Le interfacce grafiche in Java - Swing parte 1

Emanuele Ing. Benatti - ebenatti@fermimn.edu.it

Interfacce grafiche

- Tipicamente gli utenti accedono alle applicazioni attraverso un'interfaccia grafica che ne rende più semplice l'utilizzo
  - L'interfaccia grafica contiene componenti grafici come finestre, menù, aree di testo, bottoni
  - L'interazione dell'utente con l'applicazione è mediata dall'interazione dell'utente con l'interfaccia grafica
  - L'utente agisce utilizzando le periferiche del calcolatore ad esempio cliccando col mouse dei bottoni, selezionando voci di menù o operando sulla tastiera
- Un'interfaccia grafica è caratterizzata da due aspetti fondamentali:
  - Definizione del layout grafico
  - Gestione degli eventi

GUI - Graphical User Interface

- L'interfaccia grafica o GUI (Graphical User Interface) è basata su
  - Un certo numero di elementi grafici detti *componenti*
    - (as es. bottoni, aree di testo, menù…)
  - Un certo numero di componenti utilizzati per raggruppare altri componenti e detti *contenitori* (ad es. finestre, pannelli…)
  - I *gestori del layout*, cioè oggetti che regolano il posizionamento dei componenti all'interno dei contenitori
  - I *gestori degli eventi*, cioè oggetti che gestiscono l'interazione tra l'utente e i vari elementi dell'interfaccia grafica

AWT e Swing

- La definizione di interfacce grafiche in Java è supportata da due tecnologie
  - AWT(Abstract Window Toolkit) implementata nel package java.awt
    - È la prima sviluppata presente nel linguaggio sin dall'inizio
  - Swing implementata nel package javax.swing
    - È stata introdotta successivamente ad AWT

GUI con Swing

- Facendo riferimento a Swing, un'interfaccia grafica consiste di:
  - *Un contenitore principale*
    - Ad esempio un applet (finestra di dialogo (`JDialog`)
  - *Un contenitore intermedio*
    - Un pannello (`JPanel`) è l'unico componente contenuto nel contenitore principale, non è visibile ed è usato come contenitore di altri componenti atomici e/o altri pannelli
  - *Vari componenti atomici*
    - Sono gli elementi effettivamente visualizzati nell'interfaccia grafica come bottoni (`JButton`), etichette (`JLabel`), campi di testo (`JTextField`), aree di testo (`JTextArea`)

La classe JFrame

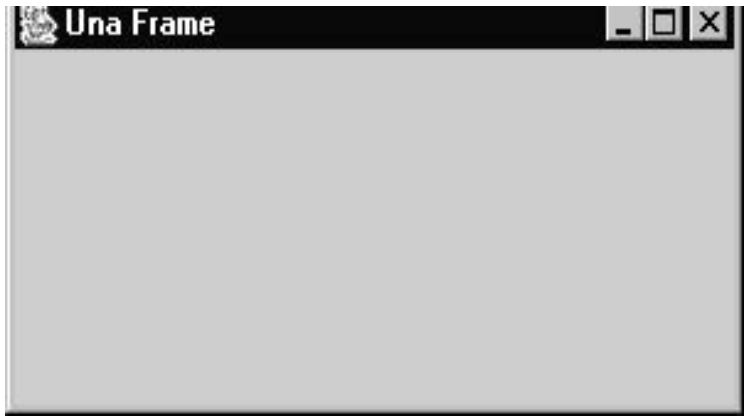
- Il *contenitore principale* di un programma dotato di interfaccia grafica può essere:
  - Un applet (il programma è un applet)
  - Una frame (il programma è un'applicazione)
  - Una finestra di dialogo (non è un programma a se stante)
- Nel caso di applicazioni basate su frame si ha:
  - Una frame è un oggetto istanza che è (polimorficamente) di tipo `Frame` (AWT) o di tipo `JFrame` (Swing)
  - La classe JFrame nel package javax.swing è il progetto di una frame elementare e deve essere estesa per implementare il comportamento desiderato

Java GUI

- Esempio una frame sullo schermo che estende la classe JFrame:

```
import javax.swing.*;
  public class SimpleFrame extends JFrame{  public SimpleFrame(){
      super();
      setTitle ("Una Frame");  setSize (300,150);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      show();
  }

    public static void main(String[] args){  new SimpleFrame();
  }
}
```



Java GUI

- Per realizzare una frame è necessario definire almeno una classe che estende la classe JFrame del package javax.swing
  - Il costruttore della classe serve per inizializzare gli oggetti creati dalla classe
  - La classe è anche la classe da cui si esegue l'applicazione tramite il metodo main() che crea un oggetto SimpleFrame e avvia l'applicazione
- La struttura può essere anche implementata con due classi:
  - Una classe SimpleFrame definisce i costruttori e i metodi della frame personalizzata
  - Un'altra classe ad esempio TestFrame con il metodo main() lancia l'applicazione ed istanzia un oggetto SimpleFrame
- Il costruttore della frame normalmente:
  - Invoca il costruttore super(), come dovrebbero fare tutte le classi estese
  - Assegna alcune proprietà alla frame con vari metodi setXxx()
  - Specifica la chiusura della frame con l'invocazione di di
    - setDefaultCloseOperation()
  - Visualizza la frame con il metodo show()

Java GUI

- Definita la frame occorre inserire i componenti grafici:
  - Ciascun componente è definito da una sottoclasse della classe `JComponent` del package javax.swing
  - Ciascun oggetto componente è un'istanza di una classe che estende `JComponent`
    - Ad esempio i bottoni sono rappresentati dalla classe JButton che estende `JComponent` uno specifico bottone è un'istanza della classe `Jbutton`
- Per utilizzare un componente all'interno di un'interfaccia grafica si deve:
  - Creare il componente (istanziare un oggetto della classe opportuna)
  - Inserire il componente all'interno dell'interfaccia grafica
  - Per gli oggetti swing i componenti non vanno inseriti direttamente nella frame ma vanno inseriti in un contenitore intermedio (il pannello) che a sua volta sarà inserito nella frame
  - Dopo aver inserito tutti i componenti nel contenitore

Esempio di interfaccia “HelloWorld”

```
import javax.swing.;

      public class Es01_HelloWorld {
      public static void main(String[] args) {
      JFrame finestra = new JFrame(Prima finestra);
      JPanel pannello = new JPanel();
      pannello.add( new JLabel(Hello World) );
      finestra.add(pannello);
      finestra.setSize(400, 200);
      finestra.setDefaultCloseOperation(
      JFrame.EXIT_ON_CLOSE);
      finestra.setVisible(true);
      }
}
```

Questo esempio è simile al precedente ma crea gli oggetti dell'interfaccia senza estendere alcuna classe.

JAVA GUI e Interfaccia utente

- Esempio di possibile struttura per costruire un'interfaccia utente:
  - Si definisce una classe che rappresenta l'interfaccia grafica, chiamiamo questa classe UserGUI
    - Tale classe estende JFrame
    - Il costruttore di tale classe inizializza la frame ma delega un altro metodo di supporto initGUI() per l'inizializzazione della parte rimanente dell'interfaccia grafica
    - Il metodo initGUI() crea il pannello, crea i componenti atomici dell'interfaccia, inserisce i componenti nel pannello ed inserisce il pannello nella frame

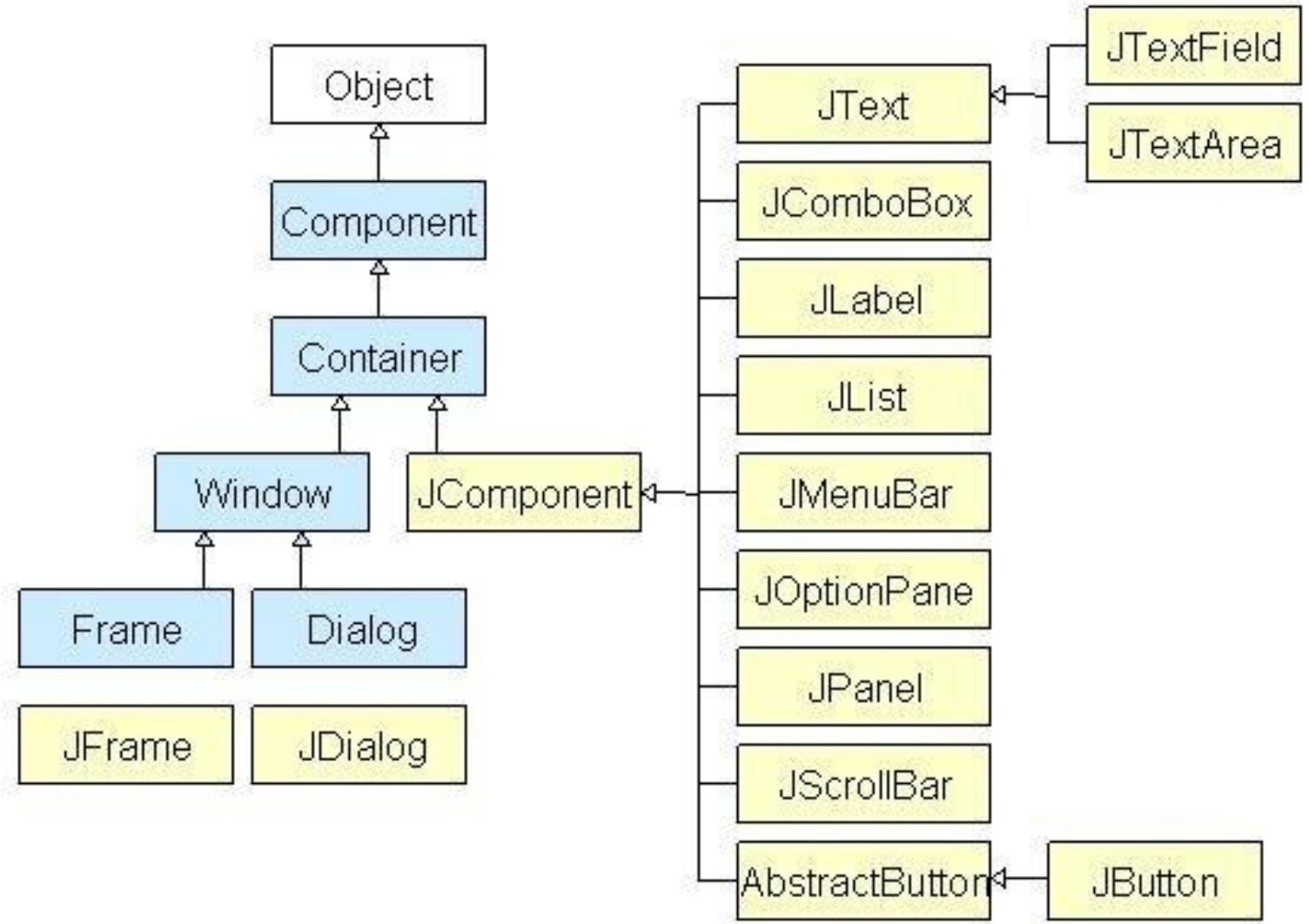
Esempi di interfacce utente

Una frame con un bottone ed una etichetta

Una frame con etichetta e campo di testo

Una frame con un’area di testo

Gerarchia di Classi nella Swing



La classe JButton

- Un oggetto `JButton` rappresenta un bottone cliccabile
- Costruttori:
  - `JButton (String s)` crea un nuovo bottone contenente l'etichetta s
  - `JButton (Icon i)` crea un nuovo bottone contenente l'immagine i
  - `JButton ()` crea un nuovo bottone senza etichetta ne immagine
- Metodi:
  - `void setText(String s)` assegna l'etichetta al bottone
  - `String getText()` accede all'etichetta del bottone
  - `void setIcon(Icon i)` assegna l'immagine al bottone
  - `Icon getIcon()` accede all'immagine del bottone
  - `void setEnabled(boolean b)` abilita (`true`) o disabilita (`false`) il bottone

La classe JLabel

- Un oggetto `JLabel` rappresenta un'area per visualizzare una breve stringa (non modificabile) o un'immagine
- Costruttori:
  - `JLabel (String s)` crea una nuova etichetta contenente il testo s
  - `JLabel (Icon i)` crea una nuova etichetta contenente l'immagine i
  - `JLabel ()` crea una nuova etichetta senza testo ne immagine
- Metodi:
  - `void setText(String s)` assegna il testo all'etichetta
  - `String getText()` accede al testo dell'etichetta
  - `void setIcon(Icon i)` assegna l'immagine all'etichetta
  - `Icon getIcon()` accede all'immagine dell'etichetta

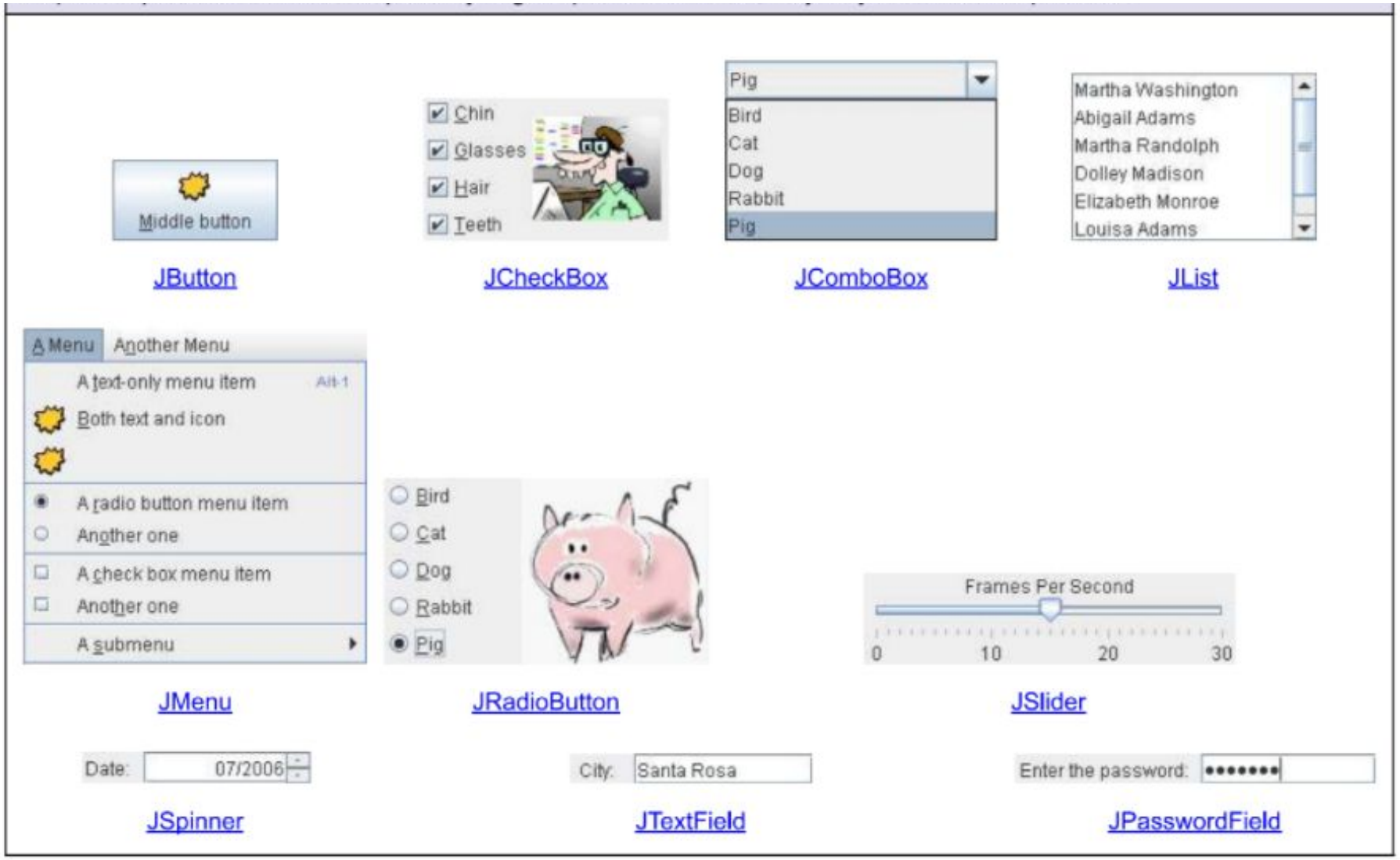
La classe JTextField

- Un oggetto `JTextField` rappresenta un'area per visualizzare una linea di testo editabile
- Costruttori:
  - `JTextField(String s)` crea un nuovo campo di testo contenente il testo s
  - `JTextField (Int i)` crea un nuovo campo di testo di lunghezza i inizialmente vuoto
  - `JTextField ()` crea un nuovo campo di testo inizialmente vuoto
- Metodi:
  - `void setText(String s)` assegna il testo al campo di testo
  - `String getText()` accede al testo del campo di testo
  - `String getSelectedText()` accede alla porzione di testo selezionata nel campo di testo
  - `void setEditable(boolean b)` abilita (`true`) o disabilita (`false`) la possibilità di modificare il campo di testo

La classe JtextArea

- Un oggetto `JTextArea` rappresenta un'area per visualizzare un gruppo di linee di testo editabili
- Costruttori:
  - `JTextArea(String s)` crea una nuova area di testo contenente il testo s
  - `JTextArea(Int r, Int c )` crea una nuova area di testo composta di r righe e c colonne inizialmente vuota
  - `JTextArea()` crea una nuova area di testo inizialmente vuota
- Metodi:
  - `void append(String s)` aggiunge la stringa s alla fine dell'area di testo
  - assegna il testo all'area di testo
  - `String getText()` accede al testo dell'area di testo
  - `String getSelectedText()` accede alla porzione di testo selezionata nell'area di testo
  - `void setEditable(boolean b)` abilita (`true`) o disabilita

Componenti

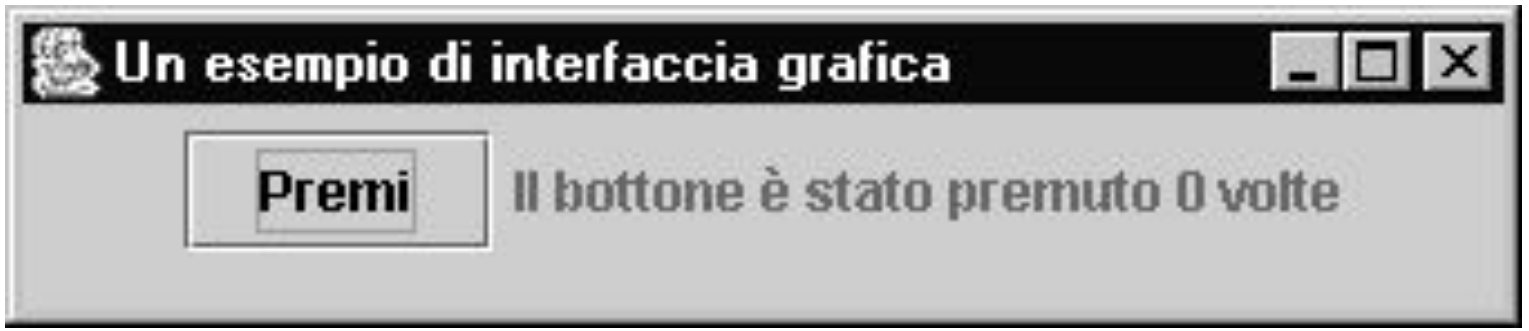


Altri componenti sono disponibili nella documentazione ufficiale di Oracle!

Interfaccia ed eventi

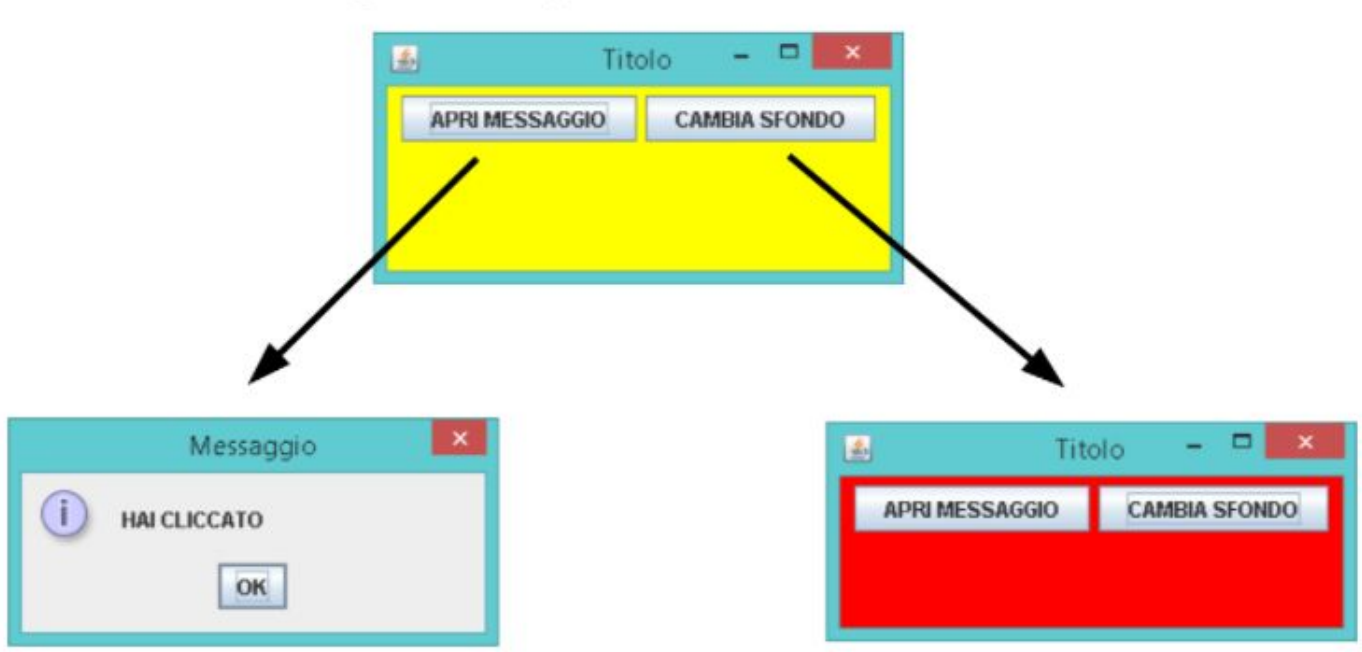
- L’interfaccia grafica di un’applicazione ha lo scopo di
  - Mostrare informazioni all’utente
  - Consentire all’utente di eseguire operazioni
- Ad es. in questo caso si ha un bottone da premere ed un'etichetta informativa
- Il bottone è l’elemento dell’interfaccia con cui l’utente può interagire

- L’etichetta ha lo scopo di mostrare quante volte il bottone è stato premuto
- Ciascuna interazione dell’utente con l’interfaccia grafica deve essere notificata all’applicazione in modo da eseguire le operazioni richieste
- Un evento è la notifica di un’interazione tra l’utente e l’interfaccia grafica



Esempio di Interfaccia ed eventi

Vogliamo realizzare la seguente "applicazione":



Il codice che segue mostra come aggiungere un ascoltatore (listener) ad un pulsante. Lo stesso codice si può applicare a molti altri componenti grafici.

- Nel contesto delle GUI, un ascoltatore è un oggetto che ha il compito di reagire agli eventi generati da un componente grafico (click, selezioni, pressione di tasti, ecc) .

Esempio di Interfaccia ed eventi

```
JPanel pannello = new JPanel();
pannello.setBackground(Color.yellow);

JButton pulsante1 = new JButton("APRI MESSAGGIO");
pulsante1.addActionListener( new MyButtonListener() );
pannello.add(pulsante1);

JButton pulsante2 = new JButton("CAMBIA SFONDO");
pulsante2.addActionListener( new MyButtonListener2(pannello) );
pannello.add(pulsante2);

JFrame f = new JFrame("Titolo");
f.setSize(300, 300);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setContentPane(pannello);
f.setVisible(true);
```

Esempio di Interfaccia ed eventi (2)

```
class MyButtonListener implements ActionListener {
public void actionPerformed(ActionEvent e) {
JOptionPane.showMessageDialog(null, "HAI CLICCATO");
 }

}
```

```
class MyButtonListener2 implements ActionListener {
JPanel pannello;
public MyButtonListener2(JPanel p) {
   this.pannello = p;
}
public void actionPerformed(ActionEvent e) {
if ( pannello.getBackground().equals(Color.red) )
        pannello.setBackground(Color.yellow);
   else
        pannello.setBackground(Color.red);
   }
}
```

L'interfaccia ActionListener richiede di implementare un unico metodo: actionPerformed
Quando un pulsante è premuto,
Swing richiama il metodo actionPerformed dei listener che sono stati aggiunti al pulsante tramite addActionListener(...)

ActionListener è figlia di EventListener, interfaccia generica di un evento.

Gli eventi e la GUI - Gli ascoltatori

- Quando si verifica un evento, questo viene gestito utilizzando tre oggetti:
- Un oggetto che rappresenta l'evento che si è verificato
  - Gli eventi sono rappresentati da oggetti istanza della class **`EventObject`** e delle sue sottoclassi (ActionEvent e MouseEvent)
- Il componente dell'interfaccia grafica che ha causato l'evento, detto la sorgente dell'evento
  - Ad esempio un bottone o un campo di testo
- Un oggetto che deve gestire l'evento, **chiamato gestore dell'evento o ascoltatore (listener) dell'evento**
  - I gestori di eventi sono rappresentati da oggetti istanza di classi che implementano l'interfaccia **`EventListener e dalle interfacce derivate ActionListener oppure MouseListener`**
  - Un'interfaccia grafica può definire uno o più ascoltatori ciascun ascoltatore è dedicato alla gestione di una o più tipologie di eventi che si possono verificare nell'interfaccia grafica

Gli eventi e la GUI(2)

- In pratica per gestire gli eventi di un'interfaccia grafica è necessario:
  - Definire delle classi per implementare ascoltatori per gli eventi che si intende gestire
    - Un ascoltatore definisce un metodo per ciascun evento che può verificarsi, e in tale metodo sono implementate le azioni che devono essere eseguite in corrispondenza di tale evento
    - Una stessa classe può implementare più ascoltatori per tipi di eventi diversi (una classe può implementare più interfacce)
  - Istanziare gli oggetti ascoltatori necessari
  - Registrare ciascuna sorgente di eventi presso l'ascoltatore corrispondente
    - La registrazione di una sorgente presso un ascoltatore rende l'ascoltatore responsabile della gestione degli eventi generati da quella sorgente
    - Gli eventi generati da una sorgente di eventi non registrata non vengono gestiti , cioè sono ignorati

Interfaccia ActionListener

```
public interface ActionListener
{

    void actionPerformed(ActionEvent e); //Invoked when an
    action occurs.

}
```

L’interfaccia MouseListener

- Interfaccia specializzata nella gestione del mouse e permette di creare eventi specifici quando il mouse svolge determinate azioni.
  - `void mouseClicked (MouseEvent i)`
    - Viene invocato quando il mouse viene cliccato in un componente passandogli come argomento l’oggetto evento corrispondente
  - `void mouseEntered (MouseEvent i)`
    - Viene invocato quando il mouse entra in un componente
  - `void mouseExited (MouseEvent i)`
    - Viene invocato quando il mouse esce da un componente
  - `void mousePressed (MouseEvent i)`
    - Viene invocato quando un tasto del mouse viene premuto in un componente
  - `void mouseReleased (MouseEvent i)`
    - Viene invocato quando un tasto del mouse viene rilasciato in un componente

Gli eventi e la GUI (3)

- Obiettivo: realizzare un "ascoltatore di eventi" basato sull'implementazione di un'interfaccia: si vuole realizzare una frame dove un'etichetta si modifica quando il mouse passa sopra ad un bottone.
  - Viene definita una classe che implementa la frame
  - Gli eventi significativi sono l'ingresso e l'uscita del puntatore del mouse nella regione occupata dal bottone
  - La sorgente di eventi è il bottone
- Eventi del tipo descritto sono di competenza dell'interfaccia

- `MouseListener` (`package.awt.event`)
  - La classe che implementa la frame viene definita in modo tale da implementare l'interfaccia `MouseListener`: in questo modo la frame sarà l"ascoltatore" degli eventi di un suo componente (il bottone)

La gestione degli eventi e gli adattatori

- È possibile implementare "l'ascoltatore" definendo una classe che implementa **tutta** l'interfaccia `MouseListener`. Si utilizzerà in realtà un'implementazione più semplice basata sull'uso di un **"adattatore". Un adattatore è una classe che implementa tutti i metodi dell'interfaccia, eventualmente senza corpo.**

- Avendo a disposizione un adattatore C per un'interfaccia I è possibile implementare l'interfaccia I definendo una nuova classe che estende C
- Questa soluzione è vantaggiosa se si vuole implementare l'interfaccia I in modo tale che solo alcuni dei suoi metodi vengano definiti in modo non banale.

In alternativa bisogna usare la classe che implementa l'interfaccia MouseInputAdapter (package javax.swing.event)

Tipologie di eventi

- Ciascun componente può generare diverse tipologie di eventi. Ad es. un bottone può generare:
  - Un evento di tipo "**azione**" (evento di tipo `ActionEvent` da `java.awt.event`) se viene premuto mediante l'uso del mouse
  - Un evento di tipo "**mouse**" (evento di tipo `MouseEvent` da `java.awt.event`) se il puntatore del mouse viene posizionato nell'area del bottone
- Ciascun ascoltatore può gestire diverse tipologie di eventi. Ad es.un ascoltatore può ascoltare e gestire:
  - Eventi di tipo `ActionEvent` se implementa l'interfaccia `ActionListener`
  - Eventi di tipo `MouseEvent` se implementa l'interfaccia `MouseListener`
  - Eventi di entrambi i tipi se implementa entrambe le interfacce

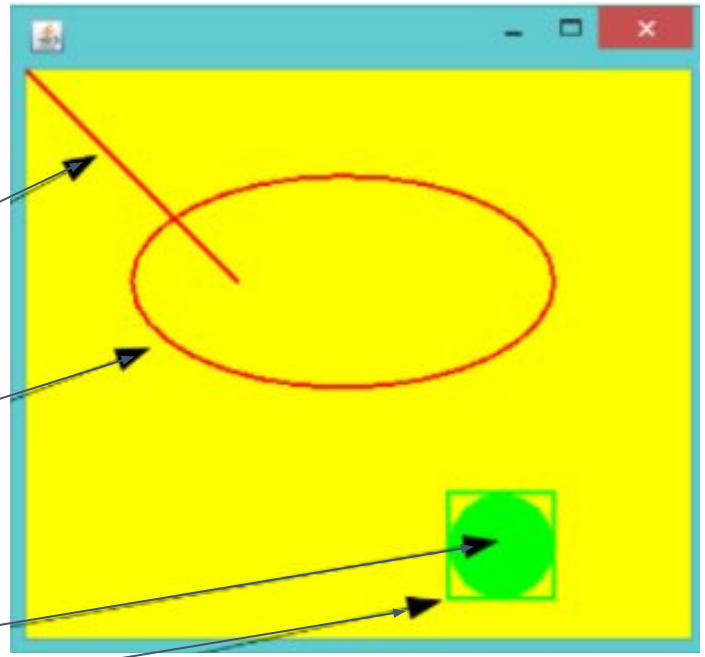
Button, ActionEvent, e ActionListener

- Un bottone (componente di tipo `JButton`) quando viene premuto genera un evento di tipo `ActionEvent`
  - Un evento di tipo `ActionEvent` può essere gestito da un oggetto che implementa l'interfaccia `ActionListener`
  - L'interfaccia `ActionListener` contiene solo la dichiarazione del metodo `void actionPerformed (ActionEvent e)`
- In pratica per gestire l'evento "un bottone è stato premuto" si deve:
  - Creare un oggetto ascoltatore che implementa l'interfaccia `ActionListener` (ad es. creando un oggetto da una "classe interna anonima" che implementa l'interfaccia `ActionListener`)
  - Registrare il bottone presso tale oggetto ascoltatore

Disegnare all'interno di un pannello

E' possibile creare grafica personalizzata creando una classe che estende JPanel e ridefinisce il suo metodo paint( )

```
class MyPanel extends JPanel {
public MyPanel() {
setBackground(Color.yellow);
}
public void paint(Graphics g) {
        super.paint(g);
        Graphics2D g2 = (Graphics2D) g;
        g2.setStroke(new BasicStroke(2));
        g2.setColor(Color.red);
        g2.drawLine(0, 0, 100, 100);
        g2.drawOval(50, 50, 200, 100);
        g2.setColor(Color.green);
        g2.fillOval(200, 200, 50, 50);
        g2.drawRect(200, 200, 50, 50);
        }
}
```



Spiegazione dell’esempio

- Ogni componente grafico in Swing possiede un metodo paint che si occupa di "disegnare l'oggetto stesso" nella finestra
- Un JPanel è un'area "vuota". Ridefinendo paint è possibile disegnare all'interno di quest'area.
- paint riceve come argomento un oggetto di tipo Graphcs.
- L'oggetto Graphics consente di disegnare all'interno dell'area di schermo occupata dal componente.
- Graphics espone metodi per tracciare figure vuote (drawRect, drawOval, ...) o piene (fillRect, fillOval, ...), immagini, testo, rette, archi, ecc.
- Le coordinate sono relative all'angolo in alto a sinistra del componente (non dell'intera finestra o del monitor).
- E' possibile cambiare il colore del tratto e lo spessore del tratto (Stroke)

Usare i gestori dei layout

- I gestori di layout sono associati ai contenitori e gestiscono l'inserimento dei componenti nel contenitore
  - In particolare è possibile associare un layout manager al pannello usato come contenitore intermedio nell'interfaccia grafica
  - L'associazione di un layout manager ad un contenitore può avvenire:
    - Al momento della creazione del contenitore specificando il layout manager come argomento del costruttore del contenitore
      - `JPanel jpanel=new JPanel (new BorderLayout());`
    - Aggiungendo il layout manager al contenitore dopo la creazione del contenitore
      - `JPanel jpanel=new JPanel ();`
      - `Jpanel.setLayout (new BorderLayout());`

Gestore dei layout

- Per inserire un componente in un contenitore si usa il metodo `add(Component c)`
  - Nell'invocare questo metodo viene specificato il componente da inserire nel contenitore ma non viene specificata la posizione che esso dovrà occupare
  - È possibile specificare la posizione in cui inserire il componente nel contenitore ma solo in modo indiretto ossia in modo relativo (ad es. dicendo che il componente deve andare in alto o a destra) e non in modo assoluto (specificandone le coordinate)
  - La possibilità di collocare un componente in un contenitore è gestita mediante oggetti associati ai contenitori chiamati "gestori di layout"
  - I "gestori di layout" sono rappresentati da sotto-classi della classe

- `LayoutManager (package java.awt)`:
  - `FlowLayout` (i componenti sono disposti come il testo in una pagina su linee che vanno da sx a dx
  - `GridLayout` (i componenti sono disposti su una griglia rettangolare)
  - `BorderLayout` (il contenitore è organizzato in 5 aree: centro, alto, basso, sinistra, destra. Ciascun componente va esplicitamente collocato in una di queste 5 aree)

FlowLayout

È il gestore di layout di default per un pannello e dispone i componenti come un testo in una pagina su linee che vanno da sx a dx

Esempio: frame con 6 bottoni disposti usando `FlowLayout`

- Effetti del ridimensionamento della frame

GridLayout

- Dispone i componenti su una griglia rettangolare, il numero di righe e colonne che costituiscono la griglia va specificato al momento della creazione
- I componenti vanno inseriti per righe da sx a dx
- Es: frame con disposizione su una griglia 3x2

- Es: frame con disposizione su una griglia 2x3

BorderLayout

- Dispone i componenti in 5 regioni identificate dalle costanti
  - `BorderLayout.NORTH`
  - `BorderLayout.SOUTH`
  - `BorderLayout.EAST`
  - `BorderLayout.WEST`
  - `BorderLayout.CENTER`
- La disposizione di un componente in una di queste 5 aree va specificata utilizzando una delle 5 costanti come secondo argomento del metodo `add`
- Se più componenti vengono collocati nella stessa regione possono sovrapporsi