

**Università degli Studi della Campania
Luigi Vanvitelli
Dipartimento di Ingegneria**

Programmazione ad Oggetti

a.a. 2020-2021

Layouts

Docente: Prof. Massimo Ficco
E-mail: massimo.ficco@unicampania.it

1

1

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.



2

Layouts

V:

AWT/Swing supportano diversi managers. Ne considereremo 4:

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

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



FlowLayout

V:

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).

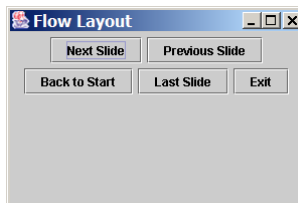
Spesso è utilizzato per inserire pulsanti in un pannello.



FlowLayout

V:

```
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

V:

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 aggizionali tra le celle.



GridLayout (cont'd)

V:

```
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"));
```

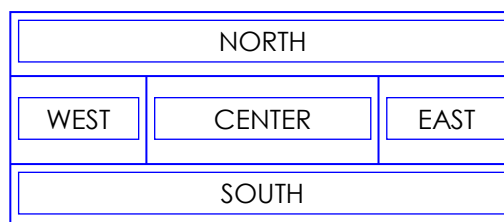
Extra
space
between
the cells



BorderLayout

V:

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



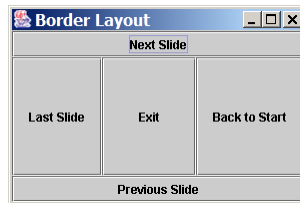
Forza le dimensioni del componente ad occupare l'intera regione.



BorderLayout (cont'd)

V:

```
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);
```



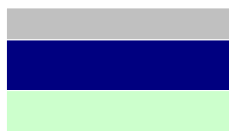
BoxLayout

V:

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)

V:

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));

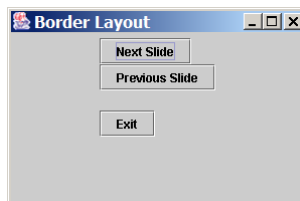
Box box2 = Box.createVerticalBox();
...
```



BoxLayout

V:

```
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*

V:

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

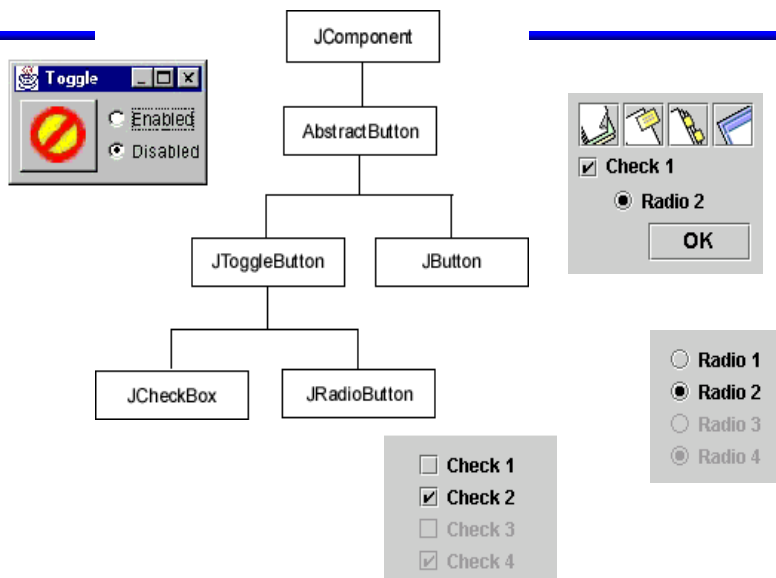
JPanel ← FlowLayout

Box ← BoxLayout



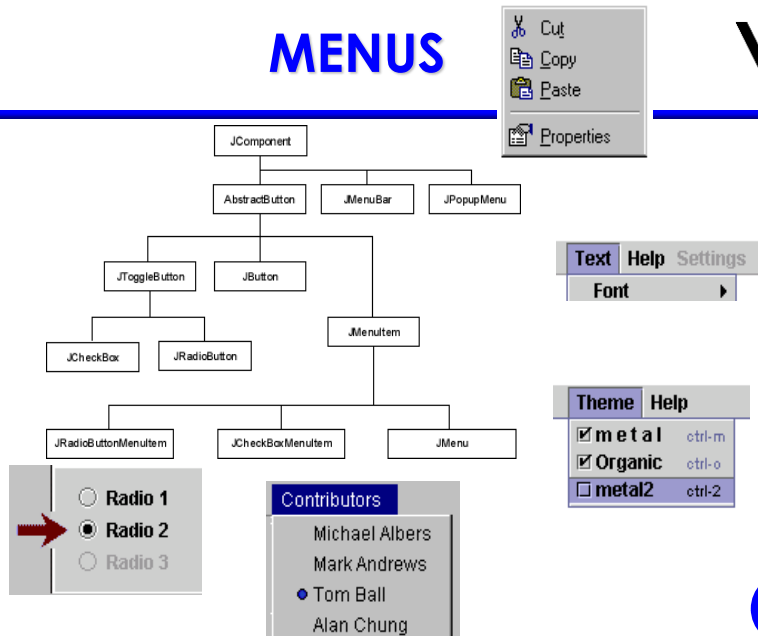
BUTTONS

V:



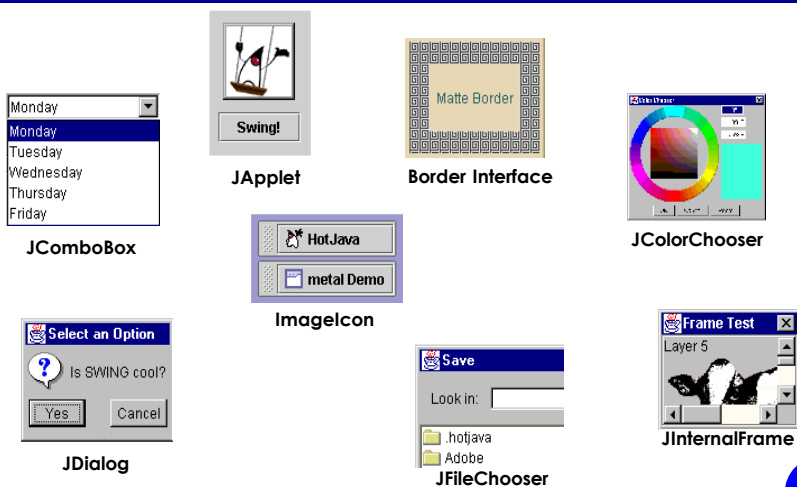
MENUS

V:



OTHER COMPONENTS

V:



OTHER COMPONENTS

V:



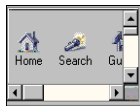
JLabel



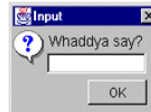
JList



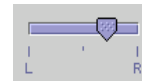
JScrollBar



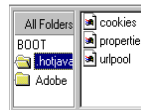
JScrollPane



JOptionPane



JSlider



JSplitPane



JTabbedPane

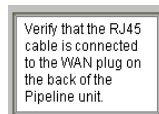


OTHER COMPONENTS

V:



JTable



JTextArea



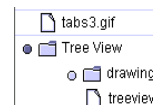
JTextField



JToolBar



JToolTip



JTree



Atomic components (3) V:

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



Pulsante con immagine V:

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 V:

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 V:

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

V:

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



JTextArea

V:

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

JTextArea(int, int);

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

JTextArea(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



TextArea

V:

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

V:

```
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

V:

```
// 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

V:

Esempi:

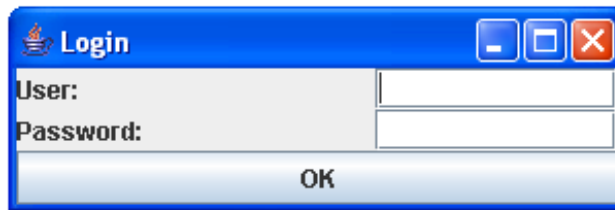
- Pulsanti
- PulsantiGrid
- PulsantiBorder
- PulsantiVBox
- JCompositeFrame
- Semplice Login



Esempio

V:

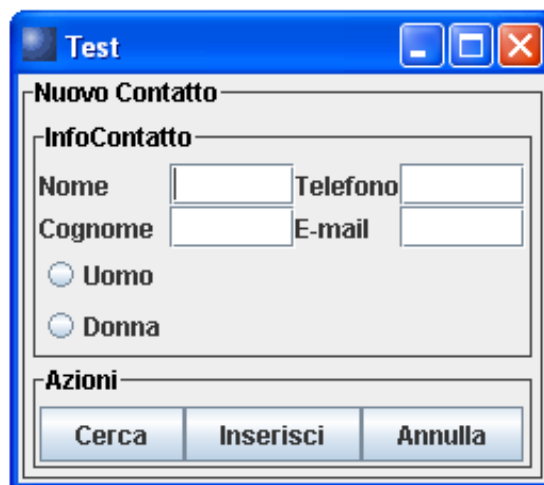
Realizzare la finestra sottostante formata da un pannello ed un pulsante



A screenshot of a Java Swing window titled "Login". The window has a blue title bar with standard OS controls. Inside, there is a light gray panel. The panel contains two text input fields: the first is labeled "User:" and the second is labeled "Password:". Below these fields is a single "OK" button.



V:



A screenshot of a Java Swing window titled "Test". The window has a blue title bar with standard OS controls. Inside, there is a light gray panel. The panel contains a section titled "Nuovo Contatto" which includes a sub-section "InfoContatto". This sub-section has four text input fields: "Nome", "Cognome", "Telefono", and "E-mail". Below these fields are two radio buttons labeled "Uomo" and "Donna". At the bottom of the panel is a section titled "Azioni" which contains three buttons: "Cerca", "Inserisci", and "Annulla".



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

V:



Gestione eventi

V:

Realizzazione di una interfaccia grafica:

- Disegno grafico: CAD
- Gestione degli eventi Scrittura codice

Ogni componente Swing è in grado di generare degli eventi generici o particolari (Ad esempio per il pulsante non siamo particolarmente interessati al fatto che sia attraversato dal mouse, ma solo al click)

Quindi cominceremo col gestire per ogni componente gli eventi principali



JButton

V:

Per registrare un gestore degli eventi per un pulsante occorre utilizzare il metodo:

`addActionListener(ActionListener a);`

del componente JButton.

Tale metodo si aspetta come parametro una classe in grado ascoltare gli eventi generati dal pulsanti e di gestirli

Tale classe per poter essere utilizzata da JButton deve implementare l'interfaccia ActionListener



ActionListener interface

V:

```
class ButtonListener implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        .....  
    }  
  
}
```

Quando viene pigiato il pulsante automaticamente viene invocato il metodo actionPerformed



Esempio

V:

```
public class Button2 extends JFrame {
    private JButton b1 = new JButton("Button 1");
    b2 = new JButton("Button 2");
    private JTextField txt = new JTextField(10);

    public Button2(){
        super("2 pulsanti");
        setSize(400,300);
        JPanel p=new JPanel();
        b1.addActionListener(new ButtonListener());
        b2.addActionListener(new ButtonListener());
        p.add(b1); p.add(b2); p.add(txt);}

    public static void main(String[] args){
        JFrame frame=new Button2();
        frame.show();
    }}

```



Programmazione ad Oggetti - Prof. Massimo Ficco

37

Esempio

V:

```
class ButtonListener implements ActionListener {
    JTextField txt;

    //public ButtonListener(JTextField t){txt=t;}

    public void actionPerformed(ActionEvent e) {
        String name = ((JButton)e.getSource()).getText();
        txt.setText(name);
    }
}

```



Programmazione ad Oggetti - Prof. Massimo Ficco

38

Generalizziamo

V:

In conclusione per gestire un evento occorre:

1. Definire una classe che implementa una interfaccia Listener
2. Ridefinire tutti i metodi dell'interfaccia
3. Aggiungere il Listener al componente



Tipi di eventi

V:

Event, listener interface and add- and remove-methods	Components supporting this event
ActionEvent ActionListener addActionListener() removeActionListener()	JButton, JList, JTextField, JMenuItem and its derivatives including JCheckBoxMenuItem, JMenu, and JpopupMenu.
AdjustmentEvent AdjustmentListener addAdjustmentListener() removeAdjustmentListener()	JScrollbar and anything you create that implements the Adjustable interface.
ComponentEvent ComponentListener addComponentListener() removeComponentListener()	*Component and its derivatives, including JButton, JCheckBox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, JFrame, JLabel, JList, JScrollbar, JTextArea, and JTextField.



Tipi di eventi

V:

ContainerEvent ContainerListener addContainerListener() removeContainerListener()	Container and its derivatives, including JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, and JFrame.
FocusEvent FocusListener addFocusListener() removeFocusListener()	Component and derivatives*.
KeyEvent KeyListener	Component and derivatives*.



Programmazione ad Oggetti - Prof. Massimo Ficco

41

Tipi di eventi

V:

Event, listener interface and add- and remove-methods	Components supporting this event
addKeyListener() removeKeyListener()	
MouseEvent (for both clicks and motion) MouseListener addMouseListener() removeMouseListener()	Component and derivatives*.
MouseEvent ⁸ (for both clicks and motion) MouseMotionListener addMouseMotionListener() removeMouseMotionListener()	Component and derivatives*.
WindowEvent WindowListener addWindowListener() removeWindowListener()	Window and its derivatives, including JDialog, JFileDialog, and JFrame.
ItemEvent ItemListener addItemListener() removeItemListener()	JCheckBox, JCheckBoxMenuItem, JComboBox, JList, and anything that implements the ItemSelectable interface.
TextEvent TextListener addTextListener() removeTextListener()	Anything derived from JTextComponent, including JTextArea and JTextField.



Programmazione ad Oggetti - Prof. Massimo Ficco

42

Listeners e interfacce

V:

Listener interface w/ adapter	Methods in interface
ActionListener	actionPerformed(ActionEvent)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
ComponentListener ComponentAdapter	componentHidden(ComponentEvent) componentShown(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent)
ContainerListener ContainerAdapter	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
FocusListener FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)
KeyListener KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)



Tipi di eventi

V:

MouseListener MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
MouseMotionListener MouseMotionAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
WindowListener WindowAdapter	windowOpened(WindowEvent) windowClosing(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent)
ItemListener	itemStateChanged(ItemEvent)



Utilizzare listener adapters V:

Se volessi implementare una interfaccia listener dovrei impazzire poiché non dovrei dimenticare nessun metodo

Per ogni listener esiste un adapter che implementa l'interfaccia corrispondente con tutti metodi vuoti

Esempio:

```
class MyMouseListener extends MouseAdapter {  
    public void MouseClicked(MouseEvent e) {  
        // Respond to mouse click...  
    }  
}
```



V:

A cura del
Prof. Massimo Ficco
e del
Prof. Salvatore Venticinque

