



UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Università degli Studi di Salerno

Dipartimento di Informatica

Programmazione ad Oggetti

a.a. 2023-2024

Layouts

Docente: Prof. Massimo Ficco

E-mail: mficco@unisa.it

Layouts

Un layout manager è una modalità di posizionamento dei componenti in un pannello o in un altro contenitore .

In Java, ogni pannello ed qualunque componente GUI può essere un **Container**.

Un particolare layout viene impostato chiamando il metodo **setLayout** del container.



Layouts

.

AWT/Swing supportano diversi managers. Ne considereremo 4:

- FlowLayout,
- GridLayout,
- BorderLayout,
- BoxLayout.

Tutte queste classi implementano l'interfaccia
java.awt.LayoutManager.



FlowLayout

Posiziona i componenti in linea fino a quando rientrano nelle dimensioni orizzontali della finestra, quindi comincia una nuova linea.

Cerca di ottimizzare lo spazio nel disporre i componenti. Per default utilizza un allineamento centrato.

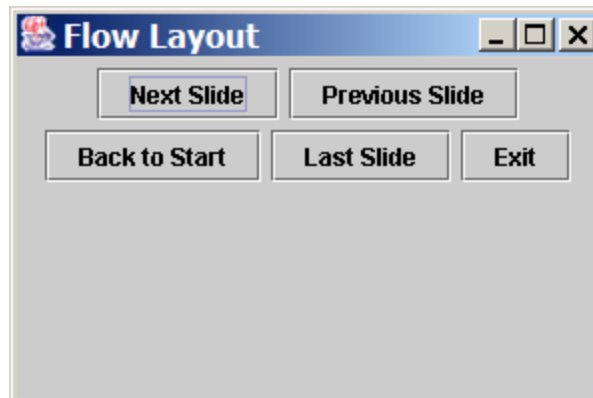
Lascia che ogni componente conservi la dimensione naturale (preferred).

Spesso è utilizzato per inserire pulsanti in un pannello.



FlowLayout

```
Container c = getContentPane();  
c.setLayout (new FlowLayout() );  
c.add (new JButton ("Next Slide"));  
c.add (new JButton ("Previous Slide"));  
c.add (new JButton ("Back to Start"));  
c.add (new JButton ("Exit"));
```



GridLayout

Divide il pannello in una griglia in un dato numero di righe e colonne.

Inserisce i componenti nelle celle della griglia.

Forza le dimensioni del componente ad occupare l'intera cella.

Permette di inserire spazi aggiuntivi tra le celle.



GridLayout (cont'd)

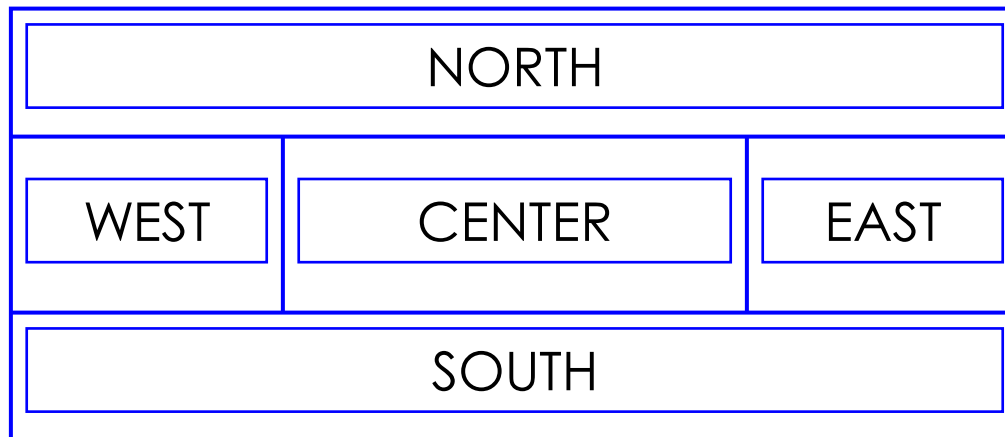
```
Container c = getContentPane();  
c.setLayout (new GridLayout(3, 2, 10, 20 ));  
c.add (new JButton ("Next Slide"));  
c.add (new JButton ("Previous Slide"));  
c.add (new JButton ("Back to Start"));  
c.add (new JButton ("Last Slide"));  
c.add (new JButton ("Exit"));
```

Extra
space
between
the cells



BorderLayout

Divide l'area in 5 regioni ed aggiunge un componente ad ogni regione.

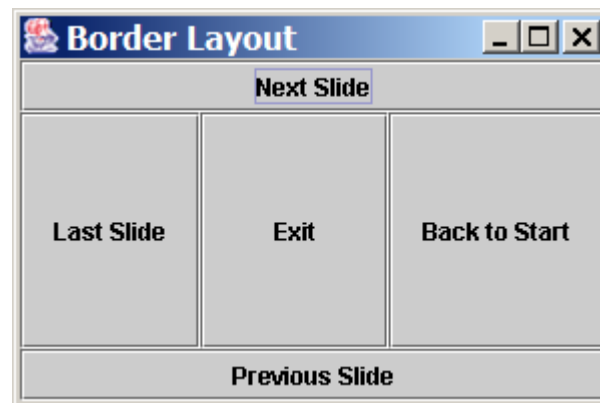


Forza le dimensioni del componente ad occupare l'intera regione.



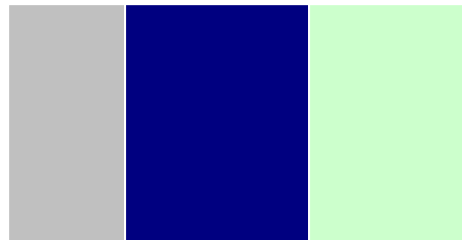
BorderLayout (cont'd)

```
Container c = getContentPane();  
c.setLayout(new BorderLayout() );  
c.add (new JButton ("Next Slide"), BorderLayout.NORTH);  
c.add (new JButton ("Previous Slide"), BorderLayout.SOUTH);  
c.add (new JButton ("Back to Start"), BorderLayout.EAST);  
c.add (new JButton ("Last Slide"), BorderLayout.WEST);  
c.add (new JButton ("Exit"), BorderLayout.CENTER);
```

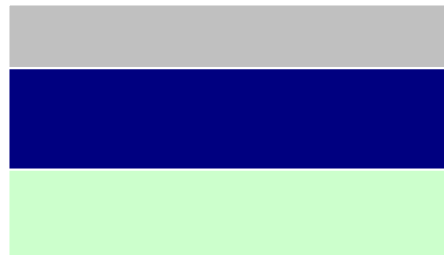


BoxLayout

In una box orizzontale i componenti sono posizionati da sinistra a destra.



In una box verticale i componenti sono posizionati dall'alto verso il basso.



“Horizontal” o
“vertical” non
hanno niente a
che fare con la
forma della box.



BoxLayout (cont'd)

BoxLayout è il layout di default per un container di classe **Box**.

Il linguaggio per utilizzare le Box è un po' differente:

```
Box box1 = Box.createHorizontalBox();
box1.add (...);

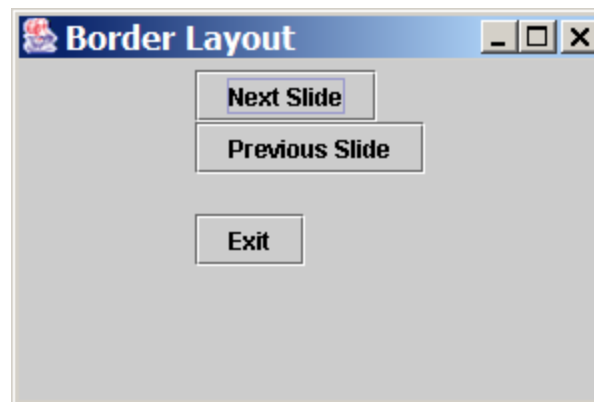
// add a spacer, 60 pixels:
box1.add(Box.createHorizontalStrut (60));

Box box2 = Box.createVerticalBox();
...
```



BoxLayout

```
Container c = getContentPane();  
c.setLayout(new FlowLayout() );  
Box box = Box.createVerticalBox();  
box.add (new JButton ("Next Slide"));  
box.add (new JButton ("Previous Slide"));  
box.add (Box.createVerticalStrut (20) );  
box.add (new JButton ("Exit"));  
c.add (box);
```



Default Layouts*

Ogni componente ha un layout di default che rimane tale fino a che non viene utilizzato il metodo `setLayout`.

Alcuni layout di default sono:

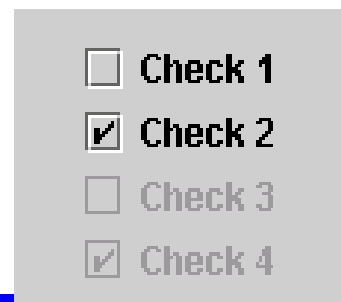
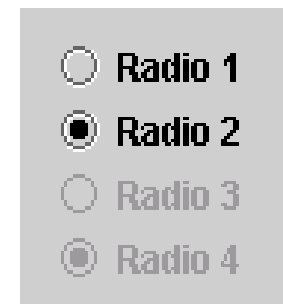
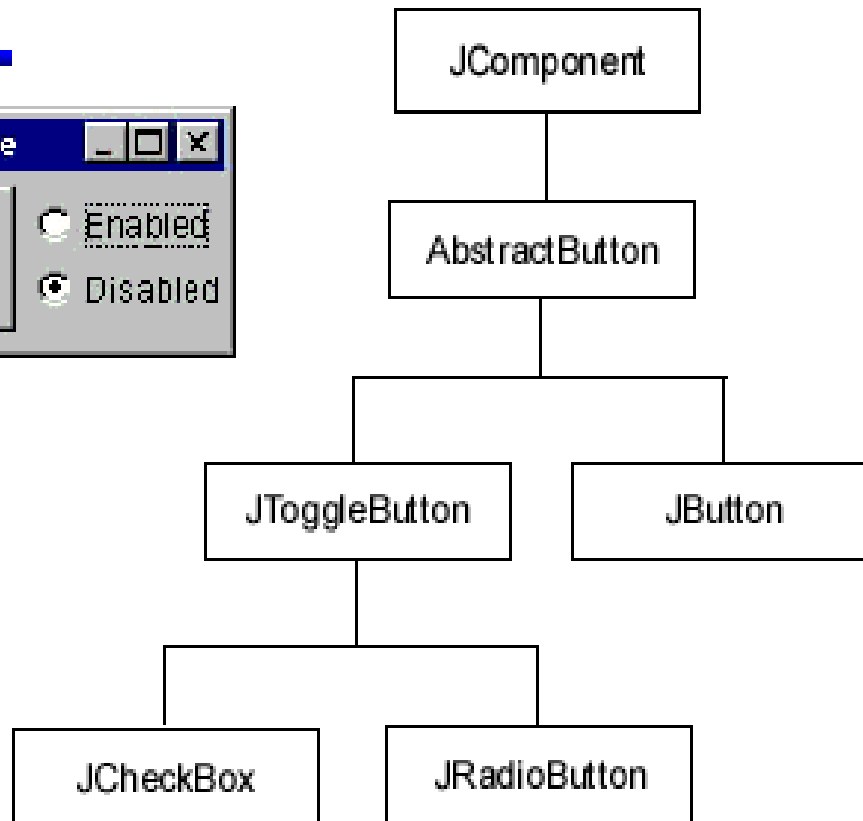
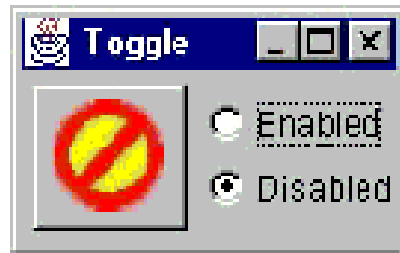
Content pane ← BorderLayout

JPanel ← FlowLayout

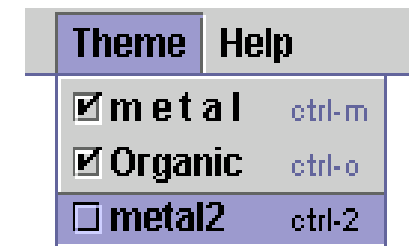
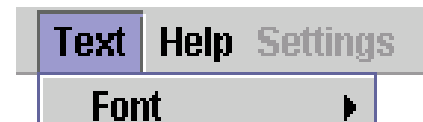
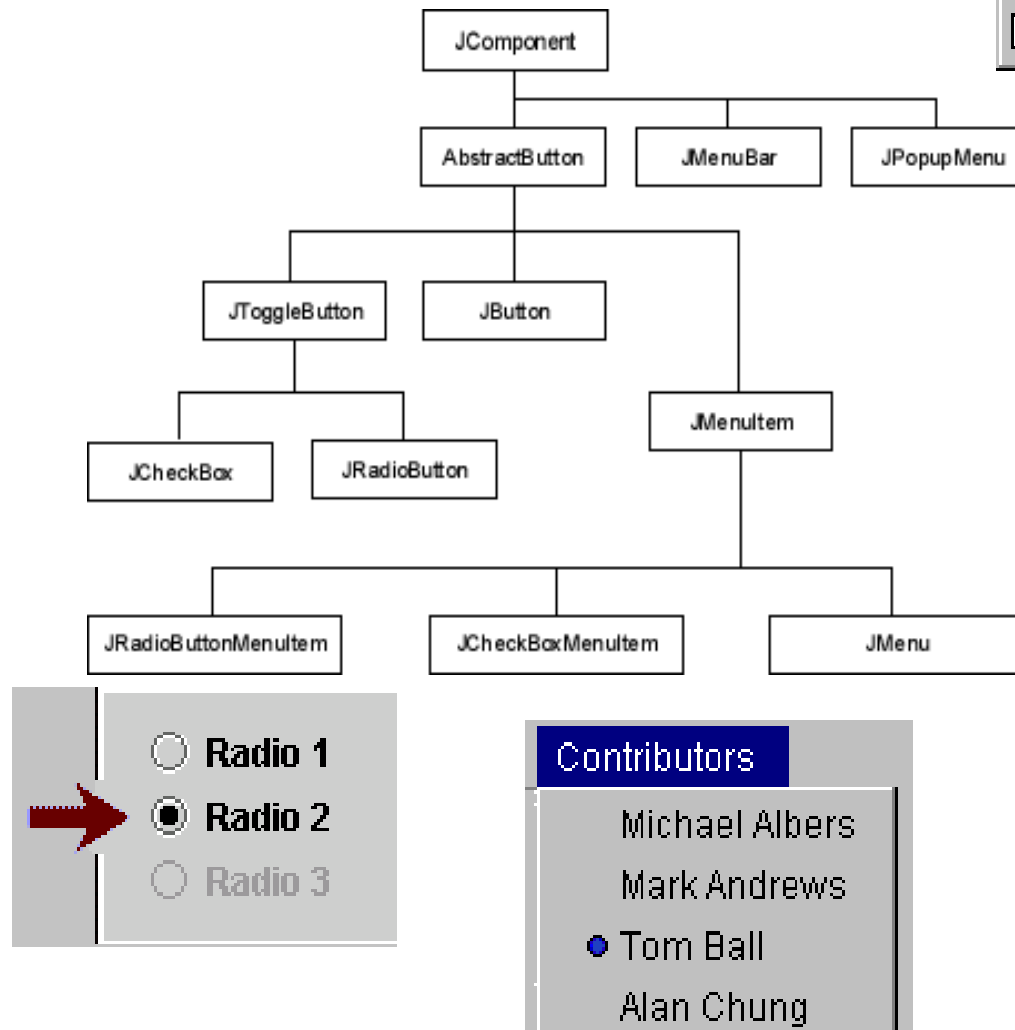
Box ← BoxLayout



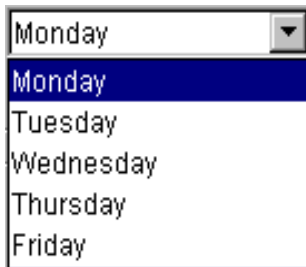
BUTTONS



MENUS



OTHER COMPONENTS



JComboBox



JApplet



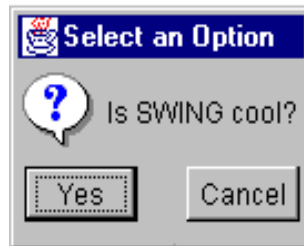
Border Interface



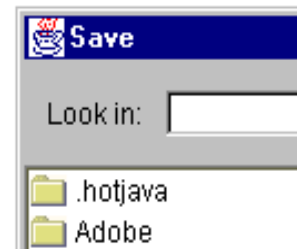
JColorChooser



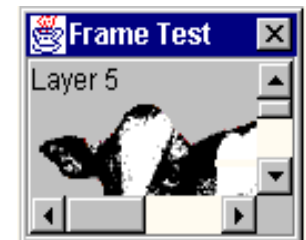
ImageIcon



JDialog



JFileChooser



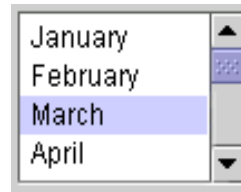
JInternalFrame



OTHER COMPONENTS



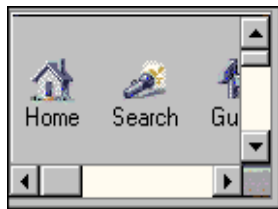
JLabel



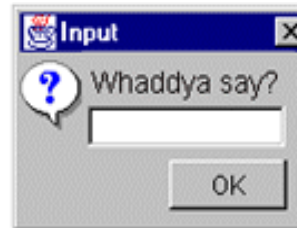
JList



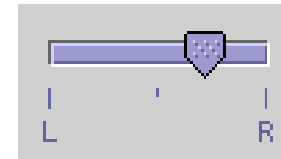
JScrollBar



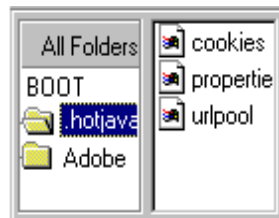
JScrollPane



JOptionPane



JSlider



JSplitPane



JTabbedPane



OTHER COMPONENTS

First Na...	Last Name
Mark	Andrews
Tom	Ball
Alan	Chung
Jeff	Dinkins

JTable

Verify that the RJ45 cable is connected to the WAN plug on the back of the Pipeline unit.

JTextArea



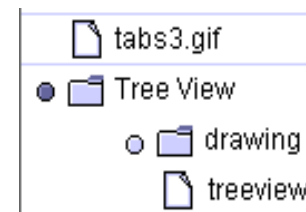
JToolBar



JToolTip

George Washington
Thomas Jefferson
Benjamin Franklin
Thomas Paine

JTextField



JTree



Atomic components (3)

Impossibile spiegare il comportamento di tutti i componenti

Poche persone conoscono tutto!! – Swing è vasto.

Riferimenti:

- Java 2 API Documentation.
- http://java.sun.com/docs/books/tutorial/uiswing/components/component_s.html



Alcuni Esempi

Pulsante con immagine

Al posto delle semplici scritte possiamo aggiungere grafica ai pulsanti con oggetti di tipo **ImageIcon** da incorporare nei bottoni stessi

```
ImageIcon img=new ImageIcon("c:\\...\\ img1.gif ");  
// legge il file .gif e lo incorpora  
// nell'oggetto img  
JButton pulsanteUno = new JButton(img);  
// passo img al costruttore del pulsante
```



Label

Componente che contiene semplicemente del testo informativo (di solo output)

Può contenere anche solo un'icona o entrambi.

I costruttori di base sono :

- **JLabel(String);** Crea un'etichetta con la stringa passata come parametro
- **JLabel(String, int);** Crea un'etichetta con la stringa passata come parametro definendone l'allineamento a mezzo di un flag
- **JLabel(String, Icon, int);** Crea un'etichetta con la stringa passata come parametro e l'icona associata. definendone l'allineamento a mezzo di un flag



Label: Allineamento

I tre flag disponibili sono:

SwingConstants.LEFT

SwingConstants.RIGHT

SwingConstants.CENTER (default)



Campo testo: JTextField

Componente che gestisce una riga di input da tastiera

Costruttori :

- **JtextField();** Genera un campo di testo vuoto
- **JtextField(int);** Genera un campo di testo di larghezza specificata
- **JtextField(String, int);** Genera un campo di testo contenente una stringa e di larghezza specificata



JTextField

setEditable(boolean); posto a “true” permette la modifica del testo contenuto, al contrario “false” non permette l'immissione di input da tastiera

boolean isEditable(); dice se il componente è modificabile o meno

setText(String); permette di immettere testo nel componente

String getText(); restituisce il contenuto in forma di stringa

String getSelectedText(); restituisce il solo testo selezionato dall'utente



Inserire passwords

Utile classe con la capacità di crittare l'output a video con un carattere a nostra scelta durante la digitazione.

Permette di definirne le dimensioni e mediante un opportuno metodo impostare il carattere per l'echo a video

JPasswordField pf = new JPasswordField(10);

- istanzia un oggetto campo password di dimensione 10

pf.setEchoChar('*');

- il classico asterisco



TextArea

Rappresenta un campo di testo in forma matriciale, con un certo numero di righe e di colonne. I costruttori sono

TextArea(int, int);

- Crea un'area di testo di dimensioni righe, colonne

TextArea(String, int ,int);

- Come sopra solo che inserisce una stringa da codice

I metodi sono **getText();** **getSelectedText()** e **setText(String)** funzionanti come per i campi di testo



JTextArea

Altre funzionalità di editing:

append(String);

- accoda del testo

insert(String, int);

- inserisce del testo alla posizione specificata

setLineWrap(boolean);

- specifica se “true” di andare a capo automaticamente a fine riga

setWrapStyleWord(boolean);

- va a capo con la parola se “true” o col singolo carattere se “false”



Esempio

```
import javax.swing.*;

public class MyJLabelJTextFieldJFrame extends JFrame {
    JLabel etichetta = new JLabel("Etichetta");
    JTextField campoDiTesto = new JTextField("Scrivi qui", 30);
    JTextArea areaDiTesto = new JTextArea("Questa é un'area “ +“di testo
di\n6 righe e 20 colonne", 6, 20);

    public MyJLabelJTextFieldJFrame() {
        super("Finestra con Etichette e Campi");
        setSize(350, 300);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        JPanel pannello = new JPanel();
        // impostiamo le proprietà dei componenti
        campoDiTesto.setEditable(true);
        areaDiTesto.setLineWrap(true);
        areaDiTesto.setWrapStyleWord(true);
    }
}
```



Esempio

```
// ora aggiungiamo i componenti al pannello
pannello.add(etichetta);
pannello.add(campoDiTesto);
pannello.add(areaDiTesto);
// rendiamo il pannello parte del nostro frame
setContentPane(pannello);
// Visualizziamo il tutto!
show();
}
public static void main(String argv[]) {
    MyJLabelJTextFieldJFrame ec =
        new MyJLabelJTextFieldJFrame();
}
}
```



Esempi

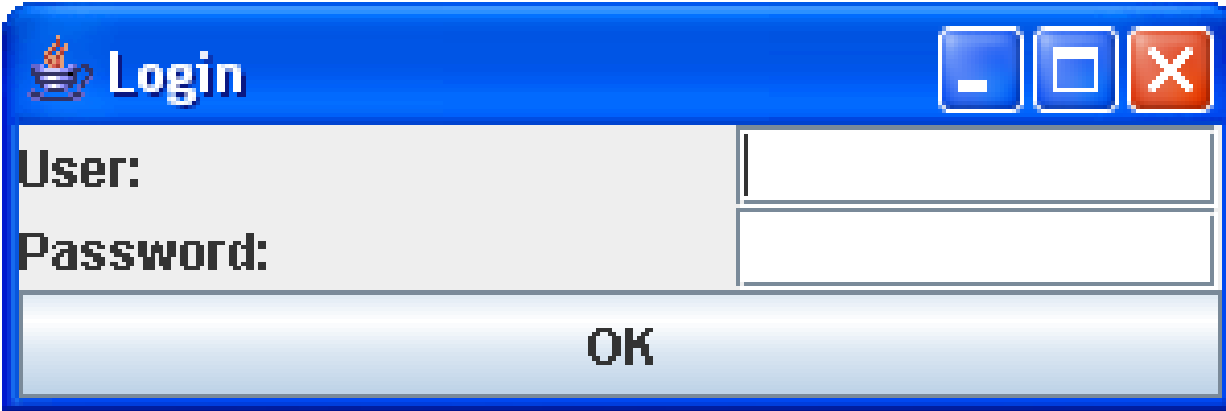
Esempi:

- Pulsanti
- PulsantiGrid
- PulsantiBorder
- PulsantiVBox
- JCompositeFrame
- Semplice Login



Esempio

Realizzare la finestra sottostante formata da un pannello ed un pulsante



The image shows a Java Swing window titled "Login". The window has a blue title bar with a coffee cup icon on the left and standard window control buttons (minimize, maximize, close) on the right. The main content area has a light gray background. It contains two labels, "User:" and "Password:", each followed by a text input field. The "User:" label is positioned above the first input field, and the "Password:" label is positioned above the second input field. At the bottom of the window, there is a large, light blue button with the text "OK" centered on it.



Test [Minimize] [Maximize] [Close]

Nuovo Contatto

InfoContatto

Nome Telefono

Cognome E-mail

☐ Uomo

☐ Donna

Azioni



Test

Cliente

Richiesta Occupazione

Informazioni Personali

Nome

Cognome

E-mail:

Telefono:

Formazione Personale

Conoscenza lingua Inglese

☐ Ottimo ☒ Buono

☐ Sufficiente ☐ Scarso

Titolo di studio

Interessi

☐ Informatica ☐ Fisica

☐ Elettronica ☐ Matematica

☐ Economia ☐ Sociologia

Invia

