**Java Swing Tutorial**

Java Swing è una libreria per creare interfacce grafiche (GUI) in Java. Offre una varietà di componenti come pulsanti, etichette, campi di testo e finestre. Swing è una parte del pacchetto Java Foundation Classes (JFC) e fornisce strumenti per creare interfacce grafiche (GUI). Swing è costruito sopra AWT (Abstract Window Toolkit), ma è molto più flessibile e potente

**Prima Finestra con JFrame**

**La classe JFrame, che fa parte del package javax.swing,** è una classe che consente di creare e gestire una finestra di primo livello in un'applicazione Java. Permette la creazione di finestre dove poi è possibile inserire etichette (JLabel), bottoni (JButton) e campi di testo (JTextField e JTextArea) per scrivere applicazioni Java basate su interfaccia grafica. Fornisce inoltre un modo indipendente dalla piattaforma per creare interfacce utente

import javax.swing.JFrame;

public class PrimaFinestra {

public static void main(String[] args) {

// Creiamo un oggetto JFrame

JFrame finestra = new JFrame("La mia prima finestra");

// Definiamo la dimensione della finestra (larghezza, altezza)

finestra.setSize(400, 300);

// Specifica cosa fare quando si chiude la finestra (terminare il programma)

finestra.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Rendiamo la finestra visibile

finestra.setVisible(true);

}

}

Le etichette (JLabel) sono usate per mostrare del testo o delle immagini. Sono statiche, nel senso che non permettono all'utente di modificarle direttamente

**Esempio di Codice – Finestra con Etichetta**

import javax.swing.\*;

public class FinestraConLabel {

public static void main(String[] args) {

// Creiamo la finestra

JFrame finestra = new JFrame("Finestra con JLabel");

finestra.setSize(400, 200);

finestra.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Creiamo una JLabel

JLabel etichetta = new JLabel("Questa è una semplice etichetta.");

// Aggiungiamo la JLabel alla finestra

finestra.add(etichetta);

// Rendiamo la finestra visibile

finestra.setVisible(true);

}

}

I campi di testo (JTextField) permettono all'utente di inserire brevi stringhe di testo. Sono molto utili quando si vogliono raccogliere informazioni come nomi, indirizzi email o password

**Esempio di Codice – Finestra con Campo di Testo**

import javax.swing.\*;

public class FinestraConTextField {

public static void main(String[] args) {

// Creiamo la finestra

JFrame finestra = new JFrame("Finestra con JTextField");

finestra.setSize(400, 200);

finestra.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Creiamo un JTextField

JTextField campoTesto = new JTextField(20); // Il numero indica la lunghezza

// Aggiungiamo il JTextField alla finestra

finestra.add(campoTesto);

// Rendiamo la finestra visibile

finestra.setVisible(true);

}

}

La classe JTextArea, invece, permette all'utente di inserire più righe di testo. È molto utile per i commenti, descrizioni o note

**Esempio di Codice - Finestra con Area di Testo**

**import javax.swing.\*;**

**public class FinestraConTextArea {**

**public static void main(String[] args) {**

**// Creiamo la finestra**

**JFrame finestra = new JFrame("Finestra con JTextArea");**

**finestra.setSize(400, 300);**

**finestra.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**// Creiamo una JTextArea**

**JTextArea areaTesto = new JTextArea(5, 30); // 5 righe e 30 colonne**

**// Creiamo uno JScrollPane per permettere lo scorrimento**

**JScrollPane scrollPane = new JScrollPane(areaTesto);**

**// Aggiungiamo la JTextArea alla finestra**

**finestra.add(scrollPane); // Aggiungiamo l'area testo dentro uno scroll pane**

**// Rendiamo la finestra visibile**

**finestra.setVisible(true);**

**}**

**}**

**La classe JButton** è un componente di Java Swing utilizzato per creare pulsanti cliccabili nelle interfacce grafiche. Può contenere testo, icone o entrambi, e permette di gestire eventi tramite **ActionListener**. Quando l'utente clicca su un JButton, l'evento viene catturato e si possono eseguire azioni specifiche, come aggiornare l'interfaccia o eseguire qualsiasi altra operazione

**Esempio di Codice - Finestra con un Pulsante**

import javax.swing.JButton;

import javax.swing.JFrame;

public class FinestraConPulsante {

public static void main(String[] args) {

// Creiamo la finestra

JFrame finestra = new JFrame("Finestra con Pulsante");

finestra.setSize(400, 300);

finestra.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Creiamo un pulsante

JButton bottone = new JButton("Cliccami!");

// Aggiungiamo il pulsante alla finestra

finestra.add(bottone);

// Rendiamo la finestra visibile

finestra.setVisible(true);

}

}

**Esempio di Codice per la gestione degli Eventi**

Una volta creato un pulsante si deve gestire ciò che accade quando l'utente interagisce con esso: ciò avviene mediante l’utilizzo dei **listener**. Ogni volta che un pulsante viene premuto, si genera un evento che deve essere catturato da un listener. Per esempio, per sapere quando un pulsante viene cliccato, utilizziamo un **ActionListener**.

import javax.swing.JButton;

import javax.swing.JFrame;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class FinestraConEvento {

public static void main(String[] args) {

JFrame finestra = new JFrame("Finestra con Evento");

finestra.setSize(400, 300);

finestra.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JButton bottone = new JButton("Cliccami!");

// Aggiungiamo un listener per catturare l'evento di click

bottone.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

System.out.println("Hai cliccato il pulsante!");

}

});

finestra.add(bottone);

finestra.setVisible(true);

}

}

**Gestione dei Layout Manager**

I **layout manager** all’interno delle finestre controllano come i componenti sono organizzati all'interno delle stesse. Esistono diversi tipi di layout manager, come ad esempio:

* **FlowLayout**: Posiziona i componenti in una riga
* **BorderLayout**: Divide la finestra in 5 aree: Nord, Sud, Est, Ovest e Centro
* **GridLayout**: Organizza i componenti in una griglia
* **BoxLayout**: organizza i componenti in una finestra in maniera lineare, disposti verticalmente o orizzontalmente. È utile quando si desidera allineare gli elementi uno sopra l'altro (asse Y) o uno accanto all'altro (asse X) rispettando l'ordine di aggiunta dei componenti

**Esempio di Codice - Utilizzo dei Layout Manager**

import javax.swing.\*;

import java.awt.\*;

public class FinestraConGridLayout {

public static void main(String[] args) {

JFrame finestra = new JFrame("Finestra con Layout");

finestra.setSize(400, 300);

finestra.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Impostiamo il layout a GridLayout (2 righe, 2 colonne)

finestra.setLayout(new GridLayout(2, 2));

// Aggiungiamo 4 pulsanti

finestra.add(new JButton("Pulsante 1"));

finestra.add(new JButton("Pulsante 2"));

finestra.add(new JButton("Pulsante 3"));

finestra.add(new JButton("Pulsante 4"));

finestra.setVisible(true);

}

}