**;

Dentro a ogni JFrame c'è un ***container*** (in cui posso aggiungere pannelli, oggetti **JPanel**);

Il container del JFrame è recuperabile mediante il metodo ***getContentPane();***

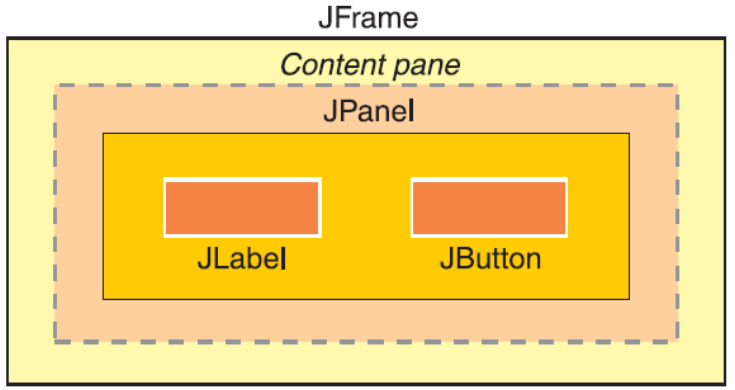
Recuperato il container ad esso tipicamente si aggiunge un pannello (**JPanel**) col metodo ***add();***

sul pannello si può:

1. **disegnare (forme, immagini...)**
2. **...o aggiungere pulsanti, etichette, icone, (cioè aggiungere altri componenti!)**

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

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

public class Main {
	public static void main(String[] v){

		MioFrame f = new MioFrame("Esempio 3");

		//ottengo il container con getContentPane()
		Container c = f.getContentPane();

		//creo un Pannello utile per aggiungere componenti
		JPanel pannello = new JPanel();

		//aggiungo il pannello creato al container con il metodo add()
		c.add(pannello);

		//rendo  visibile il frame cosi costruito
		f.setVisible(true);
	} //main
} //class
```



```
public class MioPannello extends JPanel {

        // nessun costruttore, va bene il default
        public void paintComponent(Graphics g){
                super.paintComponent(g);
                ...
                // qui aggiungeremo le nostre istruzioni di
                // disegno…
                // g è un oggetto gestito dal sistema a cui ci si
                // rivolge per disegnare
        } //func

} //class
```

Codice compatto: ma non abbiamo disegnato niente, né aggiunto componenti, sul pannello! Però, avendo, il pannello, potremmo usarlo per disegnare e inserire altri componenti!