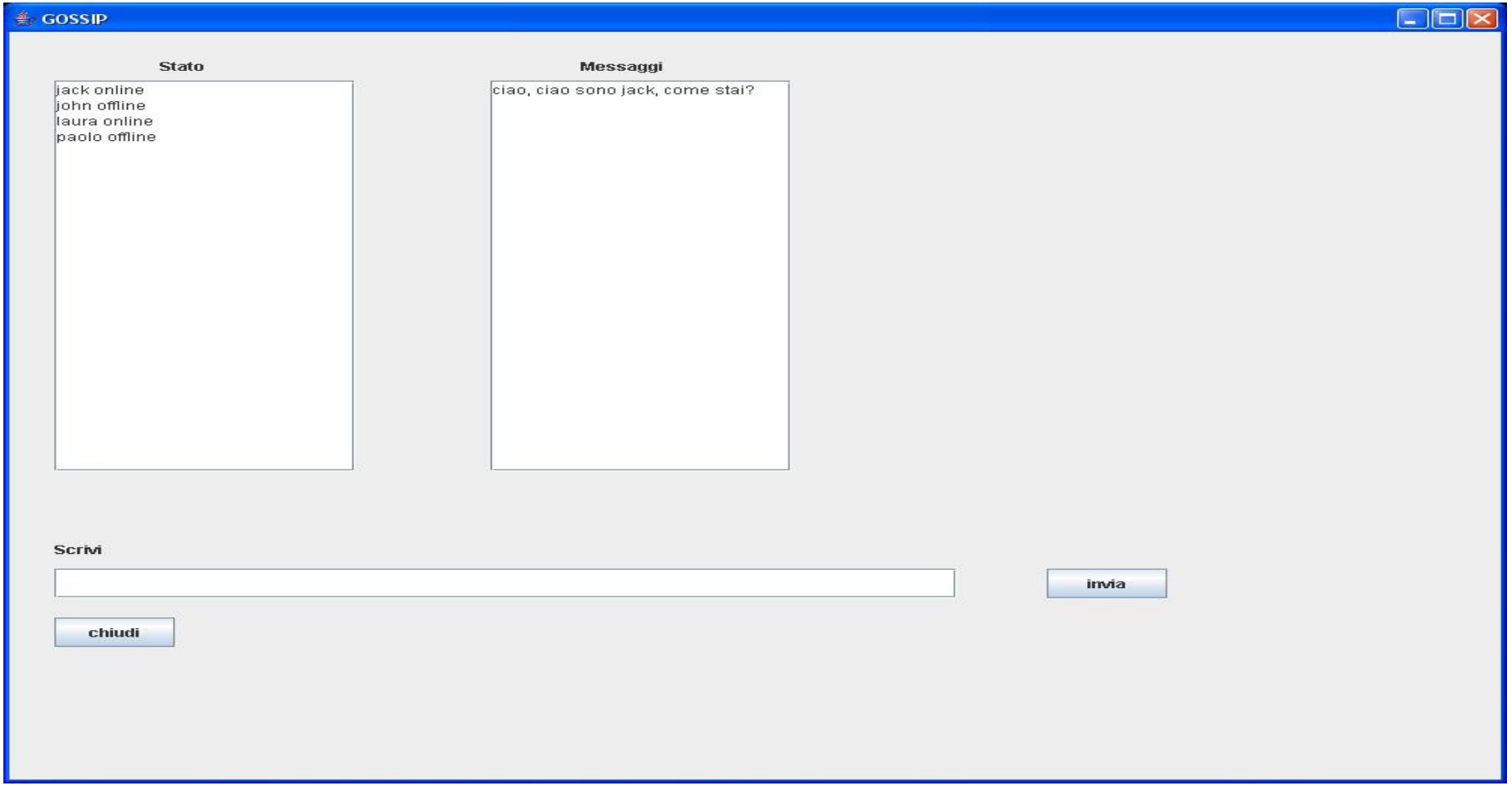
Lezione n.12
LPR Informatica Applicata
GUI programming

19/05/2008

Laura Ricci

UNA SEMPLICE INTERFACCIA PER INSTANT MESSAGING



JAVA GUI: COMPONENTI BASE

```
import java.awt.*;
import javax.swing.*;
public class GossipInterface extends JFrame {
        public GossipInterface(    ) {
        // qui alloco gli oggetti nell'interfaccia   }

public static void main (String args  [  ]  )
            {GossipInterface gi = new GossipInterface(  );
             gi.setVisible(true);      }     }
```

JAVA GUI: COMPONENTI BASE

```
public GossipInterface(   ) {
    setTitle("GOSSIP");
    setSize(1000,800);
    JPanel panel1= new Jpanel( );
    panel1.setLayout(null);   ..... }
```

- setTitle, setSize setBackground.....imposta alcune proprietà della finestra creata
- Alloco un pannello, cioè un contenitore all'interno del quale allocare componenti dell'interfaccia
- Quando il pannello è completo, va aggiunto al frame

JAVA GUI: COMPONENTI DI BASE

```
JLabel statusLabel = new Jlabel ("Stato");
statusLabel.setBounds(100,20,30,30);
panel1.add(statusLabel);
```

- JLabel = è un componente che consente di visualizzare una stringa (etichetta)
- Utile ad esempio per definire intestazioni di riferimento per altri componenti
- La Jlabel viene aggiunta al pannello (metodo add) dopo essere stata dimensionata (metodo setBounds)

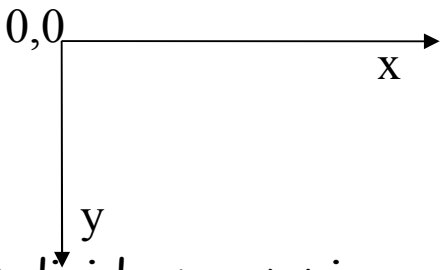
JAVA GUI: COMPONENTI DI BASE

public void setBounds(**int** x, **int** y, **int** width, **int** height)

- Metodo ereditato dalla classe awt.component
- Stabilisce la locazione del componente all'interno del pannello e la dimensione del componente

Parametri

- x,y - individuano la posizione del vertice superiore sinistro in un sistema di assi cartesiani orientati nel modo seguente



- width, height - individuano ampiezza ed altezza del componente

JAVA GUI: COMPONENTI DI BASE

```
JTextArea statusArea = new JTextArea( );

statusArea.setEditable(true);

JScrollPane scrollPaneStatus = new JScrollPane(statusArea);

scrollPaneStatus.setBounds(30,50,200,400);

panel1.add(scrollPaneStatus);
```

- JTextArea -definisce una componente in cui possono essere scritte una o più linee di testo
- SetEditable – proprietà che consente di specificare se l'area di testo può essere editata o meno dall'utente
- JscrollPane – aggiunge barre di scorrimento all'area

JAVA GUI: COMPONENTI DI BASE

```
statusArea.append("jack online\n");
statusArea.append("john offline\n");
statusArea.append("laura online\n");
statusArea.append("paolo offline\n");
```

- Metodo append: consente di aggiungere linee di caratteri ad una text area
- Le linee sono separate da un carattere di new line \n

JAVA GUI: COMPONENTI DI BASE

```
JLabel statusMsg = new JLabel("Messaggi");
statusMsg.setBounds(380,20,80,30);
panel1.add(statusMsg);
JTextArea printArea= new JTextArea();
printArea.setEditable(true);
JScrollPane scrollPane = new JScrollPane(printArea);
scrollPane.setBounds(320,50,200,400);
panel1.add(scrollPane);
printArea.append("ciao, ciao sono jack, come stai?");
```

JAVA GUI: COMPONENTI DI BASE

```
JLabel inputLabel = new JLabel("Scrivi");
inputLabel.setBounds(30,480,100,100);
panel1.add(inputLabel);
JTextField InputArea = new JTextField();
InputArea.setBounds(30,550,600,30);
InputArea.setEditable(true);
panel1.add(InputArea);
```

- JTextField = Campo di testo, si può leggere il valore inserito in questo campo dall'utente

JAVA GUI: COMPONENTI DI BASE

```
JButton invia = new JButton("invia");
invia.setEnabled(true);
invia.setBounds(690,550,80,30);
panel1.add(invia);
```

- JButton- E' un pulsante che l'utente preme per lanciare una specifica azione
- Nel nostro caso l'azione corrisponde all'invio del testo editato nell'area JtextField
- Al costruttore viene passata la stringa che rappresenta il testo visualizzto sul pulsante

JAVA GUI:COMPONENTI DI BASE

```
JButton chiudi = new JButton("chiudi");
chiudi.setEnabled(true);
chiudi.setBounds(30,600,80,30);
panel1.add(chiudi);
getContentPane( ).add(panel1);
```

- Questo pulsante è utilizzato per chiudere l'applicazione
- Infine il pannello creato viene aggiunto al Jframe (la finestra) mediante la funzione getContentPane().

JAVA GUI:GESTIONE DEGLI EVENTI

- Materiale didattico: C. Horstmann, capitolo 11
- Quando un utente esegue qualche operazione su una finestra (JFrame) viene generato un evento asincrono
  - digitazione di un tasto
  - spostamento del mouse
  - click di un pulsante del mouse
  - selezione di un elemento dell'intefaccia grafica (es: un bottone)
- Il gestore delle finestre invia una segnalazione al programma identificando l'evento mediante un codice
- L' applicazione, in generale, deve gestire solamente un sottoinsieme degli eventi generati dall'interfaccia.

JAVA GUI: GESTIONE DEGLI EVENTI

- L'applicazione deve definire un gestore di eventi (Listener) per ogni evento che interessa gestire

- JAVA definisce interfacce diverse per ogni tipo di evento che può essere generato dall'interfaccia (KeyListener, MouseListener, WindowListener, ActionListener,....)

- Il gestore degli eventi viene definito mediante una implementazione di questa interfacci

JAVA GUI: GESTIONE DEGLI EVENTI

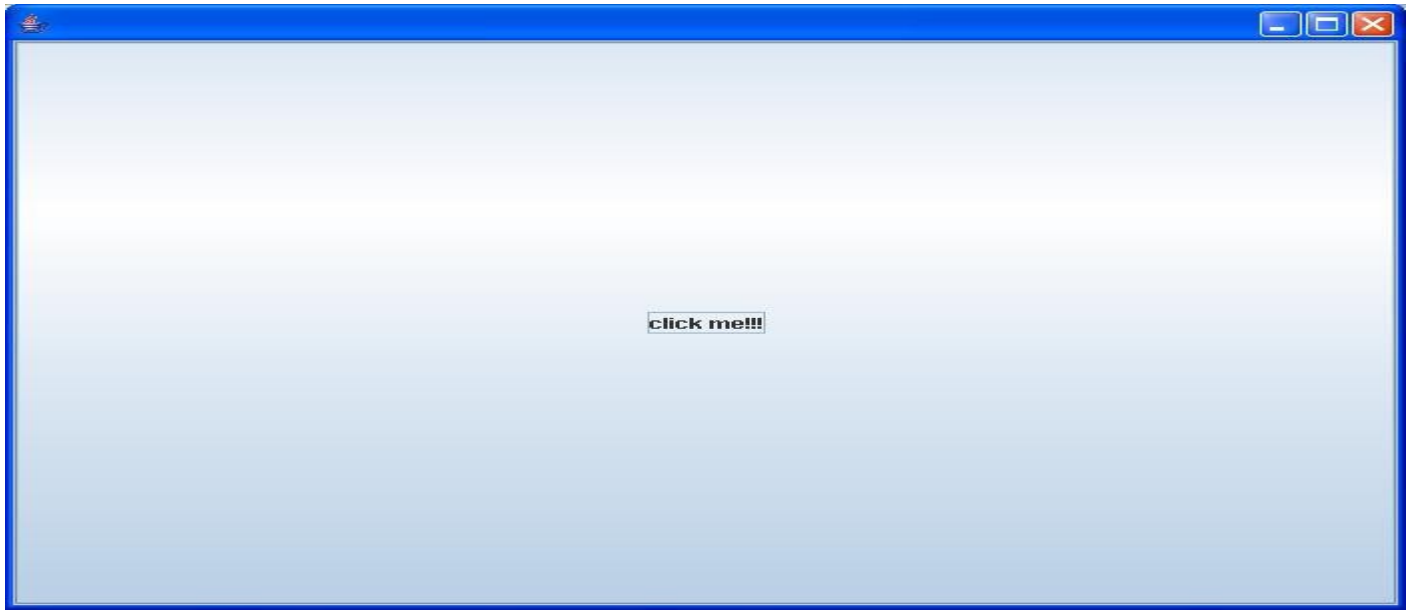
- Consideriamo un'applicazione di instant messaging
- E' necessario definire un insieme di pulsanti per interagire con l'applicazione, ad esempio il tasto invia, che consente di inviare un messaggio, o quello che consente di chiudere una finestra
- Gli eventi interessanti per l'applicazione sono quelli legati alla selezione di questi bottoni
- Interfaccia utilizzata

```
public interface ActionListener {
        void actionPerformed (actionEvent event);   }
```

- Compito del programmatore è implementare il metodo actionPerformed, in modo che esegua il codice da eseguire quando viene selezionato il pulsante

GESTIONE DEGLI EVENTI: UN ESEMPIO

- Consideriamo la seguente interfaccia:



Quando viene selezionato il bottone,l'applicazione stampa a linea di comando il messaggio "I was clicked"

JAVA GUI: GESTIONE DEGLI EVENTI

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ClickListener implements ActionListener{
      public void actionPerformed(ActionEvent event)
            {
                  System.out.println("I was clicked");     }
            }
```

- il parametro event contiene ulteriori dettagli sull'evento, ad esempio l'istante in cui esso è avvenuto

JAVA GUI: GESTIONE DEGLI EVENTI

```
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
public class interfaceevents{
private static final int FRAME_WIDTH = 700;
private static final int FRAME_HEIGHT = 500;
public static void main(String args[])
      { JFrame frame = new JFrame();
      JButton button = new JButton("click me!!!");
      frame.add(button);
```

JAVA GUI: GESTIONE DEGLI EVENTI

```
ActionListener listener = new ClickListener(  );
button.addActionListener(listener);
frame.setSize(FRAME_WIDTH,FRAME_HEIGHT);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);
    }
        }
```

UN CONTATORE DI 'CLICKS'

Modifichiamo il programma precedente in modo che, ogni volta che viene selezionato il bottone, venga visualizzato il numero di volte che il pulsante è stato selezionato fino a quel momento

UN CONTATORE DI 'CLICKS'

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

public class swingexample extends JFrame
{
    private static String labelPrefix = "Number of clicks: ";
    private int numClicks = 0;
    JLabel label = null;
    JButton button = null;
```

UN CONTATORE DI CLICKS

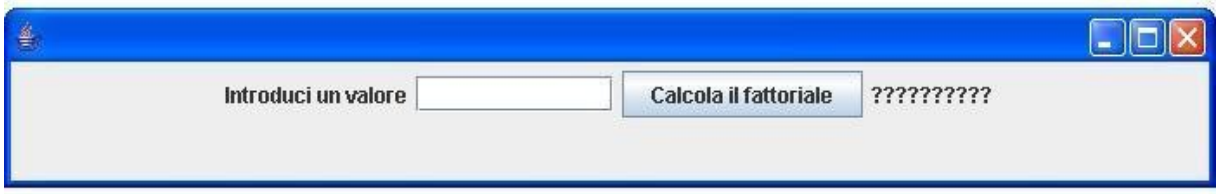
```
public swingexample( )
  {
     label = new JLabel(labelPrefix + "0    ");
     button = new JButton("Click me!");
     button.addActionListener(new ActionListener()
     {
        public void actionPerformed(ActionEvent e)
        {   numClicks++;
            label.setText(labelPrefix + numClicks);
        }
     });
```

UN CONTATORE DI CLICKS

```
    Container panel = getContentPane ();
    panel.setLayout(new GridLayout(2, 1));
    panel.add(button);
    panel.add(label);
    addWindowListener(new WindowAdapter( )
        {public void windowClosing(WindowEvent e)
            { System.exit(0);} } );
    setVisible(true);
}
```

JAVA GUI: CAMPI DI TESTO

- Consideriamo ora la seguente interfaccia, mediante la quale viene richiesto un valore intero x, ed, in corrispondenza della selezione del pulsante, calcola il fattoriale di x.



JAVA GUI: GESTIONE DEGLI EVENTI

```
import javax.swing.*;
import java.awt.*;  import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;
public class FattorialeGUI
    { public static int fattoriale(int n)
      {if (n==0) return 1;
           else return (n*fattoriale(n-1));
      };
```

JAVA GUI: GESTIONE DEGLI EVENTI

```
public static void main (String [ ] args)
    {final int FRAME_WIDTH = 700;
    final int FRAME_HEIGHT =100;

    JFrame frame = new Jframe( );

    JLabel fattlabel = new JLabel("Introduci un valore");

    final int FIELD_WIDTH = 10;
    final JTextField valuefield = new JtextField(FIELD_WIDTH);

    final JLabel resultLabel = new JLabel("??????????");
```

JAVA GUI: GESTIONE DEGLI EVENTI

```
 JButton button = new JButton("Calcola il fattoriale");
button.addActionListener ( new ActionListener()
{ public void actionPerformed(ActionEvent e)
      { int n;
      try {
            n = Integer.parseInt(valuefield.getText());
            if (n<0) resultLabel.setText("impossibile");
             else resultLabel.setText("risultato"+ fattoriale(n));
       } catch (NumberFormatException ne)
                  { resultLabel.setText("dato non numerico");} } });
```

JAVA GUI: GESTIONE DEGLI EVENTI

```
JPanel panel = new Jpanel( );
panel.add(fattlabel);
panel.add(valuefield);
panel.add(button);
panel.add(resultLabel);
frame.add(panel);
frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);   }}
```

RIFERIMENTI

- Bruce Eckel, Thinking in JAVA, Volume 3, Concorrenza ed Interfacce Grafiche., 4 edizione.
- JAVA Swing, O' Reilly