Introduzione al package javax.swing

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Obiettivi della lezione

- In questa lezione introduciamo alcuni strumenti che Java rende disponibili per la creazione di interfacce utente a manipolazione diretta, dette anche WYSISYG (*what you see is what you get), con un elevato grado di* portabilità tra piattaforme diverse.
- Vedremo alcune classi del package **javax.swing,** utili per costruire finestre, menu, bottoni ecc., e introdurremo il meccanismo di gestione degli eventi che sovraintende all'interazione tra utente e programma.

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Da awt a swing

- Fin dalle prime versioni di Java i progettisti del linguaggio misero a disposizione dei programmatori lo *Abstract Window Toolkit* (***awt), un insieme di classi che*** permettono di creare e manipolare gli elementi visuali di un'interfaccia (finestre, menu, bottoni, ecc.) in modo portabile, senza essere vincolati ad una specifica piattaforma hardware/software.
- Nonostante l'obiettivo dichiarato di permettere la costruzione di *GUI che* funzionassero bene su qualunque piattaforma, all'inizio **awt di fatto produceva** interfacce egualmente mediocri sui vari sistemi gestiti. Solo successivamente, con l'introduzione delle *Java Foundation Classes (JFC) le cui componenti GUI prendono* il nome di **swing, è stato raggiunto un livello qualitativo soddisfacente.**

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Che cos'è swing?

swing è il nome di un *package (in realtà, di una famiglia di packages) che* mette a disposizione funzionalità per la costruzione di interfacce grafiche a manipolazione diretta. Le classi di **swing consentono di definire** componenti interattive e controlli (finestre, menu, bottoni ecc.), grafica in 2D, supporto al *drag & drop (trascinamento di oggetti sullo schermo),* funzioni di accessibilità per utenti disabili; un aspetto particolare è costituito dalla possibilità di scelta, da parte del programmatore, del *look & feel col quale l'interfaccia si presenta all'utente: oltre alla modalità tipica* Java (usata in tutti gli esempi di questa e delle successive lezioni) si può scegliere che finestre e menu appaiano costruiti secondo lo stile di Windows, di Macintosh o di Linux, o ancora secondo stili speciali definiti dal programmatore.

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Implementazione di swing

- L'implementazione delle classi **swing è in puro Java, senza uso di codice** nativo (come invece accadeva con **awt). Per questo motivo le funzionalità** offerte da **swing non sono ristrette al minimo comune denominatore** delle funzioni disponibili in tutti gli ambienti operativi (Windows, Unix, Macintosh) su cui Java viene portato.
- Le componenti **swing si dicono pertanto *lightweight (leggere), termine*** che si contrappone allo *heavyweight (pesante) con cui vengono descritte* le componenti **awt (che fanno uso di codice nativo, e sono pertanto** implementate diversamente sulle varie piattaforme). Ne derivano vantaggi significativi in termini di prestazioni, portabilità ed estensione delle funzionalità.

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Classificazione delle componenti swing

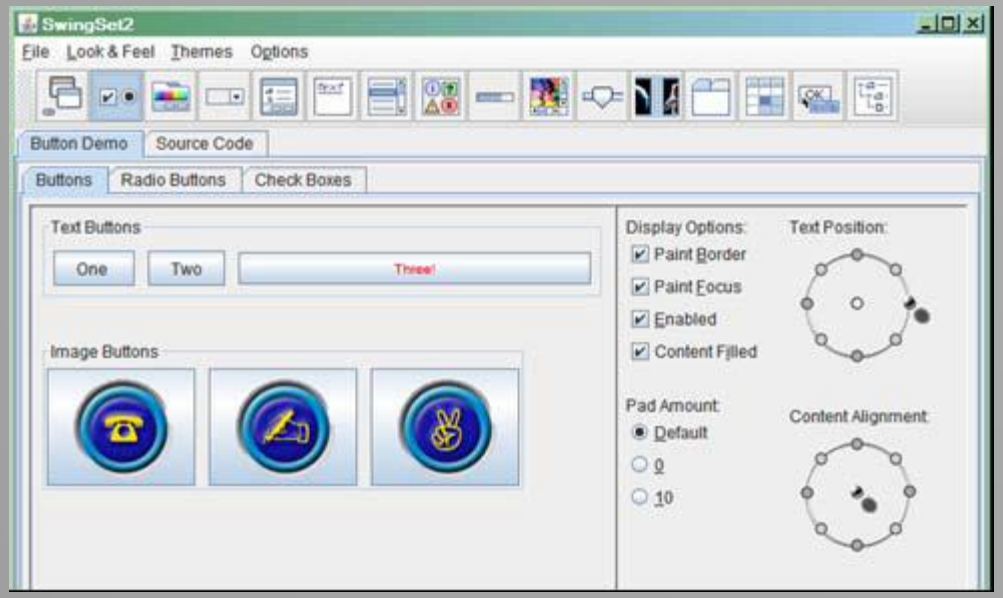
Le varie componenti d'interfaccia **swing** vengono classificate **in 6 categorie:**

- Contenitori di primo livello: le finestre che ospitano un'applicazione, un'applet o un dialogo, e costituiscono la radice della gerarchia di contenimento.
- Contenitori d'uso generale: contenitori intermedi (contenuti in altri contenitori), utili in varie circostanze.
- Contenitori speciali.
- Controlli base: componenti "atomiche" (foglie terminali della gerarchia di contenimento) la cui funzione è in genere quella di acquisire input dall'utente, e visualizzare eventualmente il proprio stato.
- Visualizzatori di informazioni in sola lettura: componenti atomiche che servono essenzialmente a fornire informazioni all'utente.
- Visualizzatori di informazioni editabili: componenti atomiche che forniscono informazioni all'utente e gli consentono di modificarle.

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

La classe JButton

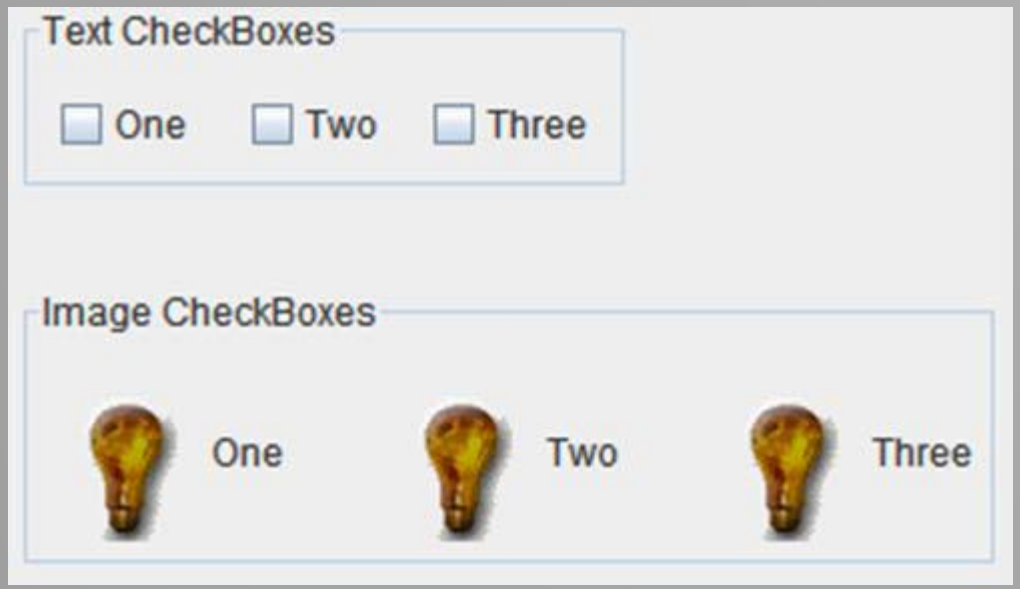
Le classi **swing** permettono la realizzazione di: bottoni, *radio buttons, check buttons* e controllarne diverse caratteristiche grafiche.



Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

La classe JCheckBox

I *check boxes (classe* **JCheckBox) sono bottoni di** tipo particolare, in genere organizzati in gruppi; in ogni momento un qualunque numero di *check boxes appartenenti a un* gruppo (classe **ButtonGroup)** può essere selezionato. I *check boxes possono anche essere* rappresentati da icone e possono essere contenuti in un menu (classe **JCheckBoxMenuItem).**



Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

La classe JRadioButton

I *radio buttons (classe* **JRadioButton) sono simili ai** *check boxes, ma al massimo uno* dei *radio buttons appartenenti a* un gruppo (**ButtonGroup) può** essere selezionato in un dato momento. Se ne seleziono un altro, il primo viene automaticamente deselezionato. I *radio buttons servono per* consentire all'utente di scegliere un'opzione tra molte.

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Esempi

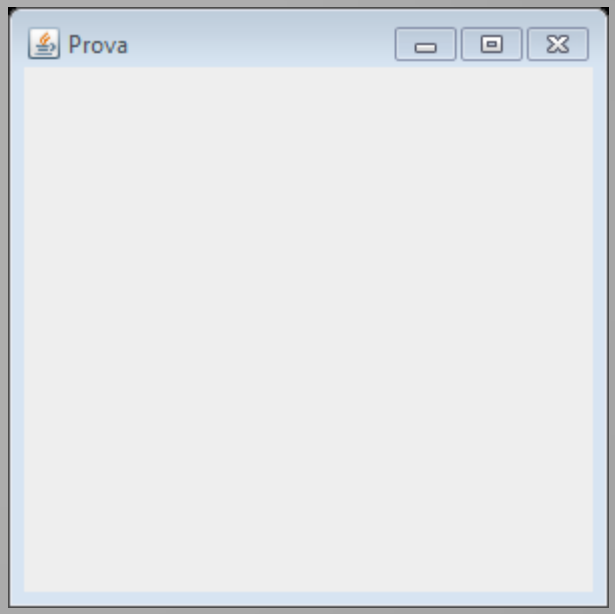
```
//file Esempio11_1_1.java
package progetto11;

import javax.swing.JFrame;

public class Esempio11_1 extends JFrame {

public Esempio11_1(String titolo) {
super(titolo);
/*
 * Qui il codice che costruisce l'applicazione
 */
setDefaultCloseOperation(EXIT_ON_CLOSE);
setSize(300, 300);
setVisible(true);
}

public static void main(String args[]) {
new Esempio11_1("Prova");
}
}
```



Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Esempi aggiunta di componenti

Supponiamo ora di voler aggiungere alla finestra alcuni controlli: a titolo di esempio, vogliamo mettere un bottone e un'etichetta. Il codice viene modificato come appare nella slide successiva(in evidenza i cambiamenti rispetto alla versione precedente):

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Esempi aggiunta di componenti

```
package progetto11;
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
public class Esempio11_2 extends JFrame {
private JButton bottone;
private JLabel etichetta;
public Esempio11_2(String titolo) {
super(titolo);
bottone = new JButton("Premi qui");
etichetta = new JLabel("Sto aspettando...");
getContentPane().setLayout(new FlowLayout());
getContentPane().add(bottone);
getContentPane().add(etichetta);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setSize(300, 300);
setVisible(true);
}
public static void main(String args[]) {
new Esempio11_2("Prova");
}
}
```

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Analisi di Esempio11_2

Il bottone (classe **JButton) e l'etichetta (classe JLabel) devono essere** dichiarati come variabili della classe e istanziate nel costruttore.

• Bisogna inoltre identificare, nel **JFrame, la zona predisposta a contenere le** componenti: un'istanza di **Container che viene ritornata dal metodo JFrame.getContentPane(). A questo contenitore possiamo aggiungere il** bottone e l'etichetta (o qualunque altra componente) usando il metodo **add().**

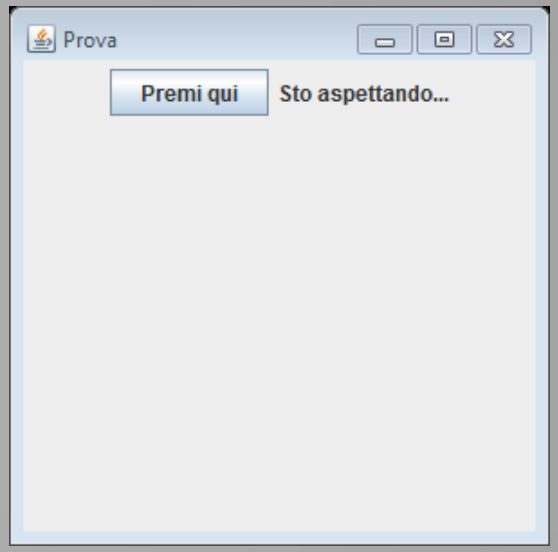
• Prima, però, bisogna stabilire come le componenti verranno allocate nello spazio della finestra: è necessario associare al contenitore un *layout manager, che si* occuperà di collocare le componenti secondo una determinata strategia, e ricalcolerà automaticamente la posizione di ognuna se, ad esempio, la finestra principale verrà ridimensionata.

• Il *layout manager che usiamo in questo programma è un'istanza della classe* **FlowLayout: le componenti vengono allocate da sinistra a destra e dall'alto in** basso. Altre classi di *layout manager consentono di organizzare lo spazio secondo* diverse strategie, ad esempio impilando le componenti in verticale, o secondo una matrice bidimensionale.

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Analisi di Esempio11_2

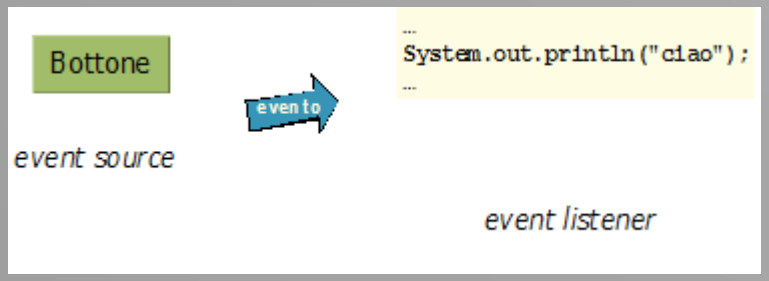
La figura a lato mostra la finestra dotata di bottone ed etichetta risultato dell'esecuzione di **Esempio11_2.**
Nota che, se la larghezza della finestra non fosse sufficiente ad ospitare i due controlli affiancati, l'etichetta verrebbe presentata sotto il bottone: prova a restringere la finestra (trascina col mouse il bordo di destra) per verificarlo.
Il codice non è ottimizzato ancora per un corretto funzionamento in quanto se si clicca sul bottone "premi qui" non succede nulla, poiché devono essere associati dei comportamenti ai bottoni ed i dettagli su come devono essere gestiti gli eventi.



Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Gestione degli eventi

Il *controllo (il bottone del nostro esempio) genera un evento quando viene* attivato; il programmatore deve predisporre un *event handler che gestisca tale* evento, e associarlo al controllo.
In generale, ogni volta che l'utente di un programma scrive qualcosa, seleziona col mouse un controllo del programma, sposta o ridimensiona una finestra ecc., *viene generato un* ***evento. Ogni evento ha una sorgente (event source), che è il controllo*** che l'ha generato: ad esempio, il bottone che è stato premuto dall'utente. In tale oggetto possono essere *registrati uno o più* ***ascoltatori (event listener), cioè*** oggetti in grado di intraprendere le azioni necessarie a gestire l'evento. La figura qui sotto schematizza questo meccanismo di *delega (infatti il controllo delega* all'*event listener il compito di gestire l'evento).*



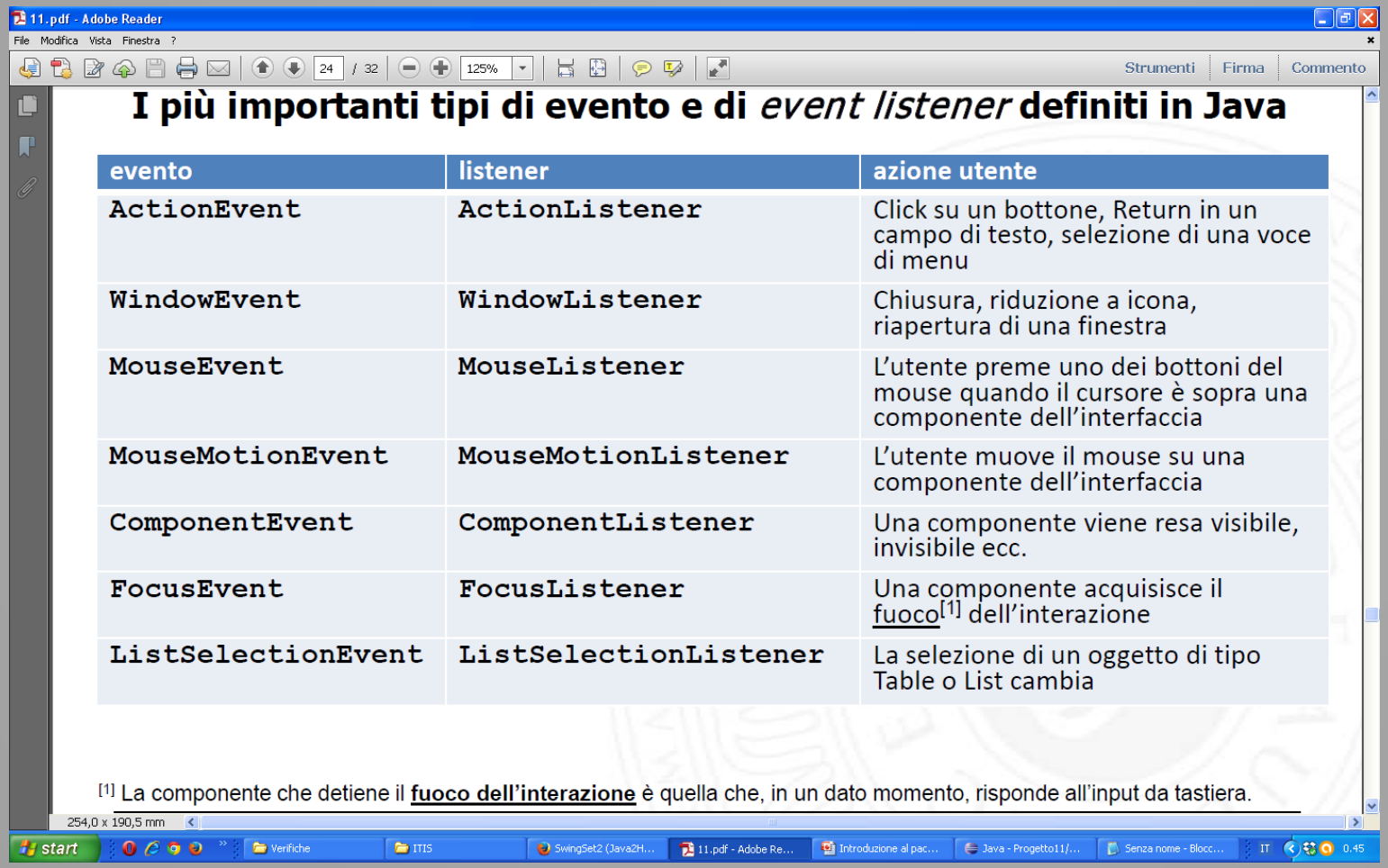
Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Event Listener

Ad ogni *event source è possibile associare più di un event listener: in* questo modo, quando si verifica l'evento tutti gli ascoltatori vengono attivati sequenzialmente, nell'ordine in cui sono stati registrati. D'altra parte uno stesso *event listener può essere associato a vari event source,* permettendo così di riusare gli stessi comportamenti in situazioni diverse.
Ad esempio, una stessa funzione potrebbe essere associata a un bottone, a un elemento di menu, a una combinazione di tasti…
Ogni evento è rappresentato da un oggetto che contiene informazioni relative all'evento stesso, tra cui ad esempio il riferimento all'*event source.* Tale oggetto è passato a parametro ai metodi dell'*event listener.*
In Java sono definiti vari tipi di eventi, cui corrispondono altrettanti tipi di *event listener. La tabella nella slide successiva riporta gli esempi di uso più* frequente. Le classi riportate nella tabella sono definite nel package **java.awt.event.**

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

I più importanti tipi di evento e di event listener definiti in Java



Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Come si scrive il codice di gestione di un evento?

In Java il codice dell'*event listener va inserito in un'apposita classe, che si* impegna a implementare un determinato insieme di metodi (una *interface). Quando la sorgente dell'evento è un bottone, l'event listener* deve essere istanza di una classe che implementi l'interfaccia **ActionListener.**

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Codice completo

```
package progetto11;

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

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class Esempio11_3 extends JFrame {
private JButton bottone;
private JLabel etichetta;
```

→ segue

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Codice completo

```
public Esempio11_3(String titolo) {
super(titolo);
bottone = new JButton("Premi qui");
etichetta = new JLabel("Sto aspettando...");
bottone.addActionListener(new Ascoltatore(etichetta));
getContentPane().setLayout(new FlowLayout());
getContentPane().add(bottone);
getContentPane().add(etichetta);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setSize(300, 300);
setVisible(true);
}
```

→ segue

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Codice completo

```
public static void main(String args[]) {
new Esempio11_3("Prova");
}
}

class Ascoltatore implements ActionListener {
JLabel etic;
static int nVolte = 0;

Ascoltatore(JLabel l) {
etic = l;
}

public void actionPerformed(ActionEvent e) {
nVolte++;
etic.setText("Bottone premuto " + nVolte + " volte");
}
}
```

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)

Esercizi

All'indirizzo

http://www.vaegar.f9.co.uk/java2html/examples/SwingSet2_demo/SwingSet2.java.html

Trovi l'applicazione swingset2 che mostra le principali funzionalità offerte da **swing.**
L'applicazione **SwingSet2** dimostra numerosi tipi di controlli che possono essere inseriti nei nostri programmi (ri)usando le classi **swing.**
SwingSet2 è uno strumento molto utile, perché per ogni tipo di componente *GUI esemplificato è sempre disponibile la tab* ***Source Code*** che ci consente di analizzare, ed eventualmente copiare, il codice Java relativo.
Esplora le numerose finestre di **SwingSet2, e osserva quanti e** quali altri tipi di controllo sono disponibili in **swing.**

Autore: Prof. Agostino Sorbara
ITIS "M. M. Milano" Polistena (RC)