Università di Pisa

FACOLTA' DI INGEGNERIA

Corso di Laurea in Ingegneria Informatica per la Gestione d'Azienda

TESI DI LAUREA

Progetto e Realizzazione di una Libreria Grafica per Applicazioni Didattiche

Relatori
Prof. Andrea Domenici
Prof.ssa Cinzia Bernardeschi

Candidato
Giovanni Alestra

Anno Accademico 2011/2012

Indice

Capitolo 1

Specifiche e Scelte di Progetto

1.1 Introduzione

L'argomento della seguente tesi è il progetto e la realizzazione di una libreria grafica per applicazioni didattiche. Il tema didattico è stato di fondamentale importanza nella fase delle scelte progettuali e del progetto. In particolare una libreria pensata per una applicazione didattica deve essere studiata, progettata e realizzata in modo da presentare l'argomento didattico in molte o quasi tutte le sue sfaccettature e proporre in chiave didattica i concetti dell'argomento. Sebbene qualsiasi argomento possa essere spiegato solamente in via teorica, la realizzazione di questa libreria, la sua implementazione, consente di applicare praticamente i concetti teorici acquisiti, e di ottenere una maggiore comprensione di questi concetti che deriva da uno sviluppo pratico degli stessi che si materializza nella realizzazione di software funzionante. Questo è un esercizio pratico che si può svolgere in lezioni fatte in laboratorio con l'ausilio del calcolatore.

L'argomento didattico trattato sono le interfacce grafiche per applicazioni desktop altrimenti dette GUI. L'argomento è caratterizzato dalla necessità di presentare i principali componenti grafici che

caratterizzano la maggior parte delle interfacce grafiche quindi la creazione di finestre e l'aggiunta a queste dei componenti grafici. Un altro aspetto fondamentale è la gestione degli eventi che permette la gestione dell'interazione dell'utente con l'interfaccia grafica e quindi l'esecuzione di statement che rappresentano la logica dell'applicazione ovvero quello che l'applicazione permette di fare a livello di elaborazione dati. In particolare si vuole mostrare che una interfaccia grafica si crea mediante la creazione di finestre e l'aggiunta a queste dei componenti grafici e si gestisce a livello di elaborazione dati mediante la gestione degli eventi che deve essere implementata. La libreria permette quindi di realizzare in via di esercizio interfacce grafiche, consente di fare delle elaborazioni dei dati immessi dall'utente e presentare sempre in via grafica i risultati di tali elaborazioni.

1.2 Specifiche

Le specifiche da cui si è partiti prima di effettuare le scelte del progetto, sono state dettate dall'esigenza di far utilizzare un linguaggio di programmazione che consenta la portabilità delle applicazioni realizzate tra i diversi sistemi operativi. Il linguaggio di programmazione scelto deve essere inoltre di interesse per la didattica e se non conosciuto dagli studenti, vicino ai linguaggi di programmazione da essi conosciuti quindi C++-Like. Deve essere inoltre di comune utilizzo anche in ambito professionale. Un'altra specifica rappresenta la scelta del package grafico su cui appoggiarsi, che deve essere utilizzato in ambito professionale, moderno e non deprecato.
Tra le specifiche troviamo anche la necessità di semplificare l'argomento trattato realizzando una libreria che sia il più possibile intuitiva nel suo utilizzo. La libreria deve poter essere introdotta e spiegata in poco tempo ed in particolare, deve essere ben documentata tanto da poter essere studiata e compresa anche senza l'ausilio del docente. Deve essere dunque corredata da un manuale d'uso chiaro e semplice, da documentazioni aggiuntive e da qualche esempio pratico.
Un'altra specifica riguarda la gestione delle eccezioni che deve essere implementata, in modo da ottenere una libreria in grado di eliminare le possibilità di errato utilizzo ed indirizzare gli utilizzatori verso un consapevole e corretto utilizzo dello strumento.

1.3 Scelte di Progetto

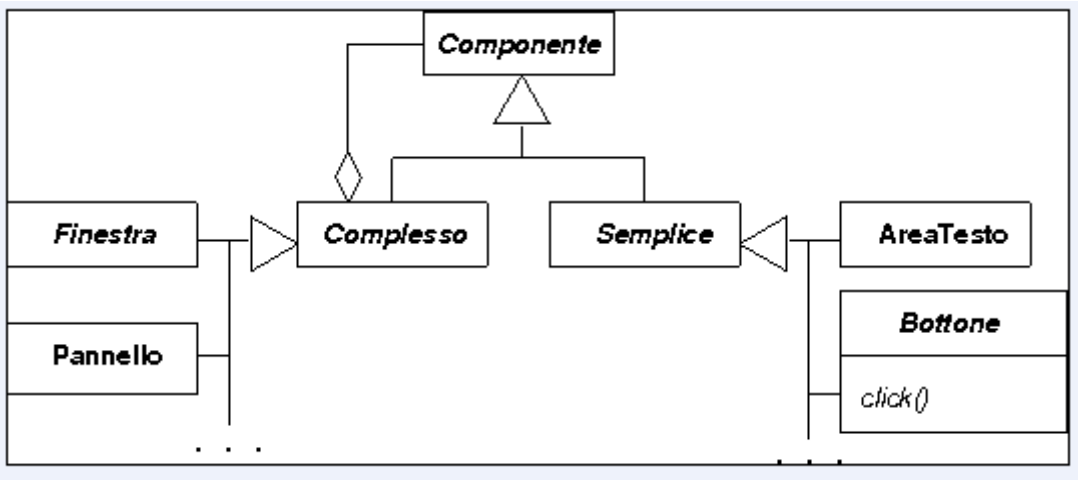
Siamo stati portati a scegliere il linguaggio **Java** perchè consente la portabilità tra i diversi sistemi operativi. Inoltre è un linguaggio ad oggetti utilizzatissimo nella pratica professionale e C++Like. Il package grafico su cui appoggiarsi scelto è **Swing**, perchè tra quelli utilizzabili nel linguaggio Java è moderno e non deprecato. La necessità di semplificare l'argomento è stata di centrale interesse nella fase di progetto e tutte le semplificazioni sono spiegate ed illustrate nel seguito di questo documento. Si è inoltre deciso di creare un manuale d'uso della libreria, e delle demo implementate. Come documentazione aggiuntiva è stato scelto il JavaDoc, strumento Java semplice ma potente in grado di fornire una documentazione su tutte le classi e sui metodi che la libreria mette a disposizione. E' stata inoltre prevista una gestione delle principali e possibili eccezioni attraverso la gestione delle eccezioni che Java mette a disposizione.
La libreria viene ad essere costituita da un unico file Jar (java-archive) ed utilizzabile semplicemente attraverso l'operazione di importazione : *import swinggl.**.

Capitolo 2

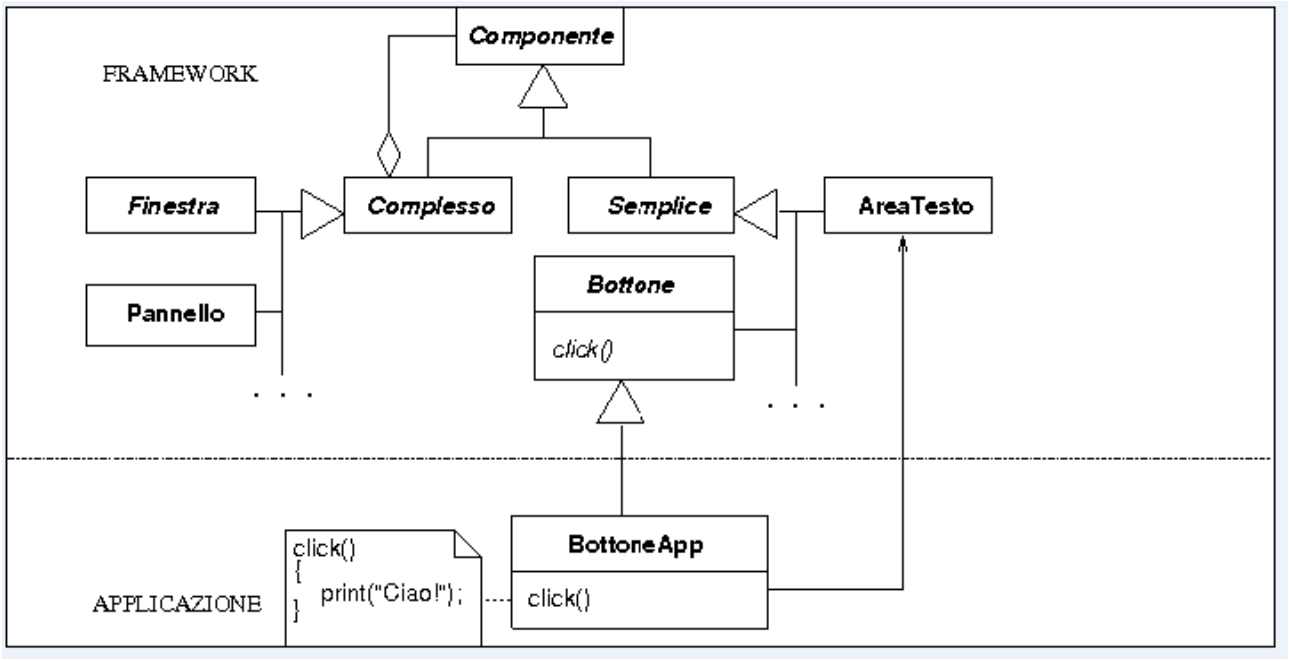
Il Package Grafico Swing

2.1 Concetto di Framework

Un framework offre uno schema di soluzione preconfezionato per un determinato tipo di problema e viene utilizzato in un'applicazione particolare specializzandone alcune caratteristiche. Questa specializzazione avviene generalmente attraverso il polimorfismo : il framework offre al progettista delle classi e delle interfacce in cui comportamento viene specializzato derivando altre classi in cui si possono ridefinire i metodi delle classi e interfacce originarie ed aggiungere ulteriori funzioni.
La seguente figura mostra uno schema molto semplificato di un framework:



Questo framework offre un insieme di elementi grafici ("Componente"), classificati in componenti semplici ("Semplice") e complessi ("Complesso"). Questi ultimi sono composti da altri componenti. Alcuni tipi di componenti semplici sono le aree testo ("AreaTesto") ed i bottoni ("Bottone"). Molte classi sono astratte perche' parzialmente implementate, e questo permette all'utente del framework di personalizzarle in base alle esigenze dell'applicazione da sviluppare, come mostra la figura seguente:



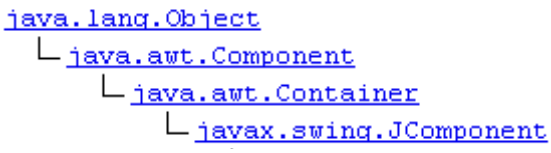
L'utente ha creato la classe BottoneApp, derivata da Bottone, implementando il metodo click().

Quest'ultimo verra` chiamato dal framework quando viene premuto il tasto del mouse.

Nello sviluppo della libreria sono state ad esempio estese le classi di swing e sfruttato le potenzialità del framework nel suo complesso.

2.2 Gerarchia delle Classi di Swing

Prima di utilizzare il package grafico Swing è stato necessario studiarne la struttura ed i componenti grafici che gli sono propri. In particolare riportiamo la gerarchia delle classi di Swing al fine di capire come sia possibile sfruttare tale gerarchia. La gerarchia delle classi di Swing è la seguente :



Nella struttura si evidenzia la differenza tra un componente Jcomponent ed un Container. In particolare è stato necessario comprendere che ogni componente grafico che si vuole aggiungere ad una interfaccia grafica deve essere aggiunto ad un container (contenitore) mediante il metodo :

java.awt.Container.Add (Component comp)

Riportiamo il JavaDoc della funzione Add.

add

public `Component` **add**(`Component` comp)

> Appends the specified component to the end of this container. This is a convenience method for addImpl(java.awt.Component, java.lang.Object, int).
>
> Note: If a component has been added to a container that has been displayed, validate must be called on that container to display the new component. If multiple components are being added, you can improve efficiency by calling validate only once, after all the components have been added.

Parameters:
comp - the component to be added
Returns:
the component argument

Mentre notiamo che i componenti grafici di Swing vengono ad estendere la classe javax.swing.JComponent. Quindi possiamo affermare che nella costruzione di una interfaccia grafica per aggiungere un componente è necessario dichiarare il contenitore nel quale aggiungerlo e richiamare la funzione Add. Quindi dato un oggetto contenitore ed un componente possiamo sempre effettuare questa operazione :

((java.awt.Container) container).Add ((javax.swing.JComponent) component);

Questo statement è stato inserito in ogni metodo che prevedeva l'aggiunta di un componente grafico.

2.3 Principali Componenti Grafici di Swing

Analizziamo adesso i principali componenti grafici che Swing mette a disposizione. Sono tutte classi che estendono javax.swing.JComponent.

1) Main Windows
   Possiamo considerare le finestre il primo componente grafico di Swing. Una finestra è un componente della classe **JFrame**.
2) Panels
   I Pannelli sono dei componenti che vengono aggiunti alle finestre ed ai frames per l'aggiunta di componenti grafici. Un Panel è un componente della classe **JPanel**
3) Labels
   Le etichette (Labels) sono delle righe di testo che si possono visualizzare. Le etichette sono un componente della classe **Jlabel**.
4) Text Fields
   I Text Fields sono campi nei quali l'utente può scrivere del testo. Un text field è un componente della classe **JTextField**
5) Text Areas

I Text Areas sono simili alle Text Fields ma consentono di inserire il testo su più di una riga. E' un componente della classe **JTextArea**

6) Password Fields

I campi password sono componenti della classe **JPasswordField**

7) Combo Boxes

I Combo Boxes sono campi nei quali si può selezionare un testo predefinito dal programmatore. Il ComboBox è un componente della classe **JcomboBox**

8) Check Boxes

I Check Boxes sono campi nei quali si può selezionare o deselezionare un valore. Il CheckBox è un componente della classe **JCheckBox**

9) Radio Buttons

Un radio button permette la selezione di una opzione tra diverse opzioni. Il Radio Button è un componente della classe **JRadioButton**

10) Tabbed Pane

Un Tabbed Pane consente di inserire diversi pannelli nello stesso componente e di selezionarli in base al loro titolo. Un Tabbed Pane è un componente della classe **JTabbedPane**

11) Buttons

Un Button è un bottone che si può cliccare. In generale per questo elemento deve essere prevista la gestione dell'evento Click e l'esecuzione di una elaborazione dati. Un Button è un componente della classe **JButton**

12) Menu Bars

Una Menù Bar è la barra dei menù che può essere aggiunta ad una finestra. E' un componente della classe **JMenuBar**

13) Menù Items

I Menù Items sono i veri e propri menù a tendina che devono essere aggiunti ad una menù bar. Deve essere anche prevista la gestione dell'evento click. E' un componente della classe **JMenuItem**

14) Desktop Panes

I desktop panes sono alternatici ai pannelli e si aggiungono ad una finestra qual'ora all'interno di questa vogliono essere visualizzate altre finestre (InternalFrames). E' un componente della classe **DesktopPane**

15) Internal Frames

Gli internal frames sono delle finestre che vengono visualizzate all'interno di una finestra

principale tipicamente come documento. E' un componente della classe **Jinternal Frames**

16) Open/Save File Dialog

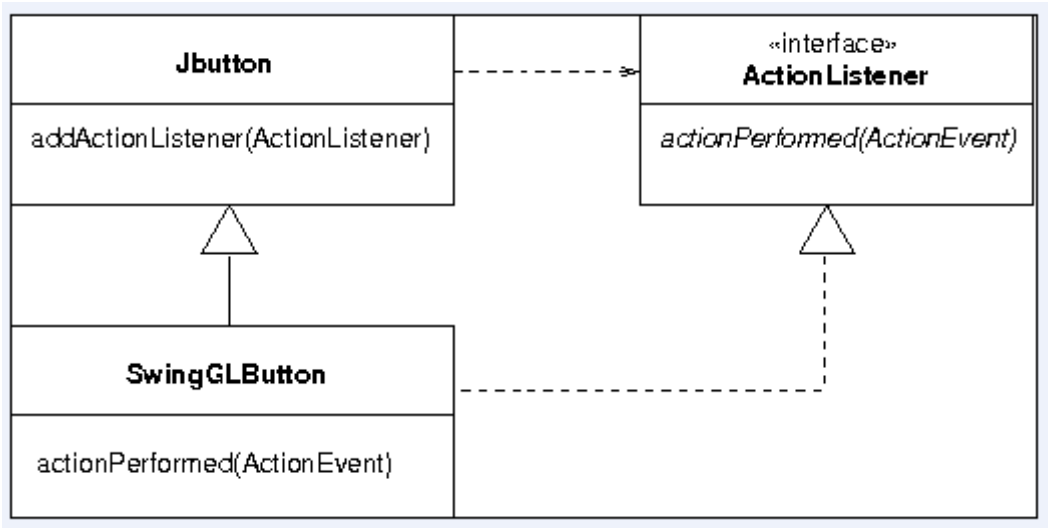
Sono finestre predefinite nella quali si può visualizzare il file system e scegliere un file da aprire o dichiarare il nome di un file su cui salvare. Sono componenti della classe **JfileChooser**

17) Message ed Input Dialogs

Sono delle Dialogs attraverso le quali si può visualizzare un messaggio o fare una domanda all'utente. Sono componenti della classe **JOptionPane**

2.4 Eventi in Swing

Un evento come ad esempio l'evento Click di un bottone può essere gestito in Swing, attraverso il meccanismo dei Listener. Esistono vari tipi di Listener ma i più comuni sono i Listener di Azioni (ActionListener). Un oggetto che è soggetto ad un evento deve aggiungere, mediante il metodo **addActionListener**, il gestore dell'evento, che altro non è che una classe che implementa l'interfaccia **ActionListener**. L'interfaccia **ActionListener** per essere implementata prevede l'implementazione del metodo : ***public void actionPerformed(ActionEvent e).*** All'interno di questo metodo è possibile inserire gli statement che rappresentano la gestione dell'evento. La figura seguente mostra, per esempio, come questo schema sia stato applicato nella classe SwingGLButton:



Riassumeremo quanto detto e faremo un esempio al fine di comprendere meglio questi concetti. Riassumendo possiamo dire che per gestire un evento dobbiamo dichiarare una classe che implementa l'interfaccia **ActionListener**, ed aggiungere questa classe ai Listener dell'oggetto mediante il metodo **AddActionListener**. Si mostrerà di seguito l'implementazione di una semplice interfaccia grafica costituita da una finestra e da un bottone che se cliccato fa comparire una stringa sul prompt dei comandi.

2.4.1 Esempio della Gestione di un Evento in Swing

Propongo questo esempio semplicissimo che ci consentirà di capire meglio come si gestiscono gli eventi in Swing. Iniziamo con la classe che implementa l'interfaccia **ActionListener**.

```
package simpleevent;

/**
 *
 * @author Giovanni Alestra
 */

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

public class ButtonClick implements ActionListener {

    public void actionPerformed ( ActionEvent e ) {
        System.out.println ("Action");
    }
}
```

La classe ButtonClick implementa l'interfaccia ActionListener e di conseguenza la funzione actionPerformed per la gestione di un evento. La gestione dell'evento prevede solamente la stampa su prompt dei comandi della stringa “Action”.
Passiamo alla classe principale nella quale vengono ad essere inseriti gli statement per la creazione della semplice interfaccia grafica.

```
package simpleevent;

/**
```

```
 * @author Giovanni Alestra
 */

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

public class SimpleEvent {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      JFrame window = createWindow ("Button Click Example");
         // Richiamo della funzione per la creazione della finestra
      JPanel panel = addPanel (window);
         // Richiamo della funzione per l'aggiunta di un pannello alla finestra
      JButton button = addButton ("OK", panel);
         // Richiamo della funzione per l'aggiunta del bottone al pannello
      ButtonClick Action = new ButtonClick();
         // Istanziazione dell'oggetto gestore dell'evento click
      addAction ( button, Action);
         // Aggiunta dell'oggetto gestore dell'evento click all'actionListener del bottone

   }
```

La funzione della classe principale prevede la creazione di una finestra, l'aggiunta a questo di un bottone cliccabile e l'aggiunta della classe gestore dell'evento all'actionListener del bottone.
Mostriamo adesso la funzione che permette di aggiungere una finestra.

```
   private static JFrame createWindow (String title) {
      JFrame frame = new JFrame (title);
         \\ Creazione della finestra a cui viene assegnato un titolo (Richiamando il costrutture della classe JFrame
      frame.setLayout(null);
         \\ Settaggio del Layout del Frame a null che ci consente di posizionare gli oggetti in posizione assoluta (x,y)
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         \\ Settaggio della operazione di default relativa alla chiusura della finestra
      frame.setBounds(100, 100, 300, 300);
         \\ Posizionamento e Dimensionamento della finestra
      frame.setVisible(true);
         \\ Rende visibile la finestra

      return frame;
   }
```

Mostriamo e commentiamo adesso la funzione per l'aggiunta del pannello alla finestra

```
   private static JPanel addPanel (JFrame frame) {
```

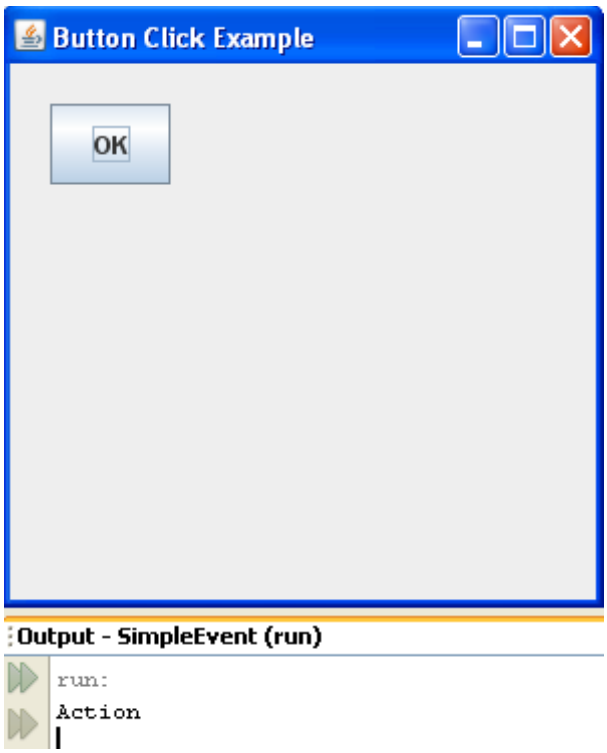
```
        JPanel panel = new JPanel();
           // Creazione del pannello
        panel.setLayout(null);
           // Settaggio del Layout
        panel.setBounds (0,0,100,100);
           // Posizionamento e dimensionamento del componente
        panel.setVisible(true);
           // Rende visibile il pannello
        frame.add(panel);
           // Aggiunta del pannello al contenitore finestra
        return panel;
    }
```

Mostriamo e commentiamo adesso la funzione per l'aggiunta del bottone al pannello.

```
    private static JButton addButton (String text, JPanel panel) {
        JButton button = new JButton (text);
           //  Creaiamo un bottone passando al costrutture del bottone il testo che compare sul bottone
stesso
        button.setBounds ( 20,20,60,40);
           // Posizionamento del bottone relativo al contenitore e suo dimensionamento
        button.setVisible(true);
           // Rende visibile il bottone
        panel.add(button);
           // Aggiunta del bottone al pannello contenitore
        return button;
    }
```

Mostriamo adesso l'aggiunta della classe gestore dell'evento click all'actionListener del bottone

```
    private static void addAction ( JButton button, ActionListener action ) {
        button.addActionListener(action);
           // Aggiunta del gestore dell'evento all'actionListener

    }
}
```


Button Click Example
OK
Output - SimpleEvent (run)
run:
Action

Capitolo 3

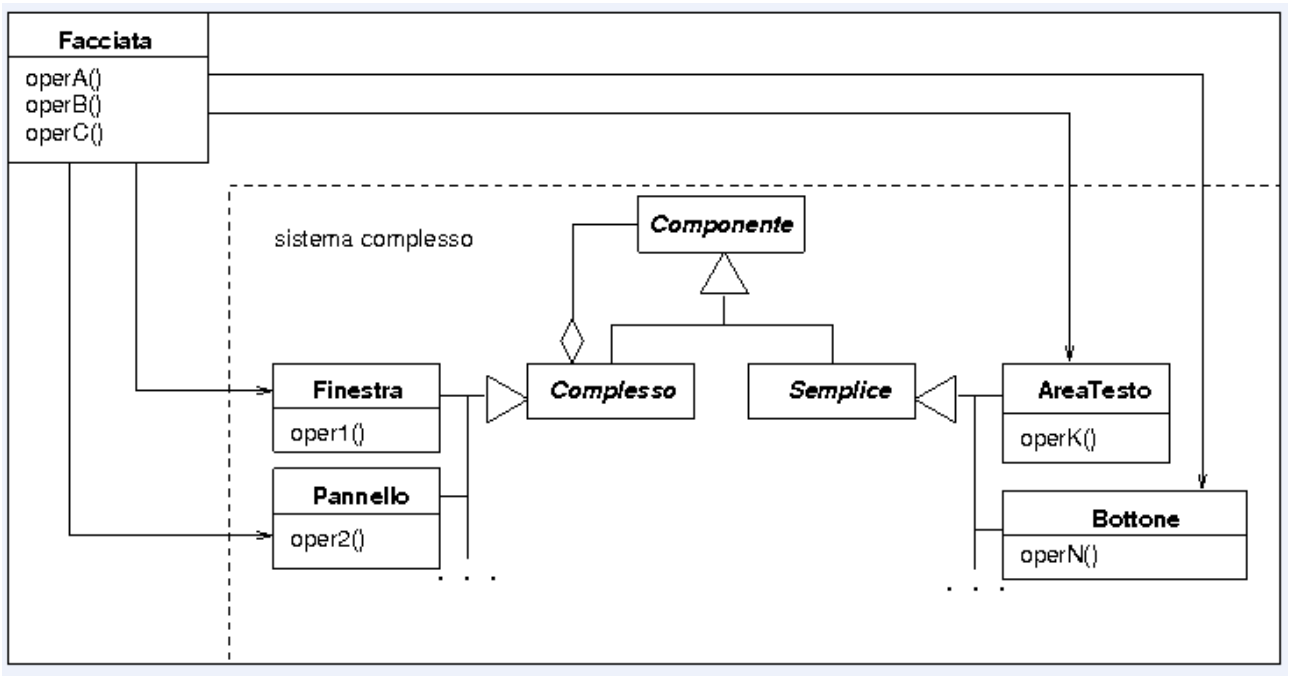
Il Progetto

3.1 Distribuzione ed Utilizzo della Libreria

La libreria che si vuole realizzare è formata da un unico package di classi. Essa può essere distribuita sotto forma di file Jar ed utilizzata in qualsiasi applicazione attraverso l'operazione di importazione del package dopo avere aggiunto il file Jar tra le librerie. Il nome dato alla libreria è **SwingGL** che sottolinea il suo basarsi su Swing. Il nome del file Jar è swinggl.jar. Il package si chiama anchesso **swinggl** e l'operazione di importazione del package è ***import swinggl.*.***

3.2 Il Design Pattern "Façade"

La principale scelta di progetto nella realizzazione della libreria SwingGL è stata l'applicazione del design pattern Façade. Lo scopo di questo schema architetturale è di fornire un'interfaccia semplificata ad un sistema complesso, quale, nel caso presente, un framework per interfacce grafiche. Secondo questo schema, i clienti del sistema complesso vi accedono attraverso un'unica classe che funge da "facciata" del sistema stesso, come mostra la figura seguente:



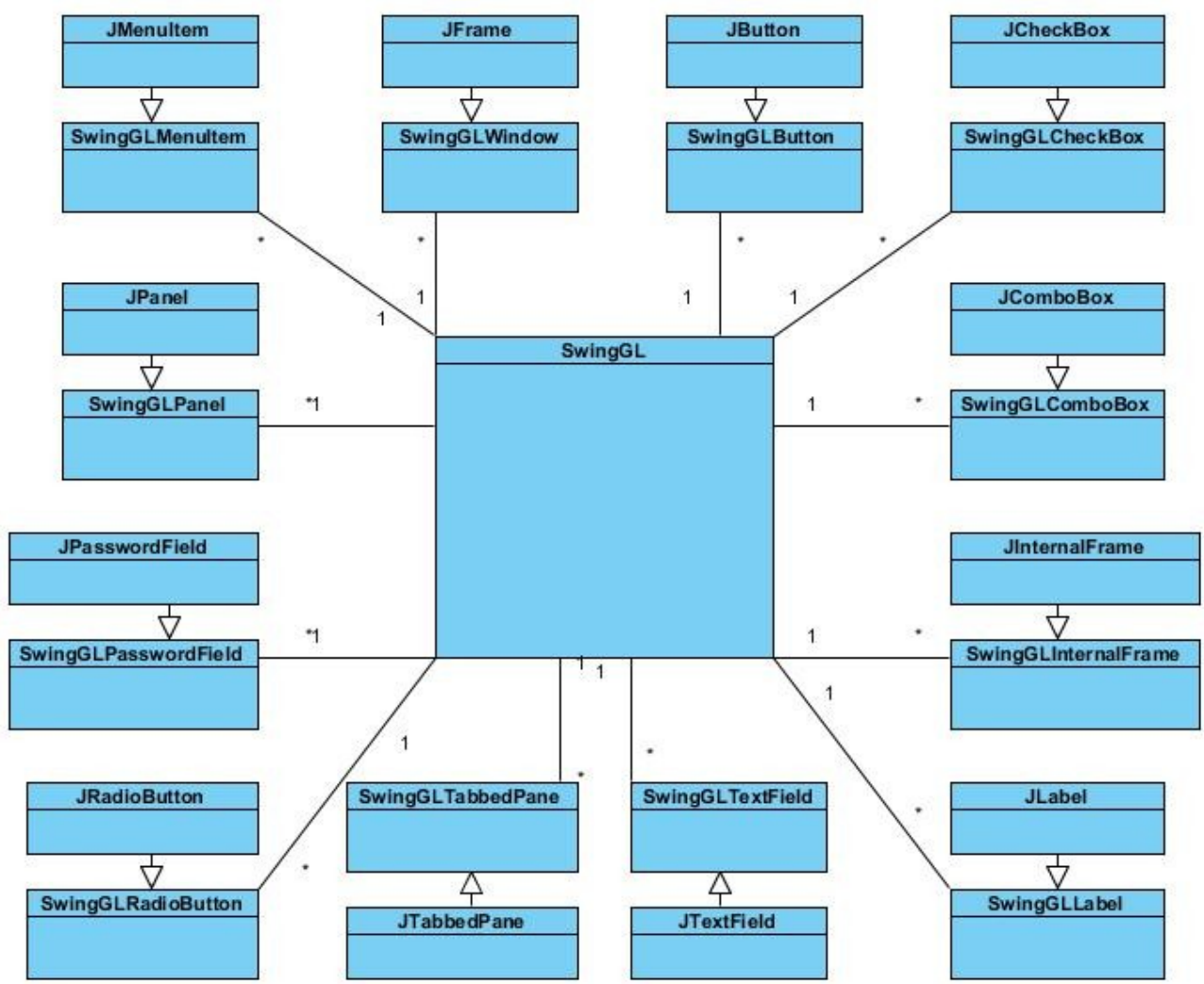
La classe SwingGL, descritta nel seguito, ha il ruolo di classe facciata.

3.3 Struttura delle Classi della Libreria

La libreria è formata da un insieme di classi. La classe più importante ai fini della costruzione e gestione di una interfaccia grafica è la classe principale che si chiama **SwingGL**. Poi sono presenti un insieme di classi che estendono le classi dei componenti grafici di Swing e per le quali è stato definito un costruttore. Poi vi sono anche delle classi di supporto ed una classe che estende java.lang.Exception per la gestione delle eccezioni.

3.3.1 Un Primo Diagramma delle Classi in UML

Viene adesso mostrato il diagramma delle classi in UML che mostra la centralità della classe SwingGL e come i componenti grafici di Swing, le classi che le implementano, siano state ad essere estese dalle classi della libreria.



Come si può notare la classe SwingGL ha funzione di contenitore di componenti grafici, infatti tutte le classi che estendono quelle dei componenti grafici hanno relazione molti a 1 con la classe SwingGL. Questo argomento è oggetto di un successivo e specifico paragrafo. Mentre un interno capitolo è dedicato alle classi che estendono le classi dei componenti grafici di Swing, in modo da farne comprendere il funzionamento.

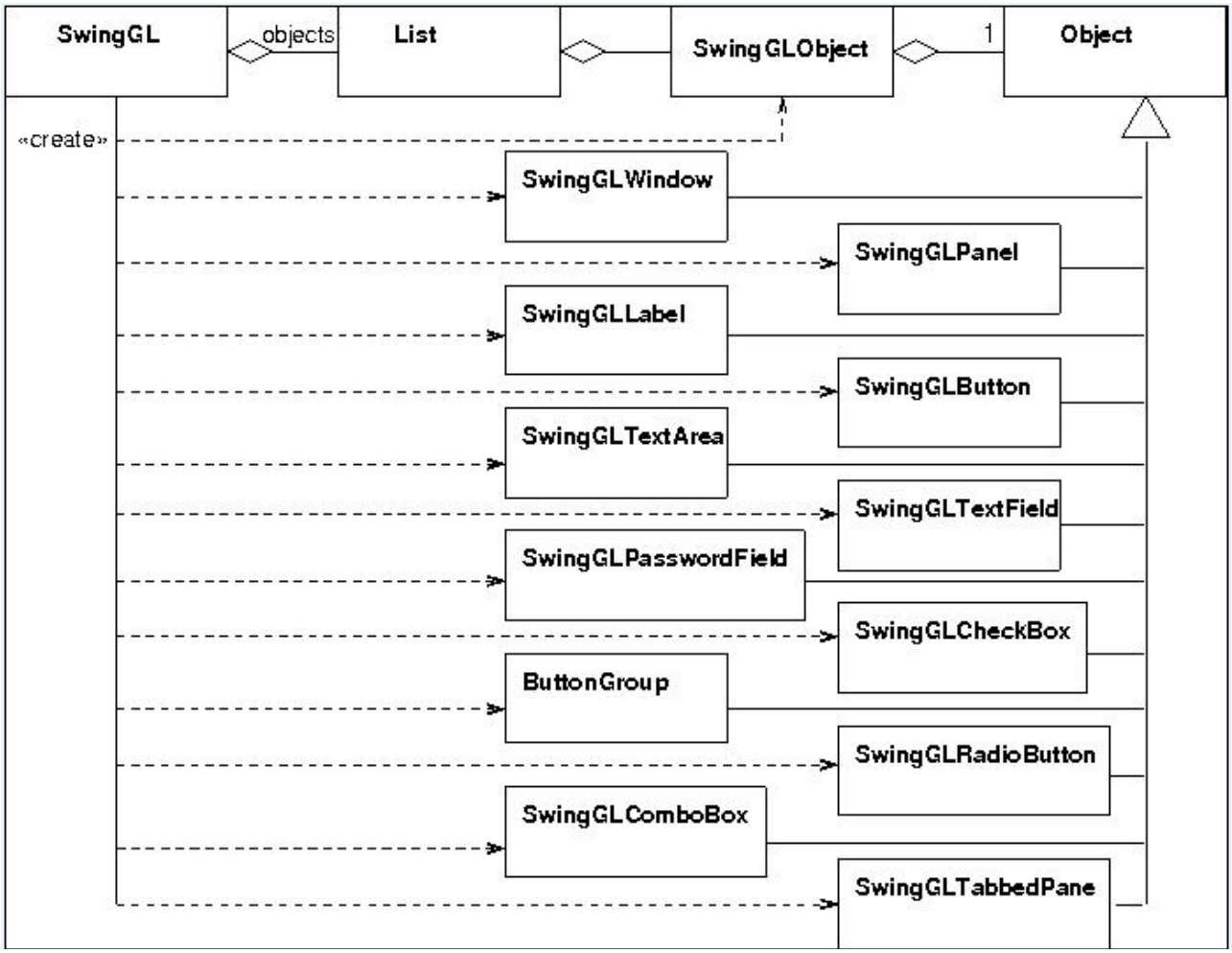
3.4 La Classe Principale

L'idea base che ha caratterizzato tutto il progetto della libreria è stata quella di creare una classe principale che è stata chiamata **SwingGL** la cui conoscenza è necessaria ma soprattutto sufficiente per la creazione e gestione di tutte le interfacce grafiche che la libreria consente di creare. Se per creare una interfaccia grafica in Swing è necessario conoscere tutte le classi di cui il package

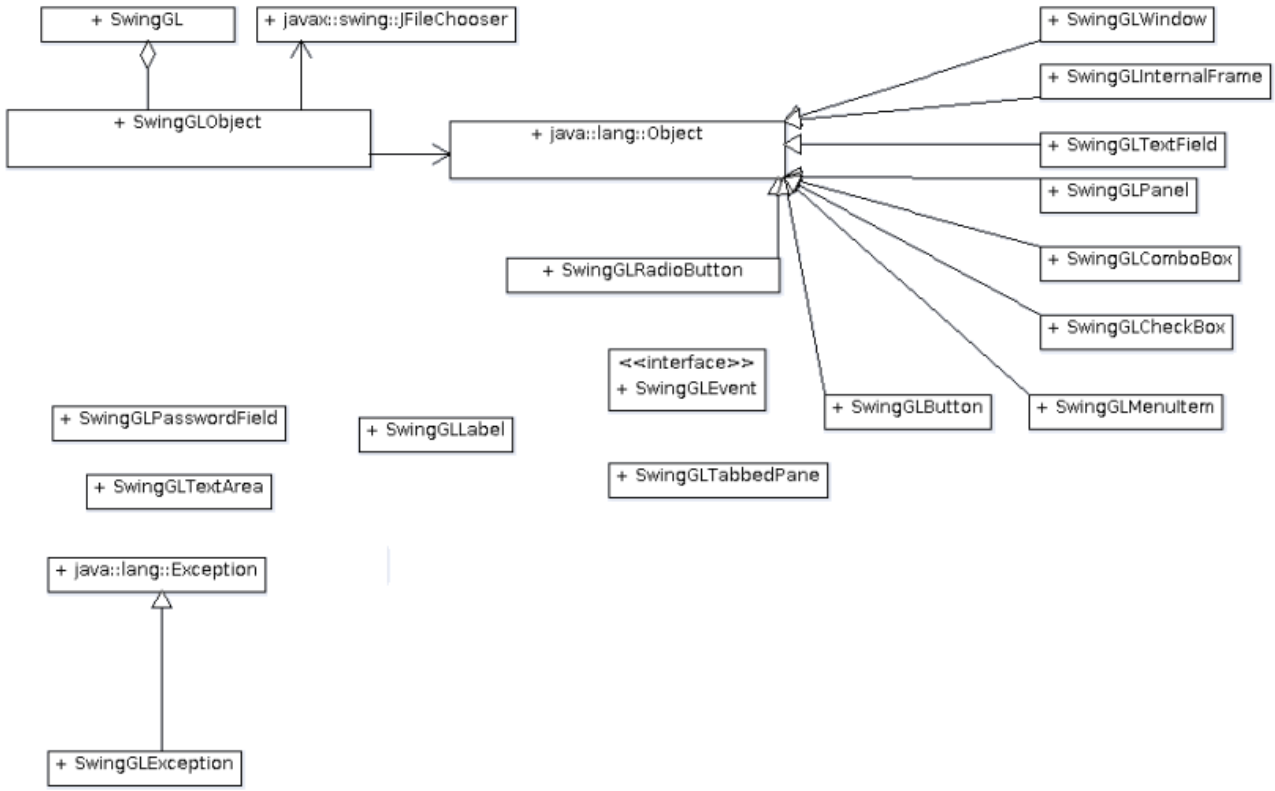
grafico Swing è composto, e dei metodi fondamentali per la visualizzazione e gestione di ogni componente grafico nella nostra libreria non è necessario conoscere tutte le classi di cui essa è composta, ma solamente la classe **SwingGL**. Questa si comporta innanzitutto da contenitore di componenti grafici e mette a disposizione una vasta gamma di metodi per l'aggiunta di componenti grafici e per la loro gestione.
Un altro notevole vantaggio della creazione di una classe principale contenitore di componenti grafici è la possibilità di passare come argomento alle varie funzioni della nostra applicazione l'intera interfaccia grafica, solamente passando il riferimento all'oggetto SwingGL.
I metodi che la classe principale mette a disposizione permettono di creare finestre ed aggiungere a queste componenti grafici. Si è cercato però come obiettivo fondamentale nella creazione di questi metodi, di far si che l'aggiunta e la desiderata visualizzazione di un componente grafico avvenisse soltanto attraverso la chiamata ad un solo metodo della classe principale. Questa è una notevole semplificazione rispetto a Swing, dove è necessario richiamare dai tre ai sei metodi in media per la corretta visualizzazione di un elemento grafico. Prima di mostrare i metodi che la classe principale mette a disposizione passiamo a fare un semplice esempio dell'uso della nostra libreria. L'esempio è il classico “Hello World!” dopo aver mostrato alcuni altri diagrammi .

3.4.1 Altri Diagrammi UML per SwingGL

Al fine di far comprendere meglio la struttura della libreria SwingGL mostriamo altri diagrammi UML. Il primo di questi diagrammi mostra a partire dalla classe Object la struttura della libreria SwingGL mostrando come la classe SwingGL sia un contenitore di oggetti implementato da una lista. Vengono mostrate alcune tra le principali classi che estendono le classi dei componenti grafici e viene evidenziata l'operazione di creazione (<<create>>) di questi componenti grafici.



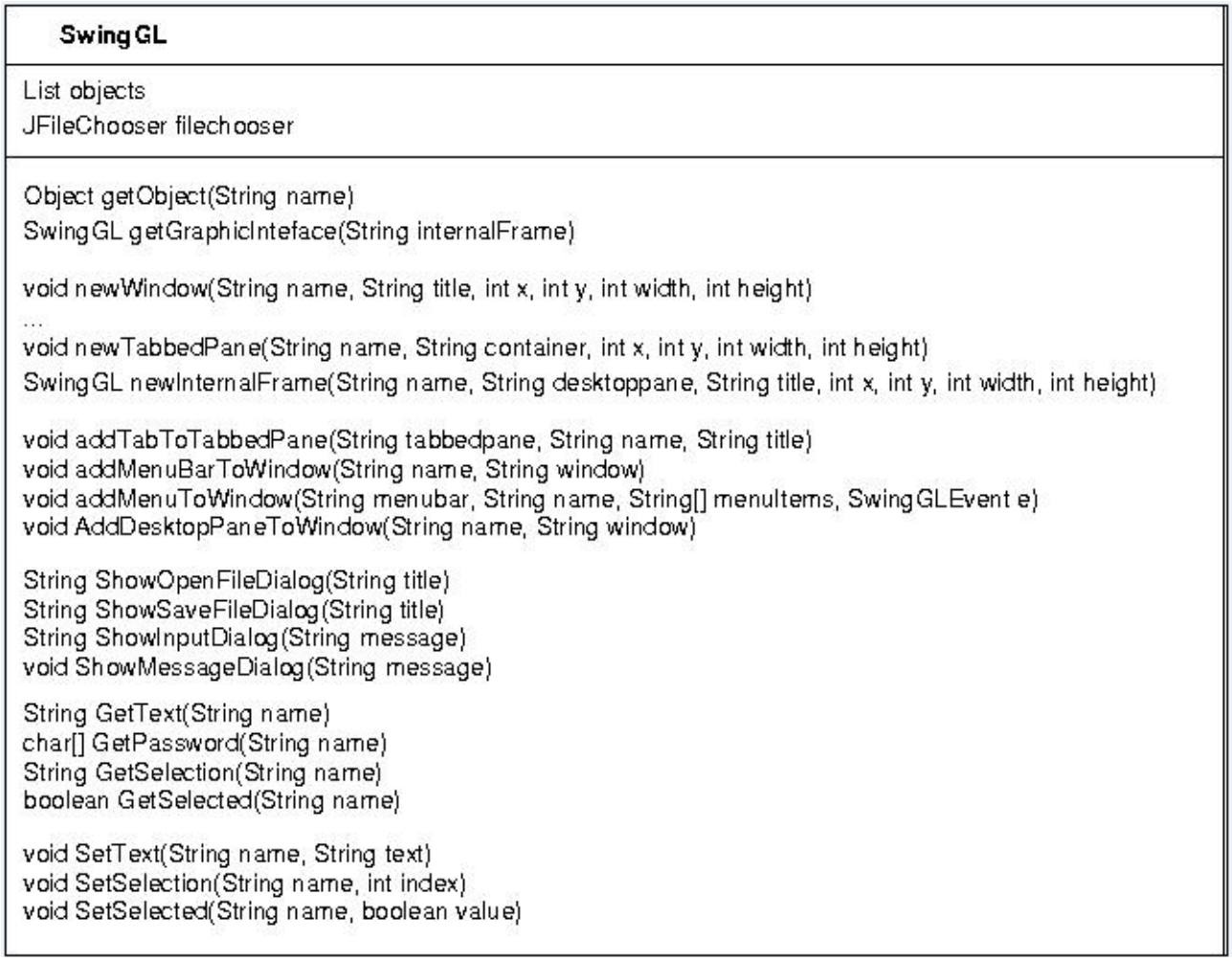
Il secondo diagramma mostra la struttura di SwingGL ed entra anche nel dettaglio delle interfacce che sono state sviluppate allo scopo della gestione degli eventi e della classe derivata da Java.lang.Exception per la gestione delle eccezioni. Volendo descrivere meglio il diagramma possiamo dire che SwingGL è un contenitore di SwingGLObjects che passando per Java.lang.Object contiene il riferimento alle classi che estendono i componenti grafici. Per la gestione degli eventi si mette in evidenza l'interfaccia SwingGLEvent. Per la gestione delle eccezioni la classe SwingGLException derivata da Java.Lang.Exception.



3.4.2 Classe SwingGL e Raggruppamento dei suoi Principali Metodi

Nel seguente schema mostriamo ancora una volta la struttura della classe principale la classe SwingGL. Essa mediante una struttura di tipo List implementa a tutti gli effetti un contenitore di componenti grafici. Vengono mostrate solo alcuni metodi, il dettaglio di tutti i metodi verrà fornito nel seguito, allo scopo di raggruppare questi metodi ed estrarre una struttura che possiamo definire più astratta della libreria. In particolare i metodi vengono ad essere raggruppati come metodi di “New” che servono per la creazione e visualizzazione dei componenti grafici. I metodi di “New” vanno da newWindow a NewInternalFrame secondo l'implementazione. Il secondo raggruppamento mostra i metodi di “Add”. Questi sono metodi che si applicano generalmente ad Oggetti creati precedentemente mediante l'operazione di “New” e sono spesso complementari ai metodi di “New” al fine della creazione dell'interfaccia grafica desiderata. Un altro raggruppamento sono i metodi di “Show” utilizzati per la visualizzazione delle finestre Dialog. Infine abbiamo le funzioni di “Get” e “Set” che sono il cuore del reperimento e del settaggio del contenuto informativo dei componenti grafici (Testo, Selezioni di Opzioni etc.), necessario se vogliamo interagire con l'utente e vogliamo sviluppare una logica elaborativa dell'applicazione che spesso viene fatta a livello di gestione degli

eventi.

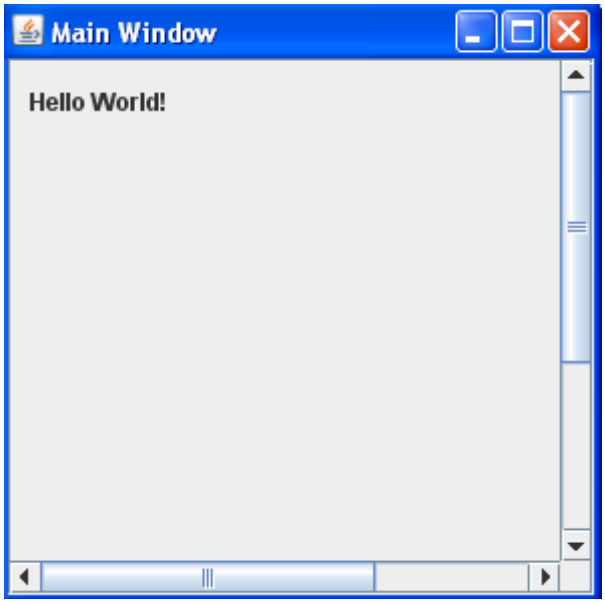


3.4.3 Hello World con SwingGL

L'esempio prevede la creazione di una finestra principale di piccole dimensioni con all'interno una etichetta di testo “Hello World!”

```
package helloworld;
/**
  @author Giovanni Alestra
 **/

import swinggl.*;
public class HelloWorld {

    /**
     * @param args the command line arguments
```

```
     */
    public static void main(String[] args) {

        SwingGL swgl = new SwingGL();
        try {
            swgl.newWindow("Main", "Main Window", 100, 100, 300, 300);
            swgl.newPanel("Panel", "Main", 400, 400);
            swgl.newLabel("Label", "Panel", "Hello World!", 10, 10, 100, 20);
        } catch ( SwingGLException e ) {
            System.out.println ( e.toString());
        }
    }
}
```



3.5 La Classe SwingGL come Contenitore di Oggetti

Come già accennato, la classe SwingGL è un contenitore di oggetti, componenti grafici, ed i suoi metodi consentono l'aggiunta di componenti grafici al contenitore e la loro gestione. A livello di implementazione è stato deciso di utilizzare una Lista. L'implementazione appare così :

```
public class SwingGL {

    List objects;
```

```
   /**
    * Costruttore
    */
   public SwingGL () {
      objects = new LinkedList();
     // ...
   }
…
```

La lista di oggetti viene ad essere riempita con oggetti di tipo SwingGLObject, che rappresentano l'associazione tra nome e oggetto/componenete grafico. L'implementazione della classe SwingGLObject è la seguente :

```
package swinggl;
public class SwingGLObject {

   String name;
   Object object;

   public SwingGLObject ( String n, Object obj ) {
      name = n; object = obj;
   }
}
```

Ogni volta che aggiungiamo un componente grafico alla Lista di oggetti viene creato un oggetto SwingGLObject e quindi associato un nome all'oggetto. L'aggiunta di un oggetto grafico avviene mediante la funzione privata della classe SwingGL addObject (SwingGLObject obj) che lancia una eccezione di tipo SwingGLException se due oggetti hanno lo stesso nome.
Per ottenere un oggetto dal suo nome è possibile utilizzare la funzione della classe SwingGL getObject (String name) che ritorna un riferimento ad oggetto, oppure null nel caso non vi sia alcun oggetto con il nome desiderato.

3.6 Eccezioni in SwingGL

La gestione delle eccezioni in SwingGL ha previsto la definizione di un nuovo tipo di eccezione, la SwingGLException. Questa è una classe che estenda java.lang.Exception e sovrascrive il metodo toString(). Mostriamo l'implementazione della classe SwingGLException :

```
package swinggl;
/**
 *
 * @author Giovanni Alestra
 */

import java.lang.Exception;

public class SwingGLException  extends Exception {
   int code;
   String message;

   public SwingGLException (int c, String m) {
      super();
      code = c;
      message = m;
   }
   @Override
   public String toString() {
      return "SwingGLException : Error Code " + code + " message - " + message;

   }
}
```

In particolare si evidenzia che ogni eccezione ha un codice ed un messaggio di errore. Tutte le funzioni che sono soggette ad errori sono state dichiarate **throws SwingGLException** e rilanciano eccezioni mediante lo statement : **throw new SwingGLException (code, message);**
Per questa ragione le funzioni della classe SwingGL escluso il costruttore per il quale non possono sollevarsi eccezioni devono essere utilizzate all'interno di un blocco try-catch. Ad esempio :

```
SwingGL swgl = new SwingGL();
      try {
         swgl.newWindow("Main", "Main Window", 100, 100, 300, 300);
         swgl.newPanel("Panel", "Main", 400, 400);
         swgl.newLabel("Label", "Panel", "Hello World!", 10, 10, 100, 20);
      } catch ( SwingGLException e ) {
         System.out.println ( e.toString());
      }
```

I principali errori rappresentano il tentativo di inserire oggetti con nomi non univoci, oppure richiamare oggetti mediante un nome errato, oppure utilizzare funzioni su di oggetti di tipo non desiderato.

3.7 Metodi della Classe SwingGL

Riportiamo adesso i metodi che la classe SwingGL mette a disposizione per creare e gestire interfacce grafiche :

3.7.1 Costruttore della Classe SwingGL

- **SwingGL()**

Il costruttore della classe principale, la classe SwingGL, non prevede argomenti e non è soggetto ad eccezioni. Il suo scopo è costruire l'oggetto SwingGL ed inizializzarlo. Non essendo soggetto ad eccezioni non è stato dichiarato **throws SwingGLException** e pertanto può essere richiamato anche all'esterno di un blocco **try-catch**. L'obiettivo del costruttore è inizializzare la lista che conterrà gli oggetti di tipo **SwingGLObject** che sono i componenti grafici di una interfaccia grafica.

3.7.2 Metodo newWindow della classe SwingGL

- **newWindow(String name, String title, int x, int y, int width, int height)**

Il metodo **newWindow** permette di creare una finestra, di assegnare un titolo alla finestra, di posizionarla sul desktop e di determinarne la larghezza e l'altezza. Il campo nome esprime il nome che si assegna all'oggetto creato, che potrà essere utilizzato per richiamare l'oggetto. La funzione **newWindow** crea un oggetto di tipo **SwingGLWindow** classe che estende **JFrame.** Il metodo può essere soggetto ad eccezioni ed è stato dichiarato **throws SwingGLException** pertanto può essere utilizzato soltanto all'interno di un blocco **try-catch**.

3.7.3 Metodo newPanel della classe SwingGL

- **newPanel (String name, String container, int width, int height)**

Il metodo **newPanel** permette di aggiungere un pannello ad un container che tipicamente è una finestra. Come argomenti oltre al nome che si vuole assegnare all'oggetto per richiamarlo successivamente, prevede l'argomento container, che è il nome del contenitore, tipicamente il nome

che è stato assegnato alla finestra. Un pannello può avere dimensioni maggiori di quelle di una finestra, in questo caso vengono visualizzate in maniera automatica delle **Scrollbars.** Il metodo crea un oggetto di tipo **SwinGLPanel** classe che estende **JPanel** e lo aggiunge mediante la funzione di aggiunta dei componenti al **container**. Per la visualizzazione delle **Scrollbars** utilizza anche oggetti di tipo **ScrollPane**. Il metodo è stato dichiarato **throws SwingGLException** in quanto può essere soggetto ad eccezioni e deve quindi essere utilizzato all'interno di un blocco **try-catch**.

3.7.4 Metodo newLabel della classe SwingGL

- **newLabel (String name, String container, String text, int x, int y, int width, int height)**

Il metodo **newLabel** permette di aggiungere una etichetta ad un contenitore che è tipicamente un pannello. Come argomenti oltre al nome che si vuole assegnare all'oggetto per richiamarlo successivamente, prevede l'argomento container, che è il nome del contenitore, tipicamente il nome che è stato assegnato al pannello. Un altro argomento è il testo dell'etichetta, come anche il suo posizionamento con coordinate relative al pannello e la sua dimensione (larghezza ed altezza del componente). Il metodo **newLabel** crea un oggetto di tipo **SwingGLLabel** e lo aggiunge, mediante la funzione di aggiunta dei componenti, al **container**. Anche questo metodo è soggetto ad eccezioni ed è stato dichiarato **throws SwingGLException**, quindi si può utilizzare all'interno di un blocco **try-catch**.

3.7.5 Metodo newTextField della classe SwingGL

- **newTextField (String name, String container, String text, int x, int y, int width, int height)**

Il metodo **newTextField** permette di aggiungere un campo di testo ad un contenitore. Un campo di testo è un componente nel quale l'utente può scrivere del testo. Come argomenti oltre al nome che si vuole assegnare all'oggetto per richiamarlo successivamente, prevede l'argomento container, che è il nome del contenitore, tipicamente il nome che è stato assegnato al pannello. Un altro argomento è il testo di default del campo di testo. Gli altri argomenti rappresentano il posizionamento e la dimensione del componente. Il metodo **newTextField** crea un oggetto di tipo **SwingGLTextField**

che estende **JTextField** e lo aggiunge, mediante la funzione di aggiunta dei componenti, al **container**. Anche questo metodo è soggetto ad eccezioni ed è stato dichiarato **throws SwingGLException**, quindi si può utilizzare all'interno di un blocco **try-catch**.

3.7.6 Metodo newTextArea della classe SwingGL

- **newTextField (String name, String container, String text, int x, int y, int width, int height)**

Il metodo **newTextArea** permette di aggiungere una area di testo ad un contenitore. Una area di testo è un componente simile al **TextField** ma permette la scrittura del testo su più righe. Il discorso è analogo a quello per i **TextField**. Gli oggetti utilizzati sono di tipo **SwingGLTextArea** classe che estende **JTextArea.** Anche questo metodo è soggetto ad eccezioni ed è stato dichiarato **throws SwingGLException**, quindi si può utilizzare all'interno di un blocco **try-catch**.

3.7.7 Metodo newPasswordField della classe SwingGL

- **newPasswordField (String name, String container, int x, int y, int width, int height)**

Il metodo **newPasswordField** permette di aggiungere un campo password che è un campo di testo nel quale si può scrivere che però non mostra il testo digitato ai fini della segretezza della password. I soli argomenti sono il nome da assegnare al componente, il container a cui aggiungere il componente ed il suo posizionamento e la dimensione. Gli oggetti utilizzati sono di classe **SwingGLPasswordField** che estende **JPasswordField.** Deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.8 Metodo newComboBox della classe SwingGL

- **newComboBox (String name, String container, String[] items, int x, int y, int width, int height)**

Il metodo **newComboBox** permette di aggiungere un ComboBox che è un oggetto nel quale è possibile selezionare dei valori predefiniti. I soli argomenti sono il nome da assegnare al componente, il container a cui aggiungere il componente, il vettore di valori Stringa selezionabili, il suo posizionamento e la dimensione. Gli oggetti utilizzati sono di classe **SwingGLComboBox** che estende **JComboBox.** Deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.9 Metodo newCheckBox della classe SwingGL

- **newCheckBox (String name, String container, String text, int x, int y, int width, int height, boolean selected)**

Il metodo **newCheckBox** permette di aggiungere un CheckBox che è un oggetto che permette il settaggio di un valore vero/falso inerente ad una selezione. Gli argomenti sono il nome da assegnare al componente, il container a cui aggiungere il componente, il testo che descrive cosa si sta selezionando. il suo posizionamento e la dimensione ed il valore di default. Gli oggetti utilizzati sono di classe **SwingGLCheckBox** che estende **JCheckBox.** Deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.10 Metodo newButtonGroup della classe SwingGL

- **newButtonGroup (String name)**

Il metodo **newButtonGroup** permette di aggiungere un ButtonGruop che è un oggetto che permette di raggruppare diversi oggetti di tipo RadioButton. Gli argomenti sono il nome da assegnare al componente. Un oggetto ButtonGroup deve essere specificato mediante il suo nome nel metodo **newRadioButton** che verrà illustrato di seguito. Il metodo deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.11 Metodo newRadioButton della classe SwingGL

- **newRadioButton (String name, String container, String group, String text, int x, int y, int width, int height, boolean selected)**

Il metodo **newRadioButton** permette di aggiungere un RadioButton che è un oggetto settabile a vero/falso simile al checkbox, ma del quale è necessario specificare il ButtonGroup di appartenenza. Gli argomenti sono il nome da assegnare al componente, il container, il nome del ButtonGroup di appartenenza, il testo inerente alla selezione, il posizionamento, la dimensione ed il valore di default. Il metodo deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.12 Metodo newTabbedPane della classe SwingGL

- **newTabbedPane (String name, String container, int x, int y, int width, int height)**

Il metodo **newTabbedPane** permette di aggiungere un TabbedPane che è un oggetto al quale è possibile aggiungere dei pannelli selezionabili mediante il loro titolo. Gli argomenti sono il nome da assegnare al componente, il container, il posizionamento e la dimensione. Il metodo utilizza oggetti di tipo **SwingGLTabbedPane** che estendono **JTabbedPane**. Il metodo deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.13 Metodo addTabToTabbedPane della classe SwingGL

- **addTabToTabbedPane (String tabbedpane, String name, String title)**

Il metodo **addTabToTabbedPane** permette di aggiungere una Tab (ovvero un pannello) ad un tabbedpane. Gli argomenti sono il nome del TabbedPane, il nome che si vuole assegnare al pannello ed il suo titolo. Il metodo deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.14 Metodo newButton della classe SwingGL

- **newButton (String name, String container, String text, int x, int y, int width, int height, SwingGLEvent e)**

Il metodo **newButton** permette di aggiungere un Bottone cliccabile e di specificare l'oggetto gestore dell'evento Click. Gli argomenti sono il nome da dare al componente, il nome del contenitore a cui aggiungerlo, il testo che compare sul bottone, la posizione, la dimensione e l'oggetto gestore dell'evento click. Sono utilizzati oggetti di tipo **SwingGLButton** che estendono **JButton**. Il metodo deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.15 Metodo addMenuBarToWindow della classe SwingGL

- addMenuBarToWindow (String name, String window)

Il metodo **addMenuBarToWindow** permette di aggiungere una Barra dei Menù ad una finestra. Gli argomenti sono il nome da dare al componente, ed il nome della finestra. Ad una menù bar possono essere aggiunti i veri e propri menù. Il metodo deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.16 Metodo addMenuToWindow della classe SwingGL

- **addMenuToWindow (String menubar, String name, String menuItems, SwingGLEvent e)**

Il metodo **addMenuToWindow** permette di aggiungere un menù a tendina ad una menùbar. Gli argomenti sono il nome della menùbar, il nome da dare al componente, il vettore di stringhe degli Items cliccabili ed l'oggetto gestore dell'evento click. Il metodo deve essere utilizzato all'interno di un blocco **try-catch** in quanto può essere soggetto ad eccezioni essendo stato infatti dichiarato **throws SwingGLException.**

3.7.17 Metodi showOpenFileDialog e ShowSaveFileDialog della classe SwingGL

- **showOpenFileDialog (String title)**
- **showSaveFileDialog (String title)**

I metodi permettono di visualizzare delle Dialog Windows per l'apertura ed il salvataggio di file. Il metodo **showOpenFileDialog** ritorna il nome del file selezionato, mentre il metodo **showSaveFileDialog**, il nome del file su cui salvare. In entrambi i casi il nome è costituito dal percorso assoluto. L'argomento è il titolo da dare alla finestra di dialogo.

3.7.17 Metodi showMessageDialog e ShowInputDialog della classe SwingGL

- **showMessageDialog (String message)**
- **showInputDialog (String message)**

I metodi permettono di visualizzare delle Dialog Windows per la visualizzazione di un messaggio e per fare una domanda all'utente. Utilizzano oggetti di tipo **JOptionPane**.

3.7.18 Metodo GetText della classe SwingGL

- **GetText (String name)**

Il metodo **GetText** permette di ottenere il testo digitato in un textfield o in una textarea a partire dal nome che è stato assegnato al componente grafico. Deve essere utilizzato all'interno di un blocco **Try-Catch** in quanto provvede a controllare se l'oggetto specificato è del tipo desiderato.

3.7.19 Metodo GetSelection della classe SwingGL

- **GetSelection (String name)**

Il metodo **GetSelection** permette di ottenere il testo selezionato in un combobox a partire dal nome che è stato assegnato al componente grafico. Deve essere utilizzato all'interno di un blocco **Try-Catch** in quanto provvede a controllare se l'oggetto specificato è del tipo desiderato.

3.7.20 Metodo GetSelected della classe SwingGL

- **GetSelected (String name)**

Il metodo **GetSelected** permette di verificare se un chekbox o un radiobutton è stato selezionato. Esso ritorna un valore booleano. Deve essere utilizzato all'interno di un blocco **Try-Catch** in quanto provvede a controllare se l'oggetto specificato è del tipo desiderato.

3.7.21 Metodo SetText della classe SwingGL

- **SetText (String name, String text)**

Il metodo **SetText** permette di settare a run-time il testo di un textfield o di una textare. Deve essere utilizzato all'interno di un blocco **Try-Catch** in quanto provvede a controllare se l'oggetto specificato è del tipo desiderato.

3.7.22 Metodo SetSelection della classe SwingGL

- **SetSelection (String name, int index)**

Il metodo **SetSelection** permette di settare a run-time il testo di un combobox. Deve essere utilizzato all'interno di un blocco **Try-Catch** in quanto provvede a controllare se l'oggetto specificato è del tipo desiderato.

3.7.23 Metodo SetSelected della classe SwingGL

- **SetSelected (String name, boolean value)**

Il metodo **SetSelected** permette di settare a run-time la selezione di un checkbox o di un radiobutton. Deve essere utilizzato all'interno di un blocco **Try-Catch** in quanto provvede a controllare se l'oggetto specificato è del tipo desiderato.

3.7.24 Metodo AddDesktopPaneToWindow della classe SwingGL

- **AddDesktopPaneToWindow (String name, string window)**

Il metodo **AddDesktopPaneToWindow** permette di aggiungere un DesktopPane ad una finestra operazione necessaria si si vogliono utilizzare degli internalframes. Gli argomenti sono il nome da dare al desktoppane ed il nome della finestra a cui aggiungerlo. Deve essere utilizzato all'interno di un blocco **Try-Catch** in quanto provvede a controllare se l'oggetto specificato è di tipo **SwingGLWindow**.

3.7.25 Metodo newInternalFrame della classe SwingGL

- **newInternalFrame (String name, String desktoppane, String title, int x, int y, int width, int height)**

Il metodo **newInternalFrame** permette di visualizzare una finestra dentro un'altra finestra che è detta principale. Gli argomenti sono il nome da dare al componente, il nome del DesktopPane, il titolo della finestra, la sua posizione e la dimensione. Deve essere utilizzato all'interno di un blocco **Try-Catch.**

3.7.26 Metodo getObject della classe SwingGL

- **getObject (String name)**

Il metodo **getObject** permette di ottenere un riferimento di tipo object all'oggetto espresso mediante il nome. Dopo un cast del riferimento object alle classi della libreria o a quelle di swing è possibile utilizzare tutte le funzioni di Swing sul componente grafico.

3.8 Eventi in SwingGL

Analizziamo adesso la gestione degli eventi che è di fondamentale importanza. Per gestire un evento in SwingGL è necessario dichiarare una classe che implementa l'interfaccia SwingGLEvent. Riportiamo l'implementazione di tale interfaccia :

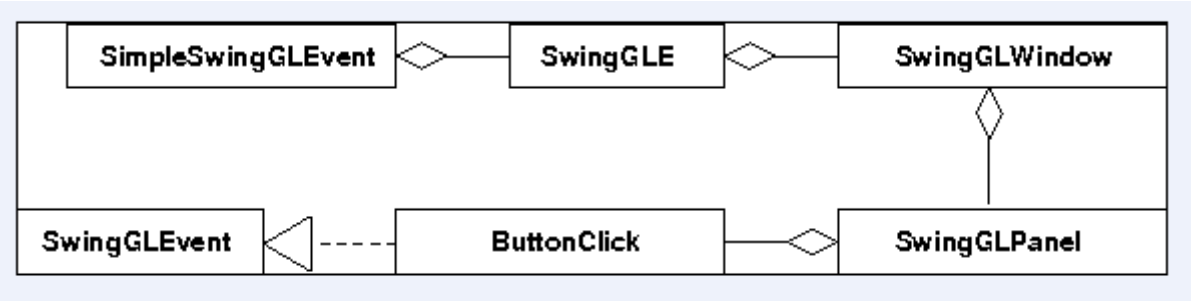
```
package swinggl;

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

public interface SwingGLEvent {

   void Event (String ActionCommand);
}
```

Come notiamo l'interfaccia è costituita da un solo metodo la cui implementazione rappresenta la gestione dell'evento. L'oggetto che implementa SwingGLEvent deve essere passato come argomento ai metodi di **new** della classe SwingGL e questo consente di gestire l'evento. Ripropongo

l'esempio sulla gestione dell'evento click di un bottone fatta in Swing, al fine di mostrare le differenze con quella fatta in SwingGL.

3.8.1 Esempio della Gestione di un Evento in SwingGL

Propongo questo esempio semplicissimo che ci consentirà di capire meglio come si gestiscono gli eventi in SwingGL, schematizzato nella figura seguente:




```
package simpleswingglevent;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class ButtonClick implements SwingGLEvent {

    public void Event (String ActionCommand) {
        System.out.println("Action");
    }
}
```

Come si vede, la classe gestore dell'evento che abbiamo chiamato ButtonClick implementa l'interfaccia SwingGLEvent la quale prevede di implementare una sola funzione, la funzione void Event (String ActionCommand). La gestione prevista prevede unicamente la stampa di un messaggio su prompt dei comandi, nella fattispecie la stringa “Action”.

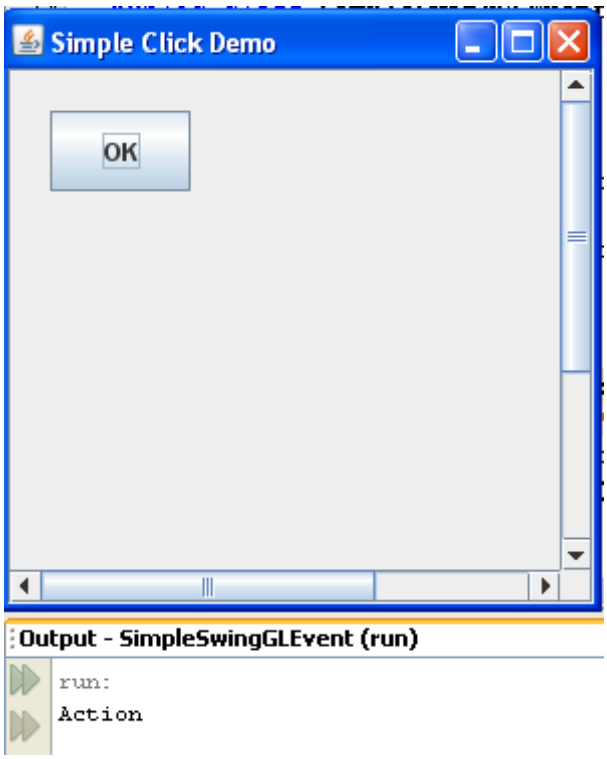
```
package simpleswingglevent;

/**
 *
 * @author Giovanni Alestra
 */
```

L'intefaccia grafica prevista per questo esempio è formata da una finestra, da un pannello e da un bottone cliccabile.

```
import swinggl.*;
```

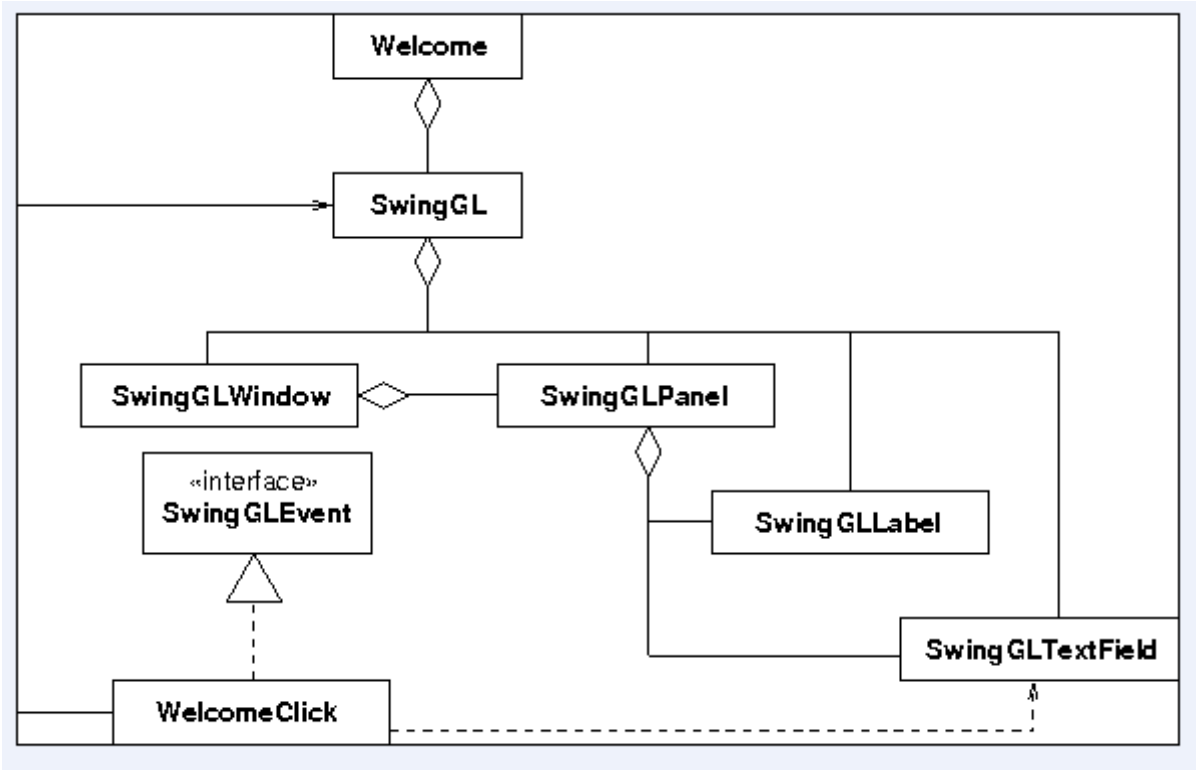
```
public class SimpleSwingGLEvent {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        SwingGL swgl = new SwingGL();
            // Creiamo l'interfaccia grafica

 try {
        swgl.newWindow("Main", "Simple Click Demo", 100, 100, 300, 300);
            //Creiamo la finestra

        swgl.newPanel("Panel", "Main", 400, 400);
            //Aggiungiamo un pannello alla finestra

        ButtonClick buttonclick = new ButtonClick();
            //Istanziamo un oggetto di classe ButtonClick da passare al metodo che aggiunge il bottone

        swgl.newButton("Button", "Panel", "OK", 20, 20, 70, 40, buttonclick);
            //Creiamo il bottone passando anche l'oggetto gestore dell'evento

        } catch ( SwingGLException e ) {
            System.out.println(e.toString());
        }
    }
}
```

Il risultato è quello mostrato in figura.



3.9 Accedere all'Interfaccia Grafica durante la Gestione di un Evento

Per accedere all'interfaccia grafica durante la gestione di un evento, tipicamente per visualizzare nuovi componenti, ottenere il testo immesso dall'utente elaborarlo e fornire i risultati delle elaborazioni è necessario che la classe gestore dell'evento abbia un riferimento all'oggetto SwingGL. Il riferimento a questo oggetto si setta generalmente mediante costruttore. Mostriamo adesso un esempio che prevede l'inserimento del nome dell'utente e l'apparizione di un messaggio di benvenuto nella forma di Message Dialog sfruttando quindi l'accesso all'interfaccia grafica durante la gestione dell'evento. Ho aggiunto i commenti al codice in modo da documentarlo nei minimi particolari, e la struttura di questa semplice applicazione e` mostrata nella figura seguente:

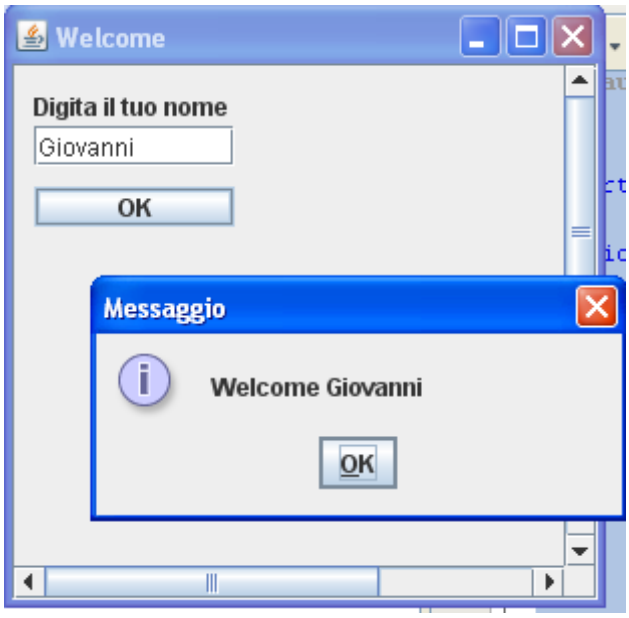



```
package welcome;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class Welcome {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        SwingGL swgl = new SwingGL();
            //Creiamo l'interfaccia grafica
        try {
        swgl.newWindow("Main", "Welcome", 100, 100, 300, 300);
            //Creaiamo una finestra
        swgl.newPanel("Panel", "Main", 400, 400);
            //Aggiungiamo un pannello alla finestra
        swgl.newLabel("Label", "Panel", "Digita il tuo nome", 10, 10, 100, 20);
            //Aggiungiamo una etichetta al pannello
        swgl.newTextField("Nome", "Panel", "", 10, 30, 100, 20);
            //Aggiungiamo il textfield nel quale verrà digitato il nome come richiesto
```

```
      WelcomeClick welcomeclick = new WelcomeClick ( swgl );
        //Istanziamo un oggetto per la gestione dell'evento passando l'oggetto SwingGL che è la nostra interfaccia grafica
    swgl.newButton("button", "Panel", "OK", 10, 60, 100, 20, welcomeclick);
        //Aggiungiamo un bottone al quale passiamo l'oggetto gestore dell'evento

      }catch ( SwingGLException e ) {
           System.out.println ( e.toString() );
        }

   }
}
```

Passiamo adesso all'analisi della classe gestore dell'evento.

```
package welcome;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class WelcomeClick implements SwingGLEvent {
 SwingGL swgl;         // Oggetto inizializzato dal costruttore contenente il riferimento all'interfaccia grafica


   public WelcomeClick ( SwingGL s ) {
      swgl = s;
        // Funzione di Inizializzazione
}

   public void Event ( String ActionCommand ) {
      try {
      String nome = swgl.GetText("Nome");
        // Dall'interfaccia grafica estraiamo una stringa che è il testo digitato nel textfield che abbiamo chiamato “Nome”.

      swgl.ShowMessageDialog("Welcome " + nome);
        // Viene visualizzato nella forma di message dialog un messaggio di benvenuto

      } catch ( SwingGLException e ) {
        System.out.println ( e.toString());
      }
   }
}
```



3.10 Architettura della Gestione degli Eventi in SwingGL

L'architettura per la gestione degli eventi in SwingGL è semplice. Si è deciso per tutti quei componenti per i quali è prevista la gestione degli eventi, non solo di avere una classe che li estende ma si è anche implementata l'interfaccia **ActionListener**. Quindi i componenti che prevedono la gestione degli eventi hanno già implementata l'interfaccia **ActionListener** e quindi anche il metodo **actionPerformed**. Quest'ultimo metodo non fa altro che richiamare il metodo Event dell'oggetto che estende SwingGLEvent.

Per facilitarne la comprensione mostriamo il codice del componente grafico SwingGLButton :

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

public class SwingGLButton extends JButton implements ActionListener {

   SwingGLEvent event;
        //Oggetto che implementa SwingGLEvent per la gestione dell'evento

  public SwingGLButton ( String text, int x, int y, int width, int height, SwingGLEvent e ) {
```

```
      super (text);
         //Chiamata al costruttore della classe JButton

      setBounds (x,y,width,height);
         //Posizionamento e dimensionamento del componente grafico

     addActionListener(this);
         //Aggiunta dell'oggetto this al Listener di Azioni per la gestione degli eventi

      setVisible (true);
         //Oggetto Visualizzabile

      event =  e;
         //Inizializzazione del riferimento all'oggetto gestore dell'evento voluto dal programmatore
  }

   public void actionPerformed(ActionEvent e) {
      if (event==null) return;
         //Se non è previsto un gestore dell'evento all'ora l'evento non viene gestito

       event.Event(e.getActionCommand());
         //Se l'evento viene gestito non si fa altro che richiamare la funzione Event dell'oggetto che
implementa SwingGLEvent con propagazione della gestione dell'evento tra gli oggetti

  }
```

Volendo ulteriormente commentare possiamo dire che la classe SwingGLButton estende la classe Jbutton ed implementa ActionListener. Essa ha un riferimento all'oggetto che implementa SwingGLEvent al suo interno di nome event. Il costruttore della classe oltre ad inizializzare l'oggetto setta anche il riferimento event. Sempre il costruttore richiama il metodo addActionListener passando come argomento this. La funzione che implementata actionPerformed esegue event.Event se l'oggetto event non è nullo.

3.11 Gestione degli Internal Frames

Qualora si voglia costruire una interfaccia grafica formata da una finestra con all'interno delle finestre interne (InternalFrames) bisogna utilizzare la libreria secondo un procedimento che è stato deciso in fase di progetto. Innanzitutto bisognerà creare una finestra principale, con al posto di un pannello un desktop pane, aggiungere alla finestra dei menù ed associare al cliccaggio dei menù la creazione degli internal frames. Gli internal frames sono spesso utilizzate come finestre documento, ovvero finestre aventi uguale struttura ma diverso contenuto informativo. Avendo uguale struttura ci si aspetterebbe di crearle sempre allo stesso modo, ma questo non è possibile se si utilizza sempre lo

stesso oggetto di tipo SwingGL, in quanto si crea un conflitto di nomi. Non è possibile aggiungere ad un oggetto SwingGL più oggetti con lo stesso nome. La soluzione prevista a livello architetturale dalla librerie è l'utilizzo di più oggetti SwingGL, così facendo oggetti con nomi uguali chiaramente possono coesistere in oggetti SwingGL diversi. Quindi quando si crea un Internal Frame,
si crea anche un oggetto SwingGL a lui proprio e si può utilizzare questo oggetto grafico per aggiungere componenti grafici al frame e per passarlo agli oggetti che gestiscono gli eventi del frame. Al fine di chiarire quanto detto facciamo un esempio di applicazione pratica di questi concetti. Ripropongo l'esempio del benvenuto all'utente, creando una applicazione che sfrutta l'architettura per gli internal frames.

3.11.1 Esempio: Welcome con Internal Frames

Mostriamo adesso un esempio che prevede l'utilizzo di internal frames. L'utilizzo di internal frames prevede innanzitutto una forma standard di finestra principale formata esclusivamente da una menùbar con dei menù ed un desktoppane.

```
package welcomeinternalframes;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class WelcomeInternalFrames {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
          SwingGL swgl = new SwingGL();
           //Creiamo l'intefaccia grafica

        try {
           swgl.newWindow("Main", "Welcome Internal Frames", 100, 100, 500, 400);
           /Creiamo la finestra

           swgl.addMenuBarToWindow("Menu", "Main");
           //Aggiungiamo una menùbar
```

```
            String[] items = { "Welcome" };
             MenuClick menuclick = new MenuClick ( swgl );
             // Creiamo un oggetto gestore dell'evento click per il menù

             swgl.addMenuToWindow("Menu", "Utente", items, menuclick);
             //Aggiungiamo un menù con relativo oggetto gestore dell'evento click

             swgl.AddDesktopPaneToWindow("DesktopPane", "Main");
             //Aggiungiamo un desktoppane alla finestra principale


         } catch ( SwingGLException e ) {
            System.out.println ( e.toString() );
         }
      }
   }
```

La classe gestore dell'evento click per il menù deve creare gli internal frames. Mostriamo adesso una procedura corretta.

```
package welcomeinternalframes;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class MenuClick implements SwingGLEvent {
   SwingGL swgl;
         //Riferimento all'interfaccia grafica

   int count;
         //Contatore utilizzato per formare il nome del frame

   public MenuClick ( SwingGL s ) {
      swgl = s;
      count = 1;
         //Inizializzazione del riferimento all'interfaccia grafica e del contatore

   }

   public void Event ( String ActionCommand ) {
      if ( ActionCommand.equals("Welcome")) {
      //Se è stato cliccato il menù “Welcome”
            try {
```

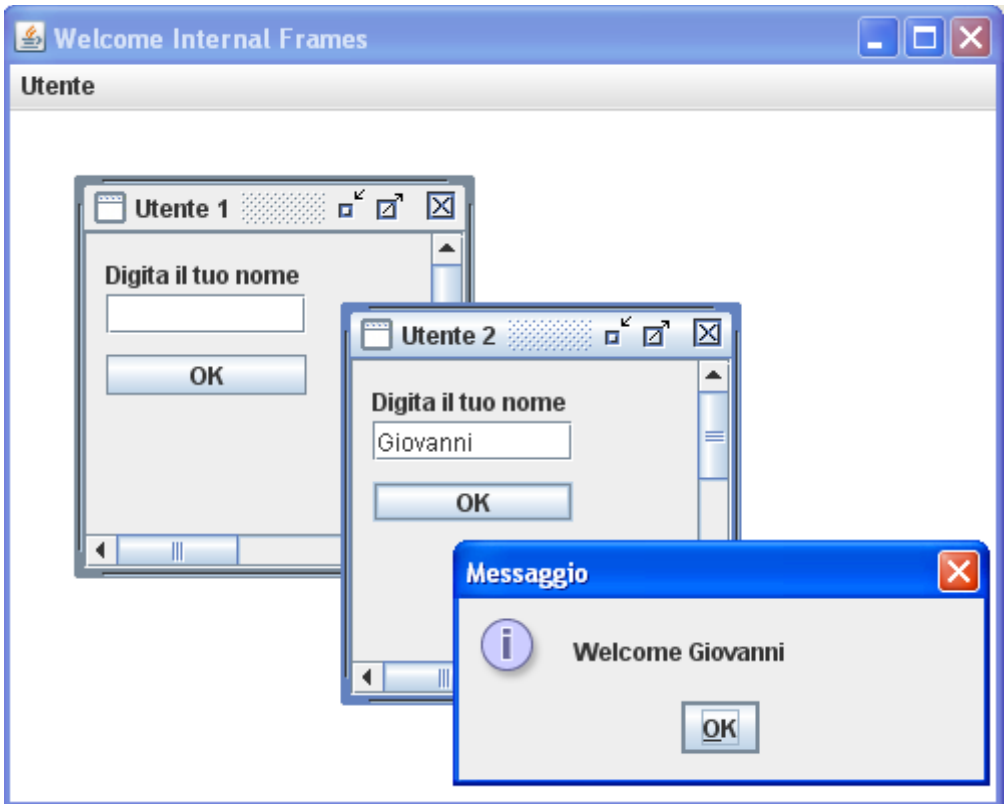
```
            String frameName = "Utente "+count; count++;
                //Formiamo il nome del frame ed aggiorniamo il contatore

                SwingGL iframe = swgl.newInternalFrame(frameName, "DesktopPane", frameName,
20+20*count, 20+20*count, 200, 200);
                //Creaiamo un internal frame e memorizziamo in un riferimento l'oggetto SwingGL a
lui proprio

            iframe.newPanel("Panel", frameName, 400, 400);
            iframe.newLabel("Label", "Panel", "Digita il tuo nome", 10, 10, 100, 20);
            iframe.newTextField("Nome", "Panel", "", 10, 30, 100, 20);
                //Utilizziamo l'oggetto SwingGL proprio dell'internalframe per aggiungere i
componenti grafici

             WelcomeClick welcomeclick = new WelcomeClick ( iframe );
                // Istanziamo un oggetto gestore dell'evento click per un bottone all'interno
dell'internal frame passando come oggetto SwingGL l'oggetto proprio dell'internalframe

             iframe.newButton("button", "Panel", "OK", 10, 60, 100, 20, welcomeclick);
                //Creiamo un bottone cliccabile con gestione dell'evento click associato all'oggetto
welcomeclick

          } catch ( SwingGLException e ) {
             System.out.println ( e.toString());
          }
       }
    }
}
```

Il gestore dell'evento dell'oggetto welcomeclick è uguale a quello descritto precedentemente e consente la visualizzazione di un message dialog con un messaggio di benvenuto.

```
package welcomeinternalframes;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class WelcomeClick implements SwingGLEvent {
  SwingGL swgl;

    public WelcomeClick ( SwingGL s ) {
       swgl = s;
    }

    public void Event ( String ActionCommand ) {
       try {
```

```
        String nome = swgl.GetText("Nome");
        swgl.ShowMessageDialog("Welcome " + nome);
        } catch ( SwingGLException e ) {

            System.out.println ( e.toString());
        }

    }
}
```

Rusultato Esempio :



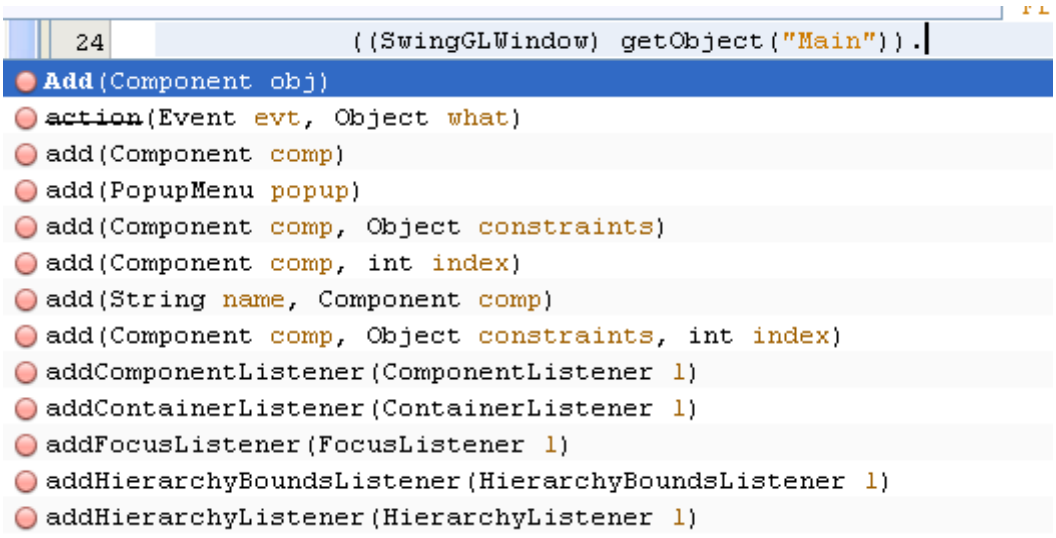
Analizzando l'esempio notiamo che alle finestre principali è stato semplicemente aggiunto un menù ed un Desktop Pane. La gestione del menù consente di creare i frames. In particolare la classe gestore ha un campo contatore che viene utilizzato per formare il nome (una Stringa) del frame e viene incrementato ad ogni click. A partire dal nome del frame si istanzia un oggetto SwingGL che si ottiene richiamando dall'oggetto SwingGL principale la funzione newInternalFrame. Dopo di che si utilizza questo oggetto secondario per aggiungere i componenti al frame e come notiamo viene anche ad essere passato agli oggetti che gestiscono eventi.

Quindi possiamo affermare che è stata reso per quanto possibile automatico l'utilizzo di più oggetti SwingGL, anzi di oggetti SwingGL dedicati a Frames lasciando intatti i principi di progettazione e di sviluppo della libreria e delle applicazioni realizzabili mediante essa.

3.12 Il Metodo getObject

La classe SwingGL mette a disposizione pubblicamente il metodo getObject che ritorna l'oggetto/componente grafico dato il suo nome. Questa funzione è stata volutamente dichiarata pubblica perchè si vuole consentire di sfruttare tutte le potenzialità di Swing, ovvero poter richiamare sull'oggetto tutte le funzioni che Swing ha messo a disposizione per lui. Se la libreria non consente di fare qualcosa su di un oggetto ma Swing lo permette è possibile allora ottenere l'oggetto dal suo nome mediante la funzione getObject e dopo un cast dell'oggetto alle classi della libreria o a quelle di Swing è possibile richiamare tutte le funzioni che Swing mette a disposizione.

Ecco una immagine esplicativa del concetto :



Questo è stato fatto in modo da poter consentire un approfondimento sulle funzioni che Swing mette a disposizione. E' possibile fare un cast degli oggetti alle classi di Swing o a quelle della libreria che estendono le classi di Swing. Per la documentazione sulle classi della libreria è possibile consultare il JavaDoc.

3.13 Considerazioni

La libreria è stata dunque progettata in modo da presentare gran parte dei componenti grafici di Swing, e permette di utilizzare questi componenti grafici a partire dalla conoscenza dei metodi che la classe principale, la classe SwingGL, mette a disposizione. La gestione degli eventi è stata resa semplice ed automatizzata. La libreria è stata progettata in modo da essere molto intuitiva nel suo utilizzo.

Capitolo 4

La Realizzazione

4.1 Classi che Compongono la Libreria

La Libreria dunque altro non è che un insieme di classi che sono state implementate dovendo realizzare i principi studiati nella fase di progettazione. Mostreremo adesso le classi che compongono la libreria e come sono state implementate senza però, per brevità, mostrarne il codice, che come sappiamo è comunque a portata di tutti.

Package swinggl

Interface Summary	
SwingGLEvent	

Class Summary

SwingGL	
SwingGLButton	
SwingGLCheckBox	
SwingGLComboBox	
SwingGLInternalFrame	
SwingGLLabel	
SwingGLMenuItem	
SwingGLObject	
SwingGLPanel	
SwingGLPasswordField	
SwingGLRadioButton	
SwingGLTabbedPane	
SwingGLTextArea	
SwingGLTextField	
SwingGLWindow	

Exception Summary	
SwingGLException	

.

4.1.1 Interface SwingGLEvent

L'interfaccia SwingGLEvent è quella che una classe deve implementare se vuole essere un gestore di un evento. Essa è composta dal metodo void Event (String ActionCommand) che deve essere implementato e la cui implementazione rappresenta la logica di elaborazione dati di una gestione di un evento. Riporto il codice :

```
package swinggl;

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

public interface SwingGLEvent {
   void Event (String ActionCommand);
}
```

4.1.2 Classe SwingGL

E' la classe principale, consente la creazione di interfacce grafiche. E' stato oggetto di discussione nel capitolo dedicato al progetto. A livello di implementazione possiamo dire che essa serve ad aggiungere componenti grafici ai contenitori ed è formato da funzioni di **NEW**, funzioni di **ADD**, funzioni di **GET** e funzioni di **SET,** I tutti i suoi metodi escluso il costruttore sono stati dichiarati **throws SwingGLException**. I metodi che la classe mette a disposizione sono stati tutti riportati nel capitolo precedente, e saranno esaminati uno per uno nel manuale allegato a questo documento.

4.1.3 Classe SwingGLButton

Estende la classe **Jbutton** di Swing, implementa **ActionListener**, e permette la visualizzazione di un bottone cliccabile cui è associata la gestione dell'evento click. Ha un unico metodo che è il costrutture:

FUNZIONI SVOLTE DAL COSTRUTTORE

- **super // Richiama il costruttore di JButton**
- **setBounds // Posiziona il componente grafico**
- **addActionListener // Aggiunge un ActionListener per la gestione dell'evento click**
- **setVisible // Visualizza l'oggetto**

4.1.4 Classe SwingGLCheckBox

Estende la classe **JCheckBox** di Swing, e permette la visualizzazione di un testo selezionabile.

FUNZIONI SVOLTE DAL COSTRUTTORE

- **super // Richiama il costruttore di JCheckBox**
- **setBounds // Posiziona il componente grafico**
- **setSelected // Setta la selezione di default**
- **setVisible // Visualizza l'oggetto**

4.1.5 Classe SwingGLComboBox

Estende la classe **JComboBox** di Swing, e permette la scelta di un testo tra una lista di possibilità.

FUNZIONI SVOLTE DAL COSTRUTTORE

- **super // Richiama il costruttore di JComboBox**
- **setBounds // Posiziona il componente grafico**
- **setSelectedIndex // Setta la selezione di default**
- **setVisible // Visualizza l'oggetto**

4.1.6 Classe SwingGLInternalFrame

Estende la classe **JInternalFrame** di Swing, e permette la visualizzazione di un Frame all'interno di una finestra

FUNZIONI SVOLTE DAL COSTRUTTORE

- **super // Richiama il costruttore di JInternalFrame**
- **setLayout// Setta il Layout del Frame**
- **setDefaultCloseOperation // Setta l'operazione di default per la chiusura**
- **setBound // Setta posizione e dimensione**
- **setVisible // Visualizza l'oggetto**

4.1.7 Classe SwingGLLabel

Estende la classe **Jlabel** di Swing, e permette la visualizzazione di una etichetta

FUNZIONI SVOLTE DAL COSTRUTTORE

- **super // Richiama il costruttore di JLabel**

- **setBounds // Setta la posizione e la dimensione**
- **setVisible // Visualizza l'oggetto**

4.1.8 Classe SwingGLMenuItem

Estende la classe **JMenuItem** di Swing ed implementa **ActionListener**, e permette la visualizzazione di un menù a tendina

FUNZIONI SVOLTE DAL COSTRUTTORE

- **super // Richiama il costruttore di JLabel**
- **addActionListener // Aggiunge il Listener per l'evento click**

4.1.9 Classe SwingGLObject

Utilizzata per uso interno associa un nome ad un componente grafico.

4.1.10 Classe SwingGLPanel

Estende la classe **JPanel** di Swing, e permette la visualizzazione di un pannello

FUNZIONI SVOLTE DAL COSTRUTTORE

- **setLayout // Setta il layout del pannello**
- **setVisible // Visualizza il componente**

4.1.11 Classe SwingGLPasswordField

Estende la classe **JPasswordField** di Swing, e permette la visualizzazione di un campo password

FUNZIONI SVOLTE DAL COSTRUTTORE

- **setBounds // Posiziona e dimensiona in componente**
- **setVisible // Visualizza il componenete**

4.1.12 Classe SwingGLRadioButton

Estende la classe **JradioButton** di Swing, e permette la visualizzazione di un radio button

FUNZIONI SVOLTE DAL COSTRUTTORE

- **super // Richiama il costruttore di JradioButton**
- **setBounds // Posiziona e dimensiona il componente**
- **setSelected // Seleziona la selezione**
- **setVisible // Visualizza il componente**

4.1.13 Classe SwingGLTabbedPane

Estende la classe **JTabbedPane** di Swing, e permette la visualizzazione di un tabbed pane

FUNZIONI SVOLTE DAL COSTRUTTORE

- **setBounds // Posiziona e dimensiona il componente**
- **setVisible // Visualizza il componente**

4.1.14 Classe SwingGLTextArea

Estende la classe **JTextArea** di Swing, e permette la visualizzazione di una area di testo

FUNZIONI SVOLTE DAL COSTRUTTORE

- **setBounds // Posiziona e dimensiona il componente**
- **setText // Setta il testo di default**

- **setVisible // Visualizza il componente**

4.1.15 Classe SwingGLTextField

Estende la classe **JTextField** di Swing, e permette la visualizzazione di un campo di testo

FUNZIONI SVOLTE DAL COSTRUTTORE

- **setBounds // Posiziona e dimensiona il componente**
- **setText // Setta il testo di default**
- **setVisible // Visualizza il componente**

4.1.16 Classe SwingGLWindow

Estende la classe **JFrame** di Swing, e permette la visualizzazione di una finestra

FUNZIONI SVOLTE DAL COSTRUTTORE

- **super // Richiama il costruttore di JradioButton**
- **setBounds // Posiziona e dimensiona il componente**
- **setLayout // Setta il Layout della finestra**
- **setDefaultCloseOperation // Setta l'operazione di chiusura di default**
- **setVisible // Visualizza il componente**

4.1.17 Classe SwingGLExecption

Estende la classe **java.lang.Exception,** e rappresenta il tipo di eccezione lanciata in situazioni erronee dalla libreria.

4.2 Componenti di Swing Utilizzati in Forma Pura

Alcuni componenti di Swing come gli **ScrollPane,** i **DesktopPane** e le **MenuBar** sono state utilizzate in forma pura ovvero senza una classe che li estendesse.

Capitolo 5

Un Esempio Pratico

5.1 Implementazione di un Esempio Pratico a Fini Dimostrativi

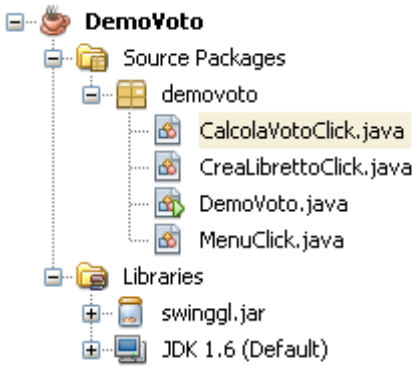
Si mostrerà adesso come sia semplice utilizzare la libreria attraverso l'implementazione di un esempio pratico che potrebbe a tutti gli effetti rappresentare una vera e propria esercitazione di laboratorio.

.5.1.1 Descrizione dell'Esercizio

Si vuole realizzare una interfaccia grafica in grado di calcolare il voto di laurea di uno studente, sia esso di laurea triennale che di laurea specialistica. Devono essere utilizzati gli internal frames e create delle finestre per l'immissione delle generalità dello studente, come nome cognome matricola corso di laurea, e deve essere creato un internal frame rappresentante il libretto dello studente che deve essere compilato ai fini della valutazione del voto di laurea. Se uno studente è di laurea specialistica dovrà inserire anche il voto di laurea triennale che avrà valore nella valutazione del

voto finale.

5.1.2 Implementazione dell'Applicazione DemoVoto



Ecco l'implementazione delle classi che descriveremo minuziosamente con commenti al codice :

La prima classe è la classe DemoVoto contenente la funzione principale (main). In questa classe creiamo la finestra principale alla quale aggiungiamo un menù ed un desktoppane. Come sappiamo l'utilizzo di internalframes prevede questa pratica standard a proposito della finestra principale.

```
package demovoto;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class DemoVoto {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        SwingGL swgl = new SwingGL();
            //Creiamo l'interfaccia grafica

        try {
            swgl.newWindow("Main", "Calcolo del Voto di Laurea", 10, 10, 700, 520);
                    //Creiamo la finestra principale
```

```
            swgl.addMenuBarToWindow("Menu", "Main");
                    //Aggiungiamo una menùbar
            String[] items = { "Studente" };
            MenuClick menuclick = new MenuClick (swgl);

            swgl.addMenuToWindow("Menu", "Studente", items, menuclick);
                    //Aggiungiamo un menù

           swgl.AddDesktopPaneToWindow("DesktopPane", "Main");
                    //Aggiungiamo un desktoppane

        } catch ( SwingGLException e ) {
            System.out.println ( e.toString());
        }
    }
}
```

Come commenti aggiuntivi possiamo dire che la logica di generazione degli internalframes è contenuta nella classe gestore dell'evento click per il menù che viene adesso descritta. La classe MenuClick deve implementare l'interfaccia SwingGLEvent e generare gli internalframes per l'immissione delle generalità dello studente.

```
package demovoto;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class MenuClick implements SwingGLEvent {
    SwingGL swgl;
            //Riferimento all'interfaccia grafica

    int count;
            //Contatore utilizzato per formare il nome dell'internalframe

    public MenuClick (SwingGL s) {
        swgl = s;
        count = 1;
            //Inizializzazione dei campi
}

    public void Event ( String ActionCommand ) {
        if (ActionCommand.equals("Studente")) {
            //Gestione dell'evento click per il menù
```

```
        try {
           String frameName = "Studente " + count; count++;
                //Composizione del nome dell'internalframe ed incremento del contatore

          SwingGL iframe = swgl.newInternalFrame(frameName, "DesktopPane", frameName,
10+20*count, 10+20*count, 270, 310);
                //Creazione dell'internalframe

           iframe.newPanel("Panel", frameName, 400, 400);
                //Aggiunta all'internal frame di un pannello

                //Le seguenti righe di codice rappresentano l'aggiunta al frame di altri componenti
grafici che caratterizzano la schermata visualizzata come etichette, textfields, radiobutton,
combobox
           iframe.newLabel("Label", "Panel", "Nome e Cognome", 10, 10, 200, 20);
           iframe.newTextField("Nome", "Panel", "", 10, 30, 200, 20);
           iframe.newLabel("Label2", "Panel", "Matricola", 10, 50, 100, 20);
           iframe.newTextField("Matricola", "Panel", "", 10, 70, 100, 20);
           iframe.newLabel("Label3", "Panel", "Corso di Laurea", 10, 90, 200, 20);
           iframe.newTextField("Corso", "Panel", "", 10, 110, 200, 20);
           iframe.newButtonGroup("BG");
           iframe.newRadioButton("Triennale", "Panel", "BG", "Triennale", 10, 145, 100, 20, true);
              iframe.newRadioButton("Specialistica", "Panel", "BG", "Specialistica", 100, 145, 100,
20,false);
           iframe.newLabel("Label4", "Panel", "Numero Materie", 30, 180, 100, 20);
                                                                          String[] nmat =
{"5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20"};
           iframe.newComboBox("NMaterie", "Panel", nmat, 150, 180, 50, 20);


                CreaLibrettoClick crealibretto = new CreaLibrettoClick ( swgl, iframe);
                //Classe gestore dell'evento click per la creazione del libretto universitario che viene
ad essere visualizzato anch'esso con un internalfreme. Viene passata l'interfaccia grafica principale e
l'internalframe appena creato

      iframe.newButton("CreaLibretto", "Panel", "Crea Libretto", 10, 210, 200, 20, crealibretto);
                //Aggiunta del bottone per la creazione del libretto

        } catch ( SwingGLException e ) {
           System.err.println( e.toString());
        }
     }
   }
}
```

Come ulteriori commenti possiamo dire che abbiamo creato un internalframe, con dei componenti grafici nei quali l'utente inserisce i propri dati ed un bottone per la creazione e visualizzazione del libretto universitario. Tuttavia abbiamo anche deciso di inserire una validazione dei campi immessi dall'utente nella gestione dell'evento click della classe CreaLibretto. La classe CreaLibretto è di seguito descritta. Essa deve implementare SwingGLEvent.

```
package demovoto;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class CreaLibrettoClick implements SwingGLEvent {
       SwingGL swgl;
          //Riferimento all'interfaccia grafica principale

SwingGL iframe;
          //Riferimento all'internalframe dove è stato generato l'evento

   int count;
          //Contatore utilizzato per formare il nome dell'internalframe rappresentante il libretto

   String nome;
   String matricola;
   String corsodilaurea;
   int numeromaterie;
   boolean specialistica;
          //Campi che vengono inizializzati in base alle informazioni immesse dall'utente


   public CreaLibrettoClick ( SwingGL s, SwingGL i ) {
      swgl = s;
      iframe = i;
      count = 1;
          //Inizializzazione dei vari riferimenti e del contantore
   }

   public void Event ( String ActionCommand ) {
     //Gestione dell'evento click per il bottone
          try {
          nome = iframe.GetText("Nome");
          matricola = iframe.GetText("Matricola");
          corsodilaurea = iframe.GetText("Corso");
          specialistica = iframe.GetSelected("Specialistica");
                    //Lettura delle informazioni immesse dall'utente mediante le funzioni di GET e SET

                    //Verifica che tutti i dati immessi siano non nulli
          if ( nome.equals("") || matricola.equals("") || corsodilaurea.equals("")) {
             swgl.ShowMessageDialog("Compilare i Campi Correttamente!");
                    //Se qualche dato è nullo visualizzazione di una message dialog che invita a
compilare i dati correttamente
```

```
return;
        }

                //Parsing del numero delle materie del libretto
        try {
           numeromaterie = Integer.parseInt(iframe.GetSelection("NMaterie"));
        } catch ( Exception e ) {}

        String frameName = "Libretto "+count; count++;
        String frameTitle = "Libretto " + nome;
                //Formazione del nome dell'internalframe e del suo titolo


SwingGL libretto = swgl.newInternalFrame(frameName, "DesktopPane", frameTitle,
300+count*20, 10+count*20, 300, 400);
        //Crazione dell'internalframe Libretto

        libretto.newPanel("Panel", frameName,300,200+numeromaterie*30);
        libretto.newLabel("Label", "Panel", "Corso di Laurea in "+corsodilaurea, 10, 10, 300, 20);
        libretto.newLabel("Label2", "Panel", nome+" "+matricola , 10, 30, 300, 20);

        libretto.newLabel("Label5", "Panel", "Materia", 10, 80, 100, 20);
        libretto.newLabel("Label6", "Panel", "Crediti", 180, 80, 50, 20);
        libretto.newLabel("Label7", "Panel", "Voto", 220, 80, 30, 20);
        //Aggiunta all'internalframe di qualche componente grafico

        for (int i=0; i<numeromaterie; i++) {
           libretto.newTextField("Materia"+i, "Panel", "", 10, 110+i*30, 150, 20);
           libretto.newTextField("Crediti"+i, "Panel", "", 180, 110+30*i, 20, 20);
           libretto.newTextField("Voto"+i, "Panel", "", 210, 110+i*30, 20,20);
        }
                //Aggiunta di un numero di textfield pari al numero delle materie, in particolare
Nome Materia, Crediti e Voto


                //Se la laurea è specialistica aggiunta di un textfield per l'immissione del voto di
laurea triennale
         if ( specialistica ) {
            libretto.newLabel("Label4", "Panel", "Voto Triennale", 10, 130+numeromaterie*30 , 200,
20);
           libretto.newTextField("Voto", "Panel", "", 100, 130+numeromaterie*30, 30, 20);
        }

                CalcolaVotoClick calcolavoto = new CalcolaVotoClick ( libretto, numeromaterie,
specialistica );
                //Classe gestore dell'evento click CalcolaVoto che serve a calcolare il voto di laurea

         libretto.newButton("Calcola","Panel", "Calcola Voto", 10, 170+numeromaterie*30, 200, 20,
calcolavoto);
                //Aggiunta del bottone il cui cliccaggio avvia il calcolo del voto di laurea
```

```
        } catch ( SwingGLException e ) {
            System.out.println (e.toString());
        }
    }
}
```

Volendo ulteriormente commentare possiamo dire che in questa classe vi è una validazione dei dati immessi dall'utente, e la creazione di un internalframe (Libretto) contenente un numero di materie variabile a seconda del valore immesso dall'utente. I nomi dei textfield utilizzati sono dunque stringhe formate anche da un valore numerico generato a runtime.
L'utima classe da descrivere è la CalcolaVotoClick che consente di calcolare il voto. Questa classe si caratterizza per il fatto di fare una validazione dei dati immessi nel libretto mentre il voto di laurea possibile viene ad essere visualizzato sotto forma di message dialog. Il costruttore della classe per semplicità prevede il passaggio del numero di materie, e del booleano che indica se si tratta di laurea specialistica o triennale.

```
package demovoto;

/**
 *
 * @author Giovanni Alestra
 */

import swinggl.*;

public class CalcolaVotoClick implements SwingGLEvent {

    SwingGL swgl;
        //Riferimento all'internalframe Libretto

    int numeromaterie;
    boolean specialistica;
    int vototriennale;
    int cumulatavoti;
    int cumulatacrediti;
        //Valori da inizializzare e campi utili nella computazione

    public CalcolaVotoClick ( SwingGL s, int numeromat, boolean spec ) {

        swgl = s;
        numeromaterie = numeromat;
        specialistica = spec;
            //Inizializzazione dei campi
}

    public void Event ( String ActionCommand ) {

        boolean compilazioneerrata=false;
        //Booleano che indica se ci sono stati errori di compilazione
```

```
        try {
        if ( specialistica ) {
                //Se la laurea è specialistica recupera e verifica il voto triennale
           vototriennale = Integer.parseInt(swgl.GetText("Voto"));
           if ( vototriennale < 60 || vototriennale > 110) compilazioneerrata=true;
        }

        cumulatavoti=0;
        cumulatacrediti=0;


        for ( int i=0; i < numeromaterie; i++) {
           int crediti = Integer.parseInt(swgl.GetText("Crediti"+i));
           int voto = Integer.parseInt(swgl.GetText ("Voto"+i));
           cumulatavoti += crediti*voto;
           cumulatacrediti += crediti;
        }
```

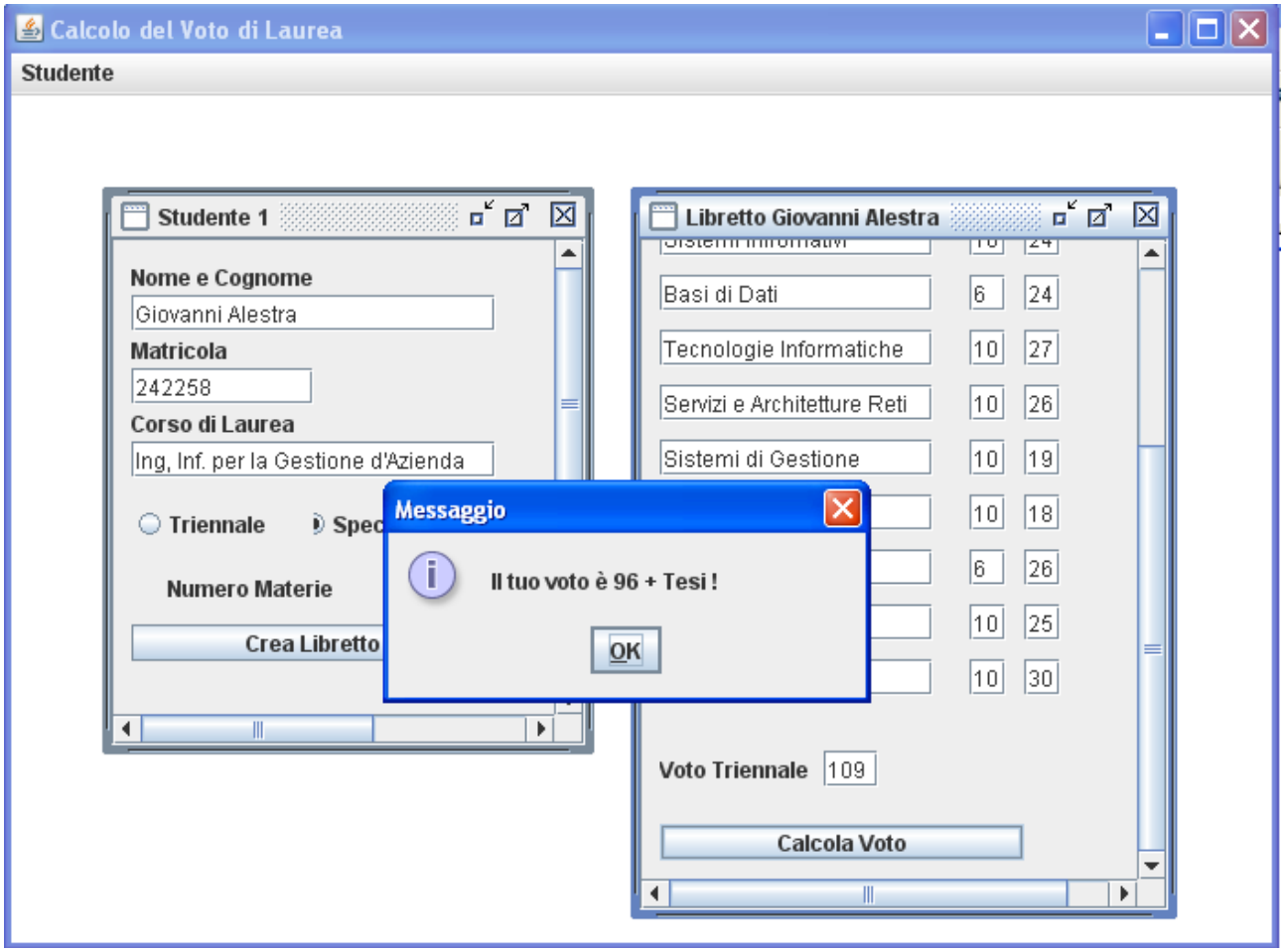
//Recupero e verifica dei dati immessi come numero di crediti e voto delle materie che se corretti vengono ad incrementare la comulatavoti e la cumulatacrediti

```
        int voto = cumulatavoti*110/cumulatacrediti/30;
        //Semplice proporzione per il calcolo del voto di laurea

         if ( specialistica ) {
                //Se di laurea specialistica si considera per 1/3 anche il voto triennale
           voto = vototriennale/3+2*voto/3;
        }

        if ( !compilazioneerrata ) { swgl.ShowMessageDialog("Il tuo voto è " + voto + " + Tesi !");}
                //Visualizzazione del voto di laurea

      } catch ( Exception e ) {
        compilazioneerrata = true;
```

//Se ci sono stati errori di compilazione e sono stati generati eventi nel momento del Parsing dei valori interi visualizza un messaggio “Compilazione Libretto Errata”

```
        }

      if ( compilazioneerrata ) {
        swgl.ShowMessageDialog("Compilare il Libretto Correttamente!");
        return;
      }
   }
}
```

Mostriamo adesso l'applicazione che abbiamo creato.


Calcolo del Voto di Laurea
Studente
Studente 1
Nome e Cognome
Giovanni Alestra
Matricola
242258
Corso di Laurea
Ing, Inf. per la Gestione d'Azienda
Triennale
Numero Materie
Crea Libretto
Messaggio
Il tuo voto è 96 + Tesi !
OK
Libretto Giovanni Alestra
Basi di Dati 6 24
Tecnologie Informatiche 10 27
Servizi e Architetture Reti 10 26
Sistemi di Gestione 10 19
10 18
6 26
10 25
10 30
Voto Triennale 109
Calcola Voto

Capitolo 6

Conclusioni

6.1 Un Obiettivo per la Didattica

Ritengo che siano stati raggiunti tutti gli obiettivi previsti per la didattica, come la familiarizzazione con i componenti delle interfacce grafiche e la possibilità di creare e gestire interfacce grafiche anche con architetture più complesse come gli Internal Frames. La gestione degli eventi permette di comprendere come il flusso delle istruzioni in una applicazione con interfacce grafiche è dettato dalla sequenza degli eventi attivata dall'utente, La libreria è molto intuitiva nel suo utilizzo e semplice, si può introdurre e spiegare in poco tempo ed è comprensibile dagli studenti solamente dalla lettura del manuale, dal JavaDoc e dall'analisi di qualche esempio pratico. Essa consente anche di approfondire il package grafico Swing, attraverso l'utilizzo della funzione getObject che è stata resa pubblica. Consente di far acquisire familiarità con le Interfacce del linguaggio java in maniera semplice, ed insegna anche a gestire le eccezioni.
In ultima analisi penso che questa libreria posso essere utilizzata per scopi didattici e consente di applicare praticamente le conoscenze teoriche necessarie per lo sviluppo di GUI in linguaggio Java.

Bibliografia

1) James Gosling, Bill Joy, Guy Steele : The Java Language Specification. Addison Wesley 1996
2) Robert Eckstein, Marc Loy, Dave Wood : Java Swing. O'reilly 1998
3) Andrea Domenici : Appunti per le lezioni di Ingegneria dei Sistemi Software 2008
4) http://download.oracle.com/javase/1.4.2/docs/api/javax/swing/
5) http://download.oracle.com/javase/tutorial/uiswing/

Libreria Grafica SwingGL

MANUALE D'USO

Autore : Giovanni Alestra

Introduzione

La libreria grafica SwingGL permette la creazione di interfacce grafiche in linguaggio Java. L'approccio della libreria rispetto a quelle utilizzate in ambito professionale è un approccio didattico all'argomento. Si vuole consentire facilmente di creare applicazioni desktop con interfacce grafiche attraverso una più semplice gestione dei componenti grafici e degli eventi cui questi componenti sono soggetti nell'interazione con l'utente. La libreria è composta da classi che estendono le classi dei componenti grafici del package grafico Swing, da classi di supporto e da una classe principale. Per creare una interfaccia grafica non è necessario conoscere tutte le classi che compongono la libreria ma è necessario conoscere solamente l'utilizzo delle funzioni che la classe principale, la classe SwingGL, mette a disposizione. Per la gestione degli eventi è necessario avere il concetto di Interfaccia tipico del linguaggio Java. La gestione degli eventi risulta essere semplificata.
La seguente guida ha lo scopo di rendere abile lo sviluppatore nella creazione di interfacce grafiche per applicazioni desktop anche se questi non ha un background sull'argomento.

INDICE

Aggiungere la Libreria al tuo Progetto

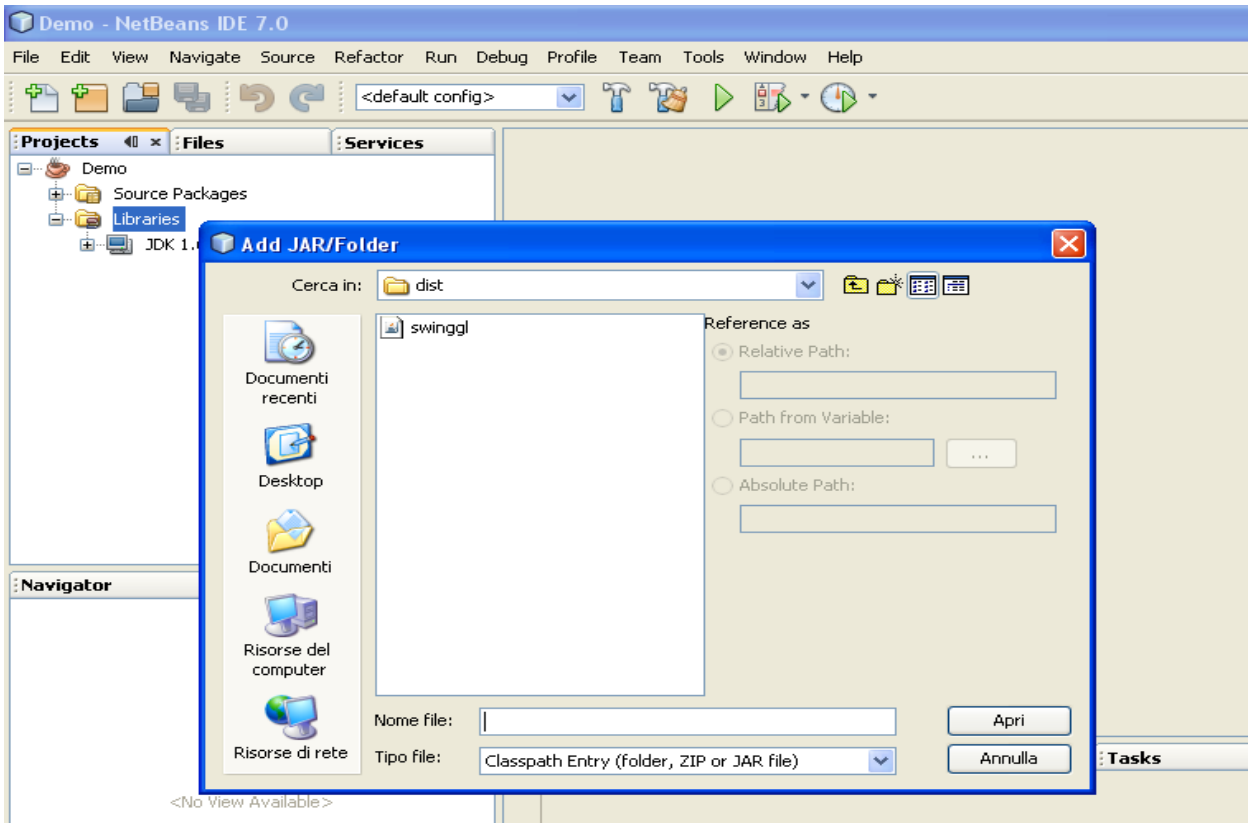
Per aggiungere la libreria al tuo progetto se si stà sviluppando in un ambiente di sviluppo quale Netbeans o Eclipse è necessario seguire i seguenti passi.

- Procurarsi il file “swinggl.jar”

Per prima cosa dobbiamo disporre del file swinggl.jar che contiene tutte le classi della libreria. A questo punto bisogna aggiungere questo file alle librerie.

Aggiungere la Libreria in Netbeans

Se si stà utilizzando netbeans, bisogna cliccare sulla cartella “Libraries” del pannello progetto e successivamente cliccare su “Add Jar/Folder”.

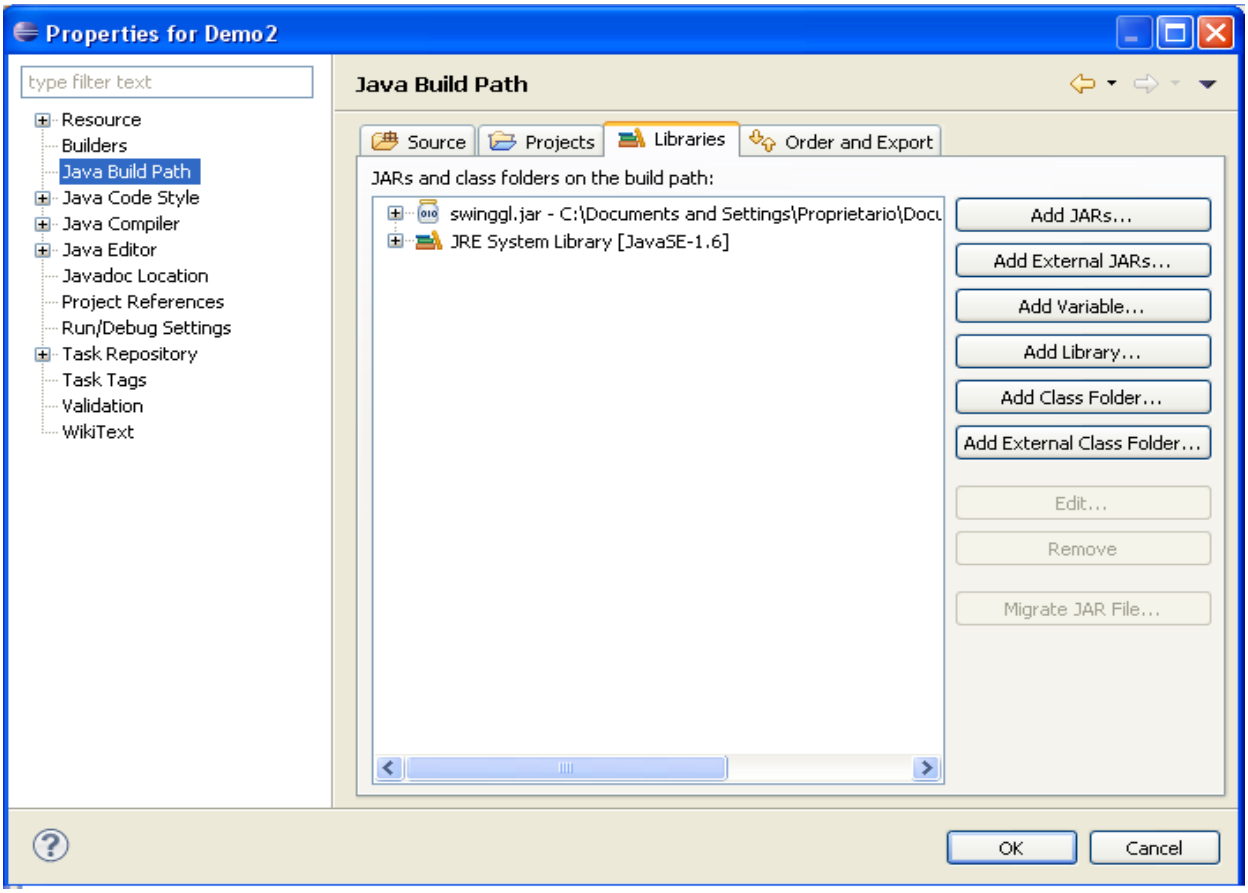


E caricare il file tra le librerie. Dopo questa operazione sarà possibile prima della definizione delle classi del nostro progetto importare il package “swinggl” tramite l'operazione :

import swinggl.*

Aggiungere la Libreria in Eclipse

Attraverso un procedimento del tutto simile a quello visto è possibile aggiungere la libreria nell'ambiente di sviluppo Eclipse. Se disponiamo del file “swinggl.jar” dobbiamo selezionare “properties” dal menù “project”, cliccare su “Java Build Path” e successivamente su “Add External JARs”.



A questo punto prima della definizione delle classi sarà possibile importare il package con :

import swinggl.*;

Creare una Interfaccia Grafica

Per creare una interfaccia grafica basta istanziare un oggetto della classe principale, la classe SwingGL scrivendo all'interno di una funzione :

SwingGL swgl = new SwingGL();

ESEMPIO :

```
import swinggl.*;

public class Demo {
```

```
   public static void main(String [ ] args) {
     SwingGL swgl = new SwingGL();
   }

}
```

A questo punto tramite l'oggetto di tipo SwingGL e le sue funzioni saremo in grado di creare una interfaccia grafica.

Creare una Finestra

Per creare una finestra è possibile utilizzare la funzione della classe principale **SwingGL.newWindow**, esplicitando il nome (una Stringa) che vogliamo dare alla finestra per poterci riferire ad essa in futuro, ed anche il titolo, la posizione, la larghezza e l'altezza.

newWindow

```
public void newWindow(java.lang.String name,
                      java.lang.String title,
                      int x,
                      int y,
                      int width,
                      int height)
```

Permette la creazione e visualizzazione di una finestra

Parameters:

`name` - Stringa contenente il nome che si vuole assegnare alla finestra
`title` - Stringa che indica il titolo che verrà visualizzato nella finestra
`x` - Posizione X nello schermo
`y` - Posizione Y nello schermo
`width` - Larghezza della finestra
`heightAltezza` - della finestra

ESEMPIO:

```
import swinggl.*;

public class Demo {

   public static void main(String [ ] args) {
     SwingGL swgl = new SwingGL();
     try {
      swgl.newWindow("Main", "Hello World!", 100, 100, 200, 200);
        } catch ( SwingGLException e ) { System.out.println ( e.toString() ); }
   }
 }
```

RISULTATO ESEMPIO:

Aggiungere un Panello ad una Finestra

Dopo aver creato una finestra se vogliamo in essa visualizzare dei componenti dobbiamo aggiungere un pannello. Per far ciò possiamo utilizzare la funzione **SwingGL.newPanel**, la quale prevede di esplicitare : il nome (una Stringa) che si vuole dare al pannello, il nome del contenitore (tipicamente il nome della finestra), la larghezza del pannello e l'altezza. Se il pannello ha dimensioni maggiori di quelle della finestra verranno visualizzate in modo automatico dell scrollbars.

newPanel

```
public void newPanel(java.lang.String name,
                     java.lang.String container,
                     int width,
                     int height)
```

Permette di visualizzare un pannello sul quale creare altri oggetti grafici

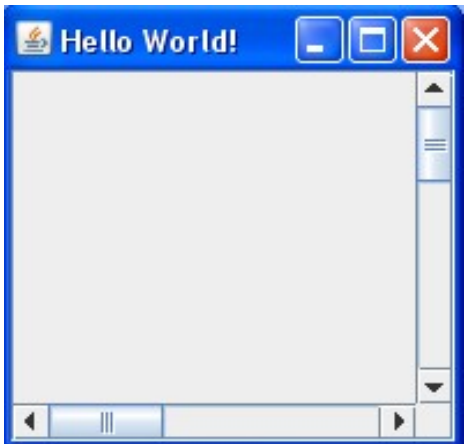
Parameters:

`name` - Stringa che contine il nome che si vuole assegnare al pannello
`container` - Contenitore del pannello tipicamente una finestra
`width` - Esprime la Larghezza del Pannello in pixel
`height` - Esprime l'Altezza del Pannello in pixel

ESEMPIO :

```
import swinggl.*;

public class Demo {
  [..]
 try {
     swgl.newWindow("Main", "Hello World!", 100, 100, 200, 200);
     swgl.newPanel("Panel", "Main", 500, 500);
   } catch ( SwingGLException e ) { System.out.println ( e.toString() ); };
 }
```

RISULTATO ESEMPIO:



Aggiungere una Etichetta ad un Pannello

Dopo aver creato un pannello possiamo visualizzare in esso molti componenti grafici. Ora trattiamo la visualizzazione delle etichette. Per visualizzare una etichetta possiamo utilizzare la funzione **SwingGL.newLabel**, la quale prevede di esplicitare : il nome (una Stringa) che si vuole dare al componente, il nome del contenitore (tipicamente il nome di un pannello), il testo, la posizione x ed y, la larghezza del pannello e l'altezza.

newLabel

```
public void newLabel(java.lang.String name,
                     java.lang.String container,
                     java.lang.String text,
                     int x,
                     int y,
                     int width,
                     int height)
```

Cosente la visualizzazione di una etichetta ed il suo posizionamento relativo al contenitore

Parameters:

`name` - Stringa contenente il nome che si vuole assegnare all'etichetta
`container` - Stringa che indica il nome del contenitore tipicamente un pannello
`text` - Stringa Testo dell'etichetta
`x` - Posizione X relativa al contenitore
`y` - Posizione Y relativa la contenitore
`width` - Larghezza dell'etichetta
`height` - Altezza dell'etichetta

ESEMPIO :

[...]

```
swgl.newPanel("Panel", "Main",500,500);
swgl.newLabel("Label", "Panel", "Hello world!", 10, 10, 200, 20);
```

[...]

RISULTATO ESEMPIO:

Aggiungere un TextField ad un Pannello

I text field sono campi nei quali si può scrivere del testo. Per inserirli possiamo usare la funzione **SwingGL.newTextField** esplicitando il nome da dare al componente, il contenitore tipicamente un pannello, il testo di default, la posizione x ed y, la larghezza e l'altezza.

newTextField

```
public void newTextField(java.lang.String name,
                         java.lang.String container,
                         java.lang.String text,
                         int x,
                         int y,
                         int width,
                         int height)
```

Permette la visualizzazione di un Text Field

Parameters:

`name` - Nome da dare al componente
`container` - Nome del contenitore tipicamente un pannello
`text` - Testo di default
`x` - Posizione X
`y` - Posizione Y
`width` - Larghezza del componente
`height` - Altezza del Componente

ESEMPIO :

[...]

swgl.newPanel("Panel", "Main",500,500);

swgl.newTextField("TextField", "Panel", "Hello world!", 10, 10, 200, 20);

[...]

RISULTATO ESEMPIO :

Aggiungere una Text Area ad un Pannello

E' possibile aggiungere una Text Area ad un pannello attraversa la funzione **SwingGL.newTextArea** esplicitando il nome che si vuole dare al componente, il nome del contenitore tipicamente un pannello, il testo di default, la posizione x ed y, la larghezza e l'altezza del componente.

newTextArea

```
public void newTextArea(java.lang.String name,
                        java.lang.String container,
                        java.lang.String text,
                        int x,
                        int y,
                        int width,
                        int height)
```

Permette la visualizzazione di una Text Area

Parameters:

`name` - Nome da dare al componente
`container` - Nome del contenitore tipicamente un pannello
`text` - Testo di default
`x` - Posizione X
`y` - Posizione Y
`width` - Larghezza del componente
`height` - Altezza del Componente

ESEMPIO :

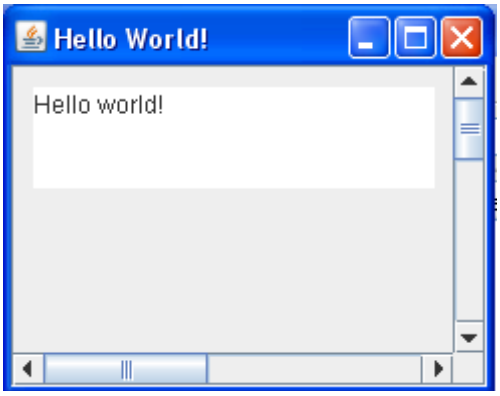
[...]

swgl.newPanel("Panel", "Main",500,500);

swgl.newTextArea ("TextArea", "Panel", "Hello world!", 10, 10, 200, 50);

[…]

RISULTATO ESEMPIO :



Aggiungere un Campo Password ad un Pannello

Per aggiungere un campo Password si può utilizzare la funzione **SwingGL.newPasswordField**, esplicitando il nome del componente, il nome del contenitore tipicamente un pannello, la posizione x ed y, la larghezza e l'altezza del componente.

newPasswordField

```
public void newPasswordField(java.lang.String name,
                             java.lang.String container,
                             int x,
                             int y,
                             int width,
                             int height)
```

Permette la visualizzazione di un campo Password

Parameters:

`name` - Nome da dare al componente
`container` - Nome del contenitore tipicamente un pannello
`x` - Posizione x
`y` - Posizione y
`width` - Larghezza del componente
`height` - Altezza del componente

ESEMPIO :

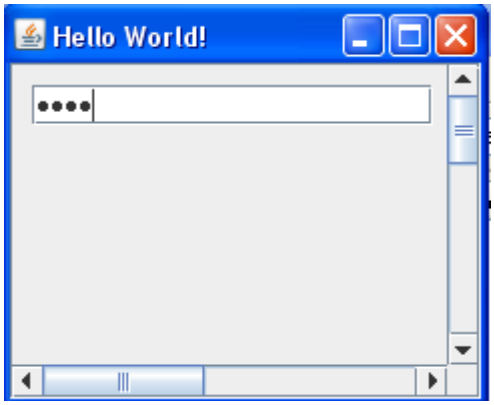
[...]

swgl.newPanel("Panel", "Main",500,500);

swgl.newPasswordField ("Password", "Panel", 10, 10, 200, 20);

[…]

RISULTATO ESEMPIO :



Aggiungere un ComboBox ad un Pannello

Per aggiungere un combobox ad un pannello possiamo utilizzare la funzione **SwingGL.newComboBox**, esplicitando il nome del componente, il nome del contenitore tipicamente un pannello, un vettore di stringhe che rappresentano i valori selezionabili, la posizione x ed y, la larghezza e l'altezza del componente.

newComboBox

```
public void newComboBox(java.lang.String name,
                        java.lang.String container,
                        java.lang.String[] items,
                        int x,
                        int y,
                        int width,
                        int height)
```

Permette la visualizzazione di un ComboBox

Parameters:

`name` - Nome da dare al Componente
`container` - Nome del contenitore tipicamente un pannello
`items` - Lista dei valori selezionabili
`x` - Posizione x
`y` - Posizione y
`width` - Larghezza del componente

`height` - Altezza del componente

ESEMPIO :

[…]

```
swgl.newPanel("Panel", "Main",500,500);
String[] Items = { "Bird", "Cat", "Dog", "Rabbit", "Pig" };
swgl.newComboBox("ComboBox", "Panel", Items, 10, 10, 100, 20);
```

[…]

RISULTATO ESEMPIO :

Aggiungere un CheckBox ad un Pannello

Per aggiungere un checkbox possiamo utilizzare la funzione **SwingGL.newCheckBox** esplicitando il nome del componente, il nome del contenitore tipicamente un pannello, il testo inerente alla selezione, la posizione x ed y, la larghezza, l'altezza ed il valore di default.

newCheckBox

```
public void newCheckBox(java.lang.String name,
                        java.lang.String container,
                        java.lang.String text,
                        int x,
                        int y,
                        int width,
                        int height,
                        boolean selected)
```

Permette di visualizzare un CheckBox

Parameters:

`name` - Nome del componente
`container` - Nome del contenitore tipicamente un pannello
`text` - Testo inerente alla selezione
`x` - Posizione x
`y` - Posizione y

`width` - Larghezza del componente
`height` - Altezza del componente
`selected` - Selezionato vero/falso

ESEMPIO:

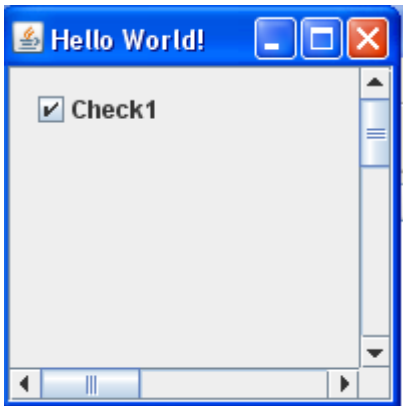
[…]

swgl.newPanel("Panel", "Main",500,500);

swgl.newCheckBox("CheckBox","Panel", "Check1", 10, 10, 100, 20, true);

[…]

RISULTATO ESEMPIO :



Aggiungere dei RadioButton ad un Pannello

Prima di aggiungere dei radiobutton dobbiamo creare un button group mediante la funzione **SwingGL.newButtonGroup :**

newButtonGroup

```
public void newButtonGroup(java.lang.String name)
```

Successivamente possiamo creare i radio button mediante la funzione :

newRadioButton

```
public void newRadioButton(java.lang.String name,
                           java.lang.String container,
                           java.lang.String Group,
                           java.lang.String text,
                           int x,
                           int y,
                           int width,
                           int height,
                           boolean selected)
```

Permette la visualizzazione di un radio button associato ad un button group

Parameters:

`name` - Nome del componenete
`container` - Nome del contenitore tipicamente un pannello
`Group` - Nome del button group
`text` - Testo inerente alla selezione
`x` - Posizione x
`y` - Posizione y
`width` - Larghezza del componente
`height` - Altezza del componente
`selected` - Selezionato Vero/Falso

ESEMPIO E RISULTATO:

swgl.newPanel("Panel", "Main",500,500);

swgl.newButtonGroup("BG");

swgl.newRadioButton("RButton", "Panel", "BG","Uno", 10, 10, 80, 20, true);

swgl.newRadioButton("RButton2", "Panel", "BG","Due", 10, 30, 80, 20, false);

Aggiungere un TabbedPane ad un Pannello

Per aggiungere un TabbedPane possiamo utilizzare la funzione **SwingGL.newTabbedPane**, esplicitando il nome che si vuole dare al tabbedpane, il nome del contenitore tipicamente un pannello, la posizione x ed y, la larghezza e l'altezza del tabbedpane.

newTabbedPane

```
public void newTabbedPane(java.lang.String name,
                          java.lang.String container,
                          int x,
                          int y,
                          int width,
                          int height)
```

Permette la visualizzazione di un TabbedPane

Parameters:

`name` - Nome da dare al componente
`container` - Nome del contenitore tipicamente un pannello
`x` - Posizione x

`y` - Posizione y
`width` - Larghezza del componente
`height` - Altezza del componente

ESEMPIO :

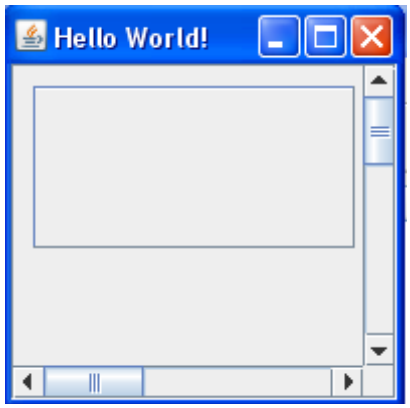
[...]

swgl.newPanel("Panel", "Main",500,500);

swgl.newTabbedPane("TPane", "Panel",10, 10, 160, 80);

[…]

RISULTATO ESEMPIO:



Aggiungere una Tab ad un TabbedPane

Per aggiungere una o più tab ad un tabbedpane possiamo utilizzare la funzione **SwingGL.addTabToTabbedPane**, esplicitando il nome del tabbedpane, il nome da dare al nuovo pannello che si stà creando ed il titolo del pannello.

addTabToTabbedPane

```
public void addTabToTabbedPane(java.lang.String tabbedpane,
                               java.lang.String name,
                               java.lang.String title)
```

Permette di aggiugere un pannello ad un TabbedPane

Parameters:

`tabbedpane` - Nome del TabbedPane
`name` - Nome del pannello che si vuole aggiungere
`title` - Titolo del pannello

ESEMPIO:

[…]

```
swgl.newPanel("Panel", "Main",500,500);
swgl.newTabbedPane("TPane", "Panel",10, 10, 160, 80);
swgl.addTabToTabbedPane("TPane", "LibrettoTab", "Libretto");
swgl.addTabToTabbedPane("TPane", "TesiTab", "Tesi");
```

[…]

RISULTATO ESEMPIO:

Aggiungere un Bottone ad un Pannello

Per aggiungere un bottone ad un pannello possiamo utilizzare la funzione **SwingGL.newButton**, esplicitando il nome da dare al componente, il nome del contenitore tipicamente un pannello, il testo del bottone, la posizione x ed y, la larghezza, l'altezza e la classe gestore dell'evento click. Se non è previsto un gestore dell'evento scriveremo **null**.

newButton

```
public void newButton(java.lang.String name,
                      java.lang.String container,
                      java.lang.String text,
                      int x,
                      int y,
                      int width,
                      int height,
                      SwingGLEvent e)
```

Consente la creazione e visualizzazione di un Bottone su cui si può cliccare

Parameters:

`name` - Stringa contenente il nome che si vuole assegnare al Bottone
`container` - Stringa contente il nome del contenitore tipicamente un pannello
`text` - Testo del Bottone
`x` - Posizione X relativa al contenitore
`y` - Posizione Y relativa al contenitore
`width` - Larghezza del Bottone
`height` - Altezza del bottone
`e` - Classe che implementa SwingGLEvent per la gestione dell'evento Click settabile su null se non si vuole gestire l'evento

ESEMPIO :

[…]

```
swgl.newPanel("Panel", "Main",500,500);
swgl.newButton("Button", "Panel", "OK", 10, 10, 30, 20, null);
```

[…]

RISULTATO ESEMPIO :

Scrivere un Gestore dell'Evento Click per un Bottone

Per scrivere un gestore dell'evento click per un bottone è necessario scrivere una classe che implementi la classe **SwingGLEvent,** istanziare un oggetto di tale classe e passarlo alla funzione **SwingGL.newButton**. Un semplice gestore dell'evento click potrebbe stampare sul prompt dei comandi una stringa. Ecco un esempio :

```
public class Click implements SwingGLEvent {
   public void Event ( String ActionCommand ) {
      System.out.println ( "Action" );
   }
}
```

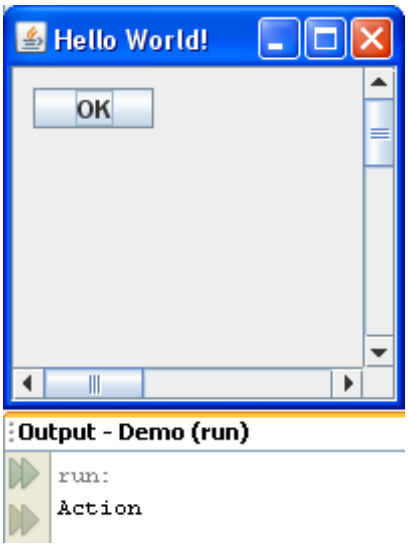
E scrivere nella funzione che richiama **newButton :**

[...]

```
swgl.newPanel("Panel", "Main",500,500);
Click click = new Click();
swgl.newButton("Button", "Panel", "OK", 10, 10, 60, 20, click);
```

[…]

RISULTATO ESEMPIO :



Consiglio Operativo

Al fine di poter avere accesso all'interfaccia grafica quando si gestisce un evento è opportuno che la classe gestore dell'evento abbia un riferimento all'oggetto **SwingGL** e che il riferimento a questo oggetto venga settato dal costruttore della classe.

ESEMPIO :

```
import swinggl.*;
public class Click implements SwingGLEvent {
   SwingGL swgl;
```

```
   public Click (SwingGL swinggl ) {
      swgl = swinggl;
   }

   public void Event ( String ActionCommand ) {
      System.out.println ( "Action" );
      try {
         // swgl. ...
         } catch ( SwingGLException e ) { System.out.println ( e.toString() ); };
}
}
```

E nella funzione che crea l'oggetto :

```
SwingGL swgl = new SwingGL();
Click click = new Click(swgl);
```

In questo modo possiamo avere accesso all'interfaccia durante la gestione degli eventi.

Aggiungere i Menù ad una Finestra

Per aggiungere una menù bar ad una finestra, dobbiamo utilizzare la funzione **SwingGL.AddMenuBar** esplicitando il nome del componente ed il nome della finestra.

addMenuBarToWindow

```
public void addMenuBarToWindow(java.lang.String name,
                               java.lang.String window)
```

Aggiunge una menù bar ad una finestra

Parameters:

`name` - Nome da dare al componente
`window` - Nome della finestra

Per aggiungere i menù ad una menùbar dobbiamo utilizzare la funzione **SwingGL.AddMenuToWindow** esplicitando il nome della menùbar, il nome del menù, i campi visualizzati del menù ed il gestore dell'evento click.

addMenuToWindow

```
public void addMenuToWindow(java.lang.String menubar,
                            java.lang.String name,
                            java.lang.String[] menuItems,
                            SwingGLEvent e)
```

Aggiunge un menù alla finestra

Parameters:

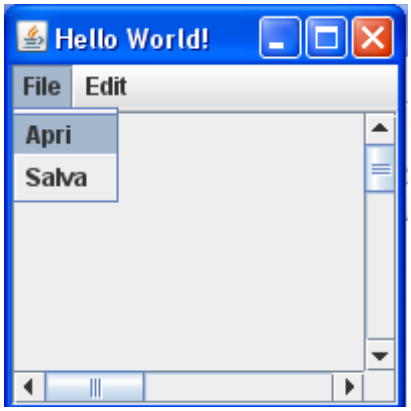
`menubar` - Nome della menùbar
`name` - Nome del menu che si stà aggiungendo
`menuItems` - Campi del menù
`e` - Gestore dell'evento click sul menù

ESEMPIO :

[..]

```
swgl.newWindow("Main", "Hello World!", 100, 100, 200, 200);
Click click = new Click(swgl);
String[] Items = { "Apri", "Salva" };
String[] Items2 = { "Copia" };
swgl.addMenuBarToWindow("Menu", "Main");
swgl.addMenuToWindow("Menu", "File", Items, click );
swgl.addMenuToWindow("Menu", "Edit", Items2, click );
```

RISULTATO ESEMPIO :



Per la gestione degli eventi, nella funzione della classe che implementa **SwingGLEvent** bisogna con un **if** seguito da un **equals**, gestire l'evento relativo al menù che è stato realmente cliccato, analizzando la stringa **ActionCommand** che ritorna il testo del menuItem che è stato cliccato.

```
import swinggl.*;

public class MenuClick implements SwingGLEvent {

   SwingGL swgl;

   public MenuClick (SwingGL swinggl ) {
      swgl = swinggl;
   }

   public void Event ( String ActionCommand ) {
     if (ActionCommand.equals("Apri")) { /* try … swgl. … catch */ };
   }
}
```

Visualizzare Open/Save File Dialog

Per visualizzare una openfiledialog e per visualizzare una savefiledialog è possibile utilizzare le funzioni : **SwingGL.ShowOpenFileDialog** e la funzione **SwingGL.ShowSaveFileDialog**.

ShowOpenFileDialog

```
public java.lang.String ShowOpenFileDialog(java.lang.String title)
```

Visualizza una OpenFileDialog

Parameters:
`title` - Titolo della finestra
Returns:
String Nome del file selezionato

ShowSaveFileDialog

```
public java.lang.String ShowSaveFileDialog(java.lang.String title)
```

Visualizza una FileSaveDialog

Parameters:
`title` - Titolo della finestra di dialogo
Returns:
String Nome del file sul quale salvare salvare

ESEMPIO :

[…]

```
String file = swgl.ShowOpenFileDialog("Apri");
System.out.println (file);
String file2 = swgl.ShowSaveFileDialog("Salva");
System.out.println (file2);
```

[…]

Visualizzare Message ed Input Dialogs

Per visualizzare un messaggio possiamo utilizzare la funzione **SwingGL.ShowMessageDialog**, passando alla funzione il messaggio da visualizzare.

ShowMessageDialog

```
public void ShowMessageDialog(java.lang.String message)
```

Permette la visualizzazione di un messaggio

Parameters:
`message` - Messaggio da visualizzare

Per visualizzare una Input Dialog possiamo utilizzare la funzione **SwingGL.ShowInputDialog**, fornendo la domanda cui l'utente deve rispondere.

ShowInputDialog

```
public java.lang.String ShowInputDialog(java.lang.String message)
```

Permette la visualizzazione di una finestra di input

Parameters:

`message` - Messaggio da visualizzare

Returns:

Stringa di risposta da parte dell'utente

Ottenere il testo di un TextField o TextArea

Per ottenere il testo di una text field o text area tipicamente durante la gestione di un evento possiamo utilizzare la funzione **SwingGL.GetText.**

GetText

public java.lang.String **GetText**(java.lang.String name)

Applicabile a TextField e TextArea ritorna una stringa contenente il testo digitato sul componente

Parameters:

name - Nome del Componente

Returns:

Testo Digitato

Ottenere il testo selezionato in un ComboBox

Per ottenere il testo selezionato in un combo box tipicamente durante la gestione di un evento possiamo utilizzare la funzione **SwingGL.GetSelection**.

GetSelection

public java.lang.String **GetSelection**(java.lang.String name)

Applicabile a ComboBox ritorna il testo selezionato

Parameters:

name - Nome del Componenete

Returns:

Testo Selezionato

Verificare se una CheckBox o un Radio Button è stato Selezionato

Per verificare se una CheckBox o un Radio Button è stato selezionato possiamo utilizzare la funzione **SwingGL.GetSelected**.

GetSelected

public boolean **GetSelected**(java.lang.String name)

Applicabile a CheckBox e RadioButton ritorna se l'elemento è stato selezionato

Parameters:
name - Nome del Componente
Returns:
Booleano Vero/Falso

Settare il testo di un textfield o textarea

Per settare il testo ti un text field o text area possiamo utilizzare la funzione **SwingGL.SetText.**

SetText

public void **SetText**(java.lang.String name,
java.lang.String text)

Applicabile a TextField e TextArea setta il testo del componente

Parameters:
name - Nome del componente
text - testo da settare

Settare la selezione di un combobox

Per settare il la selezione di un combo box possiamo utilizzare la funzione **SwingGL.SetSelection**.

SetSelection

public void **SetSelection**(java.lang.String name,
int index)

Applicabile a ComboBox setta il testo della possibile selezione

Parameters:
name - Nome del componente
index - Intero che indica la posizione del testo da settare nel vettore String[] Items

Settare la selezione di una CheckBox o Radio Button

Per settare il la selezione di una CheckBox o di un Radio Button possiamo utilizzare la funzione **SwingGL.SetSelected.**

SetSelected

public void **SetSelected**(java.lang.String name,
boolean value)

Applicabile a CheckBox e RadioButton setta la selezione

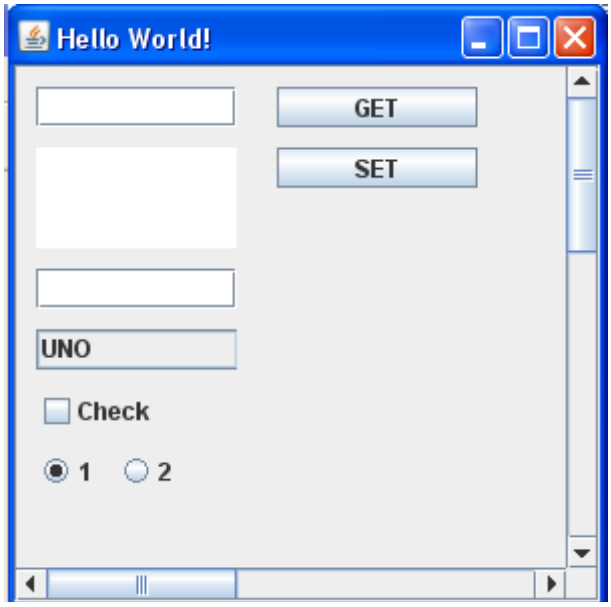
Parameters:
name - Nome del Componente
value - Booleano Vero/Falso

Esempio Completo Funzioni di Get e Set

```
import swinggl.*;

public class Demo {

   public static void main(String [ ] args) {
     SwingGL swgl = new SwingGL();
    try {
     swgl.newWindow("Main", "Hello World!", 100, 100, 300, 300);
     swgl.newPanel("Panel", "Main",700,700);

     swgl.newTextField("TextField", "Panel", "", 10, 10, 100, 20 );
     swgl.newTextArea("TextArea", "Panel", "", 10, 40, 100, 50);
```

```
      swgl.newPasswordField("Password", "Panel", 10, 100, 100, 20);
      String[] Items = { "UNO" , "DUE" };
      swgl.newComboBox("Combo", "Panel", Items, 10, 130, 100, 20);
      swgl.newCheckBox("Check", "Panel", "Check", 10, 160, 100, 20, false);
      swgl.newButtonGroup("BG");
      swgl.newRadioButton("Radio1", "Panel", "BG", "1", 10, 190, 50, 20, true);
      swgl.newRadioButton("Radio2", "Panel", "BG", "2", 50, 190, 50, 20, false);

      ClickGet Get = new ClickGet (swgl);
      ClickSet Set = new ClickSet (swgl);

      swgl.newButton("GET", "Panel", "GET", 130, 10, 100, 20, Get);
      swgl.newButton("SET", "Panel", "SET", 130, 40, 100, 20, Set);
     } catch ( SwingGLException e ) {
        System.out.println(e.toString());
     }
   }
 }

import swinggl.*;

public class ClickGet implements SwingGLEvent {

   SwingGL swgl;

   public ClickGet (SwingGL swinggl ) {
      swgl = swinggl;
   }

   public void Event ( String ActionCommand ) {
     try {
      String textfield = swgl.GetText("TextField");
      String textarea = swgl.GetText("TextArea");
      char[] password = swgl.GetPassword("Password");
      String combobox = swgl.GetSelection("Combo");
```

```
        boolean checkbox = swgl.GetSelected("Check");
        boolean radiobutton1 = swgl.GetSelected("Radio1");
        boolean radiobutton2 = swgl.GetSelected("Radio2");

        System.out.println (textfield);
        System.out.println (textarea);
        System.out.println (password);
        System.out.println (combobox);
        System.out.println (checkbox);
        System.out.println (radiobutton1);
        System.out.println (radiobutton2);
      } catch ( SwingGLException e ) { System.out.println ( e.toString()); }
    }
}

import swinggl.*;

public class ClickSet implements SwingGLEvent {

    SwingGL swgl;

    public ClickSet (SwingGL swinggl ) {
        swgl = swinggl;
    }

    public void Event ( String ActionCommand ) {
    try {
        swgl.SetText("TextField", "Setted");
        swgl.SetText("TextArea", "Setted");
        swgl.SetSelection("Combo",1);
        swgl.SetSelected("Check", true);
        swgl.SetSelected("Radio2",true);
    } catch ( SwingGLException e ) { System.out.println ( e.toString()); };
    }
}
```



Uso della Funzione GetObject

Al fine di poter sfruttare tutte le potenzialità di Swing, la classe **SwingGL,** mette a disposizione pubblicamente la funzione **SwingGl.GetObject.**

getObject

public java.lang.Object **getObject**(java.lang.String name)

Restituisce un riferimento di tipo Object all'oggetto (componente grafico) a cui ci si riferisce mediante una stringa che è il nome dell'oggetto

Parameters:
name -
Returns:
Object

La funzione permette di ottenere l'oggetto (componente grafico) dal suo nome. Dopo un cast alle classi della libreria, o a quelle di swing è possibile utilizzare tutte le funzioni di Swing sul componente. L'immagine della seguente pagina mostra il concetto. Per la documentazione sulle classi della libreria vedere il JavaDoc.

Aggiungere un Desktop Pane ad una Finestra

Per creare un contenitore di finestre che tipicamente rappresentano documenti bisogna aggiungere un Desktop Pane ad una finestra. Possiamo farlo utilizzando la funzione della classe **SwingGL**:

AddDesktopPaneToWindow

public void **AddDesktopPaneToWindow**(java.lang.String name,
java.lang.String window)
throws SwingGLException

Aggiunge un DeskTop Pane alla Finestra

Parameters:

name - Nome del Desktop Pane
window - Nome della Finestra

Aggiungere un Internal Frame ad un Desktop Pane

Per creare una finestra docuemento, ovvero un internal frame possiamo utilizzare la funzione della classe **SwingGL** :

newInternalFrame

public SwingGL **newInternalFrame**(java.lang.String name,
java.lang.String desktoppane,
java.lang.String title,
int x,
int y,
int width,
int height)
throws SwingGLException

Aggiunge un InternalFrame ad un Desktop Pane

Parameters:

name - Nome dell'Internal Frame
desktoppane - Nome del Desktop Pane
title - Titolo dell'Internal Frame
x - Posizione x
y - Posizione y
width - Larghezza del Frame

height - Altezza del Frame

Returns:

Oggetto di tipo SwingGL

Ottenere l'oggetto SwingGL di un Internal Frame

Per ottenere l'oggetto **SwingGL** di un internal frame possiamo utilizzare la funzione della classe **SwingGL :**

getGraphicInteface

public SwingGL **getGraphicInteface**(java.lang.String internalFrame)
throws SwingGLException

Restituisce l'oggeto SwingGL di un Internal Frame

Parameters:

internalFrame - Nome dell'Internal Frame

Returns:

Per comprendere come utilizzare i Desktop Pane e gli Internal Frame studiare l'esempio qui proposto :

Esempio Funzioni di Get e Set con Internal Frames

```
import swinggl.*;

public class Demo2 {

  /**
   * @param args the command line arguments
   */
  public static void main(String[] args) {

    SwingGL swgl = new SwingGL();
    try {
      swgl.newWindow("Main", "Demo Internal Frames", 100, 100, 500, 400);
      swgl.addMenuBarToWindow("Menu", "Main");
      String [] items = { "Demo" };
```

```
        MenuClick mclick = new MenuClick ( swgl );
        swgl.addMenuToWindow("Menu", "Demo",items, mclick);
        swgl.AddDesktopPaneToWindow("Desktoppane", "Main");

      } catch ( SwingGLException e ) { System.out.println ( e.toString()); }
   }
}

import swinggl.*;

public class MenuClick  implements SwingGLEvent {
   SwingGL swgl;
   int count;

   public MenuClick ( SwingGL swinggl ) {
      swgl = swinggl;
      count = 1;
   }

   @Override
   public void Event ( String ActionCommand ) {
      if (ActionCommand.equals("Demo")) {
         String frameName = "Demo " + count; count++;
         try {
               SwingGL iframe = swgl.newInternalFrame(frameName, "Desktoppane", frameName,
10+count*20, 10+count*20, 300,300);
            iframe.newPanel("Panel", frameName,700,700);
            iframe.newTextField("TextField", "Panel", "", 10, 10, 100, 20 );
            iframe.newTextArea("TextArea", "Panel", "", 10, 40, 100, 50);
            iframe.newPasswordField("Password", "Panel", 10, 100, 100, 20);
            String[] Items = { "UNO" , "DUE" };
            iframe.newComboBox("Combo", "Panel", Items, 10, 130, 100, 20);
            iframe.newCheckBox("Check", "Panel", "Check", 10, 160, 100, 20, false);
            iframe.newButtonGroup("BG");
            iframe.newRadioButton("Radio1", "Panel", "BG", "1", 10, 190, 50, 20, true);
```

```
            iframe.newRadioButton("Radio2", "Panel", "BG", "2", 50, 190, 50, 20, false);

            ClickGet Get = new ClickGet (iframe);
            ClickSet Set = new ClickSet (iframe);

            iframe.newButton("GET", "Panel", "GET", 130, 10, 100, 20, Get);
            iframe.newButton("SET", "Panel", "SET", 130, 40, 100, 20, Set);
          } catch ( SwingGLException e ) { System.out.println (e.toString()); };
       }
    }
}

import swinggl.*;

public class ClickGet implements SwingGLEvent {

   SwingGL swgl;

   public ClickGet (SwingGL swinggl ) {
      swgl = swinggl;
   }

   public void Event ( String ActionCommand ) {
     try {
      String textfield = swgl.GetText("TextField");
      String textarea = swgl.GetText("TextArea");
      char[] password = swgl.GetPassword("Password");
      String combobox = swgl.GetSelection("Combo");
      boolean checkbox = swgl.GetSelected("Check");
      boolean radiobutton1 = swgl.GetSelected("Radio1");
      boolean radiobutton2 = swgl.GetSelected("Radio2");

      System.out.println (textfield);
      System.out.println (textarea);
      System.out.println (password);
```

```
        System.out.println (combobox);
        System.out.println (checkbox);
        System.out.println (radiobutton1);
        System.out.println (radiobutton2);
       } catch ( SwingGLException e ) { System.out.println ( e.toString()); }
    }
}

import swinggl.*;

public class ClickSet implements SwingGLEvent {

    SwingGL swgl;

    public ClickSet (SwingGL swinggl ) {
        swgl = swinggl;
    }

    public void Event ( String ActionCommand ) {
    try {
        swgl.SetText("TextField", "Setted");
        swgl.SetText("TextArea", "Setted");
        swgl.SetSelection("Combo",1);
        swgl.SetSelected("Check", true);
        swgl.SetSelected("Radio2",true);
    } catch ( SwingGLException e ) { System.out.println ( e.toString()); };
    }
}
```

RISULTATO

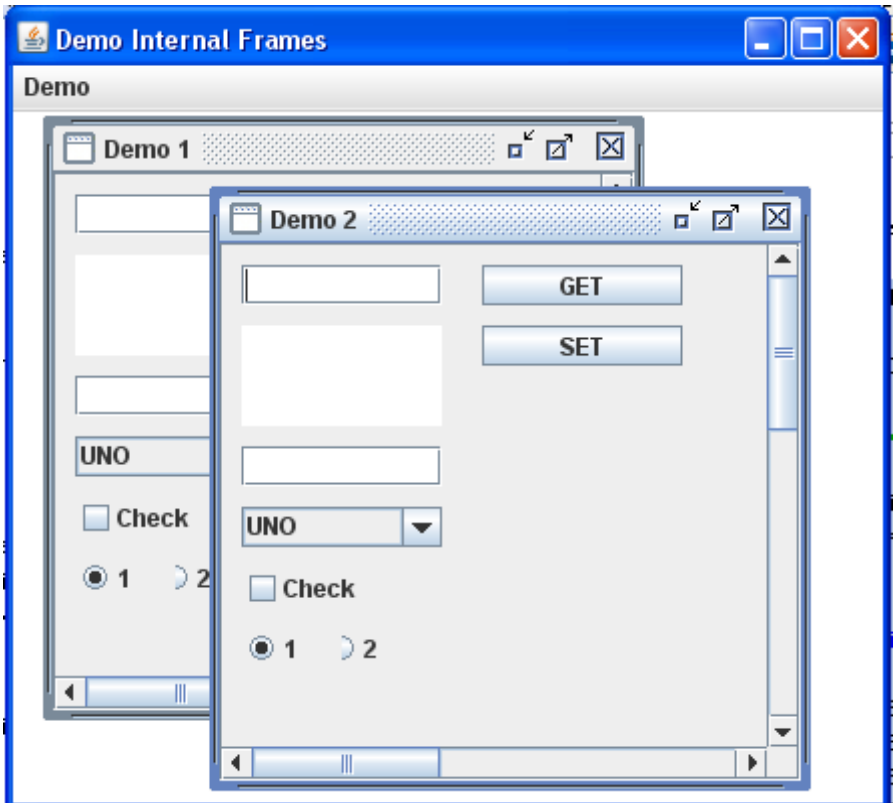

Demo Internal Frames
Demo
Demo 1
UNO
Check
1
2
Demo 2
GET
SET
UNO
Check
1
2

TABELLA CODICI ED ERRORI ECCEZIONI

Codice	Tipo Errore
1	Nome Oggetto Esistente e Non Univoco
2	Nome Oggetto Errato
3	Get da un Oggetto di tipo Errato
4	Set su di un Oggetto di tipo Errato