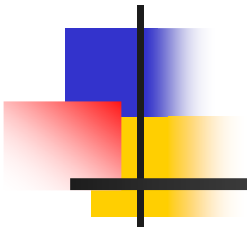
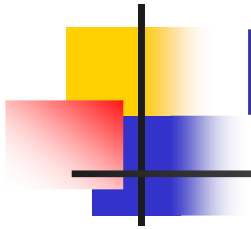


Librerie Grafiche in Java: Flexdock e SwingX

Laboratorio di Programmazione

Tutor Ing. Massimo Guarascio
Docente Ing. Riccardo Ortale



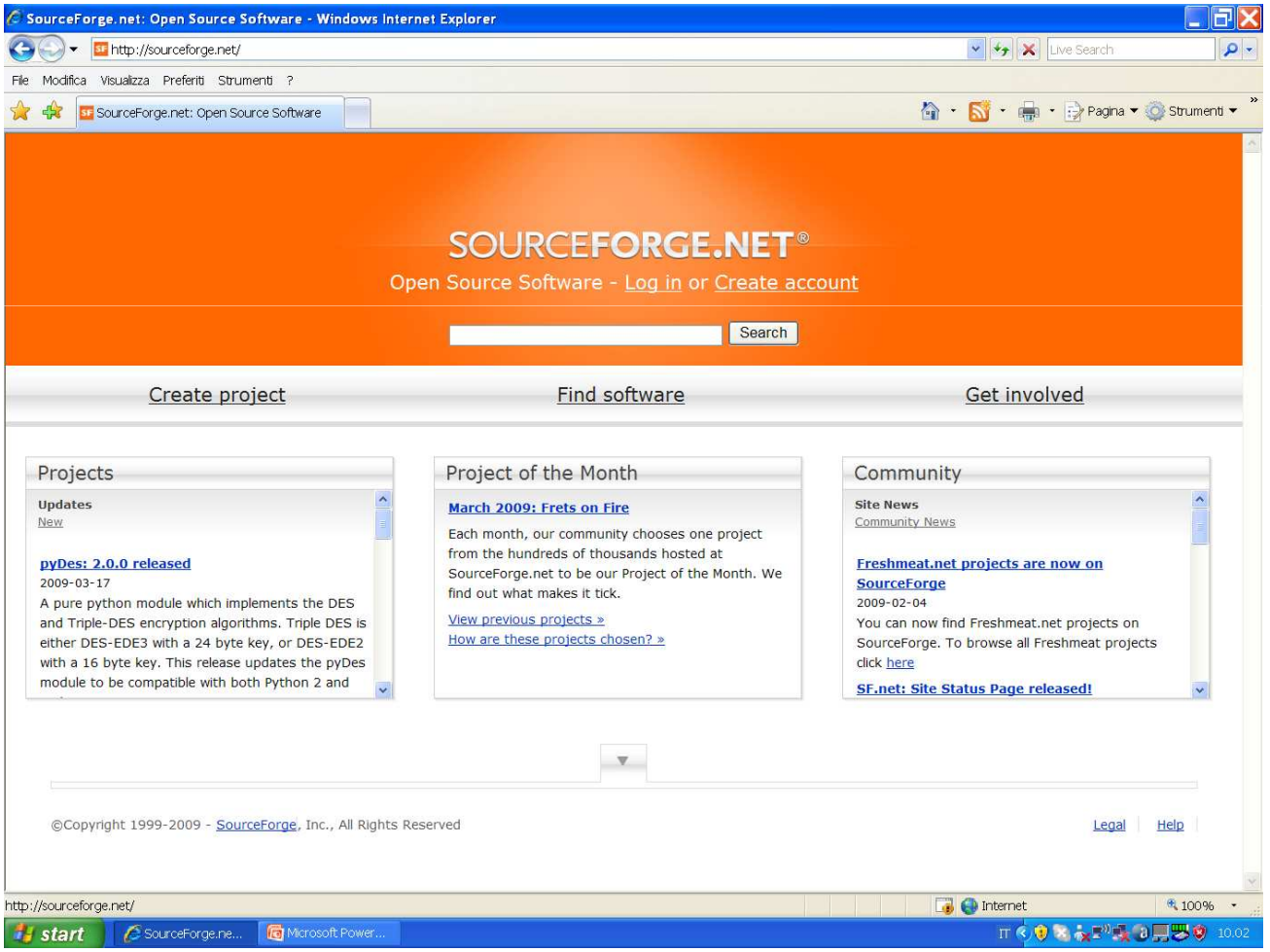
Motivazioni

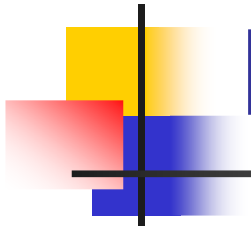
- Molto spesso le componenti grafiche standard disponibili nel package swing non riescono a soddisfare tutte le nostre necessità

- Soluzioni:
  - Ridefinire tali componenti al fine di realizzare il comportamento atteso;

  - Individuare delle librerie (es. open source) che soddisfino le nostre esigenze.

Source Forge

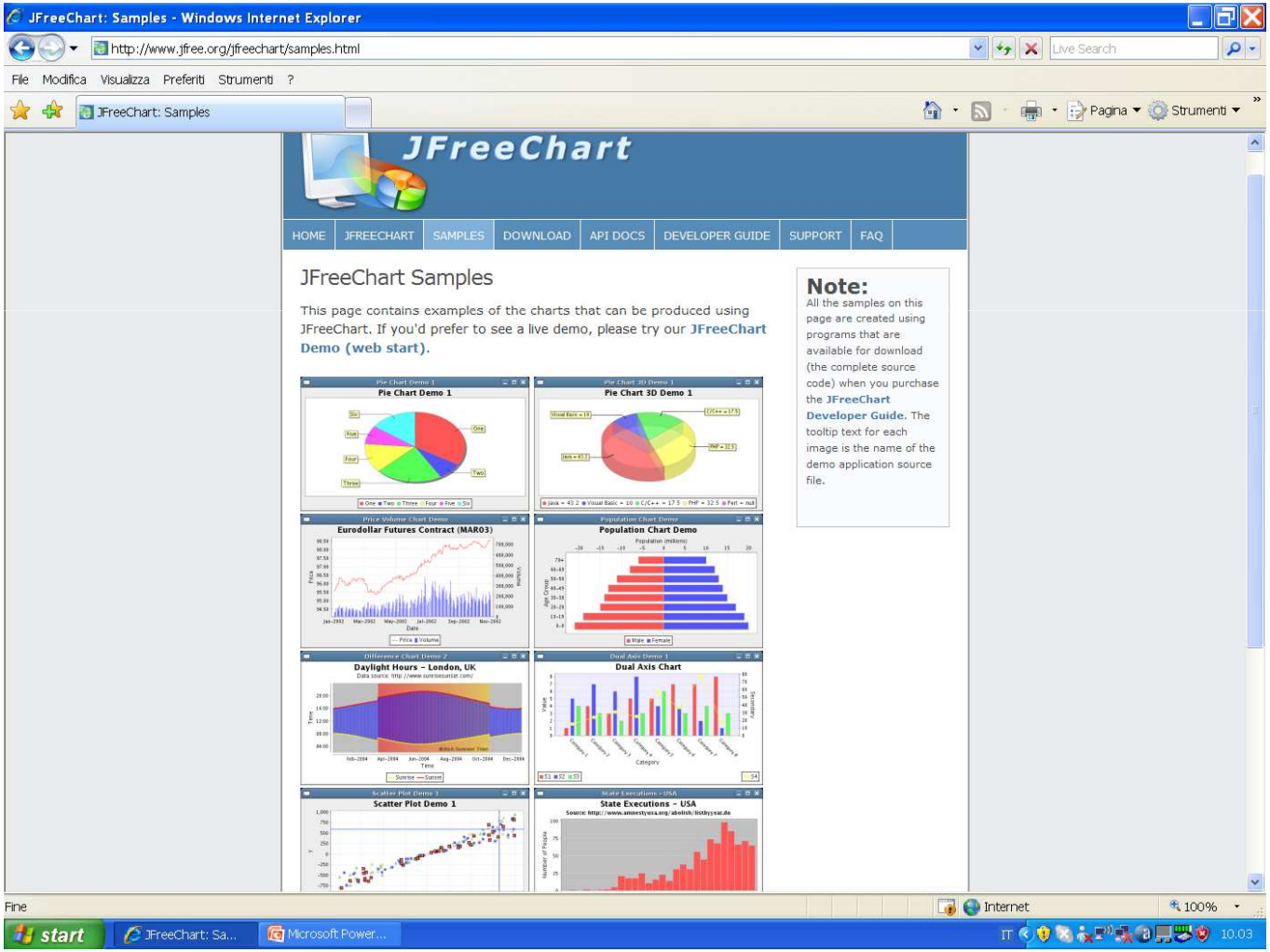




Librerie

- FlexDock;
- SwingX;
- JFreeChart;
- JGraph;
- Substance;
- Prefuse;
- Etc.

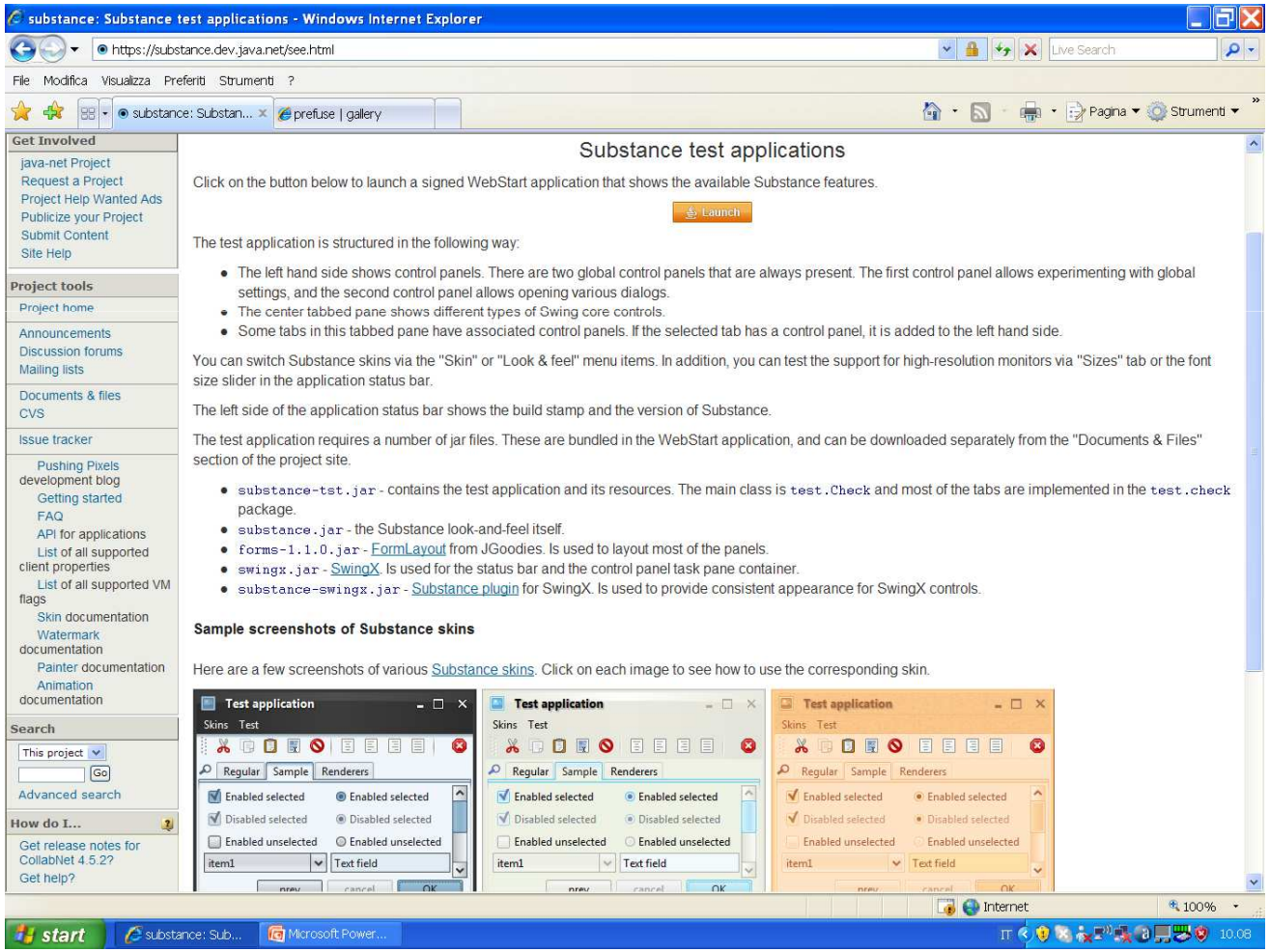
JFreeChart


JFreeChart: Samples - Windows Internet Explorer
http://www.jfree.org/jfreechart/samples.html
File Modifica Visualizza Preferiti Strumenti ?
JFreeChart: Samples
Pagina Strumenti
JFreeChart
HOME JFREECHART SAMPLES DOWNLOAD API DOCS DEVELOPER GUIDE SUPPORT FAQ
JFreeChart Samples
This page contains examples of the charts that can be produced using JFreeChart. If you'd prefer to see a live demo, please try our JFreeChart Demo (web start).
Pie Chart Demo 1
Pie Chart 3D Demo 1
Eurodollar Futures Contract (MAR03)
Population Chart Demo
Daylight Hours - London, UK
Dual Axis Chart
Scatter Plot Demo 1
State Executions - USA
Note:
All the samples on this page are created using programs that are available for download (the complete source code) when you purchase the JFreeChart Developer Guide. The tooltip text for each image is the name of the demo application source file.
Fine
Internet
100%
start
JFreeChart: Sa...
Microsoft Power...
10.03

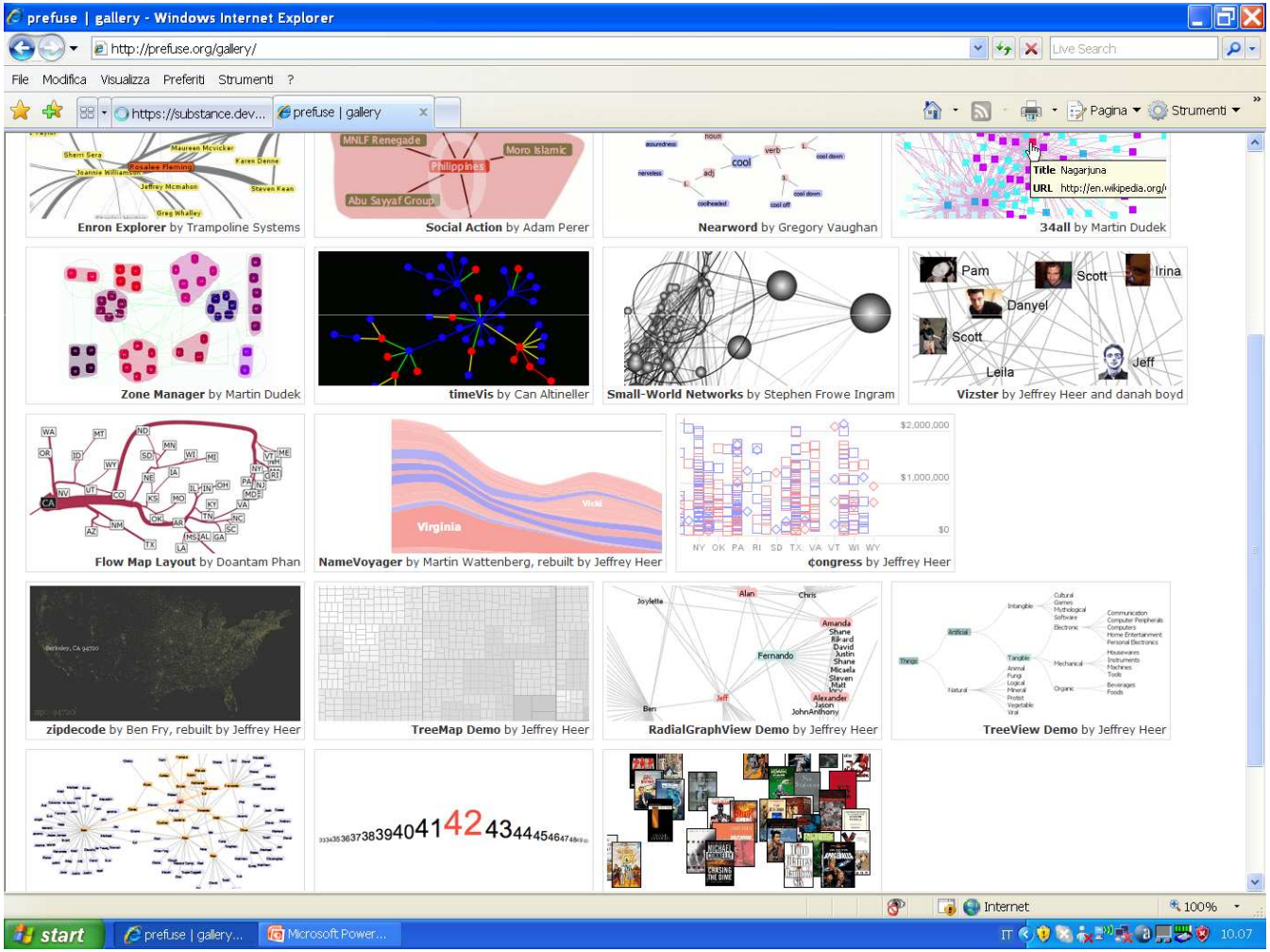
JGraph

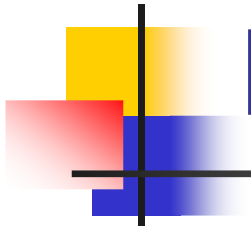

Screenshots - Windows Internet Explorer
http://www.jgraph.com/screenshots.html
File Modifica Visualizza Preferiti Strumenti ?
Screenshots
Pagina Strumenti
Screenshots
JGraph X
JGraph
Layout Pro
JGraphpad Pro
Purchase
Support
JGraph for the Web:
mxGraph
Aqua Data Studio
Aqua Data Studio is a database query tool and administration tool that allows developers to easily create, edit, and execute SQL scripts, as well as browse and visually modify database structures. Aqua Data Studio provides an integrated database environment with a single consistent interface to all major relational databases. This allows the database administrator or developer to tackle multiple tasks simultaneously from one application.
Submitted by Niels Gron, AquaFold
SQL Developer
SQL Developer is a is a database administration and query tool that provides a single consistent interface for various databases. Visually navigate through your database structure, create and execute SQL queries and scripts the easy way. Or reverse engineer complete data models with the integrated diagram editor.
Submitted by Jan Borchers, SOLYP Informatik GmbH
Scoreboard
Scoreboard is a web-based J2EE corporate performance management tool used for measuring performance and communicating strategy. Administrators may create and
Download immagine http://www.jgraph.com/images/emergysmall.gif...
Internet
100%
start
Screenshots - ...
Microsoft Power...
10.05

Substance


substance: Substance test applications - Windows Internet Explorer
https://substance.dev.java.net/see.html
Substance test applications
Click on the button below to launch a signed WebStart application that shows the available Substance features.
Launch
The test application is structured in the following way:
The left hand side shows control panels. There are two global control panels that are always present. The first control panel allows experimenting with global settings, and the second control panel allows opening various dialogs.
The center tabbed pane shows different types of Swing core controls.
Some tabs in this tabbed pane have associated control panels. If the selected tab has a control panel, it is added to the left hand side.
You can switch Substance skins via the "Skin" or "Look & feel" menu items. In addition, you can test the support for high-resolution monitors via "Sizes" tab or the font size slider in the application status bar.
The left side of the application status bar shows the build stamp and the version of Substance.
The test application requires a number of jar files. These are bundled in the WebStart application, and can be downloaded separately from the "Documents & Files" section of the project site.
substance-tst.jar - contains the test application and its resources. The main class is test.Check and most of the tabs are implemented in the test.check package.
substance.jar - the Substance look-and-feel itself.
forms-1.1.0.jar - FormLayout from JGoodies. Is used to layout most of the panels.
swingx.jar - SwingX. Is used for the status bar and the control panel task pane container.
substance-swingx.jar - Substance plugin for SwingX. Is used to provide consistent appearance for SwingX controls.
Sample screenshots of Substance skins
Here are a few screenshots of various Substance skins. Click on each image to see how to use the corresponding skin.
Test application
Get Involved
java-net Project
Request a Project
Project Help Wanted Ads
Publicize your Project
Submit Content
Site Help
Project tools
Project home
Announcements
Discussion forums
Mailing lists
Documents & files
CVS
Issue tracker
Pushing Pixels development blog
Getting started
FAQ
API for applications
List of all supported client properties
List of all supported VM flags
Skin documentation
Watermark documentation
Painter documentation
Animation documentation
Search
Advanced search
How do I...
Get release notes for CollabNet 4.5.2?
Get help?
start

Prefuse


prefuse | gallery - Windows Internet Explorer
http://prefuse.org/gallery/
Live Search
File Modifica Visualizza Preferiti Strumenti ?
https://substance.dev...
prefuse | gallery
Pagina
Strumenti
Enron Explorer by Trampoline Systems
Social Action by Adam Perer
Nearword by Gregory Vaughan
34all by Martin Dudek
Zone Manager by Martin Dudek
timeVis by Can Altineller
Small-World Networks by Stephen Frowe Ingram
Vizster by Jeffrey Heer and danah boyd
Flow Map Layout by Doantam Phan
NameVoyager by Martin Wattenberg, rebuilt by Jeffrey Heer
congress by Jeffrey Heer
zipdecode by Ben Fry, rebuilt by Jeffrey Heer
TreeMap Demo by Jeffrey Heer
RadialGraphView Demo by Jeffrey Heer
TreeView Demo by Jeffrey Heer
Internet
100%
start
prefuse | gallery...
Microsoft Power...

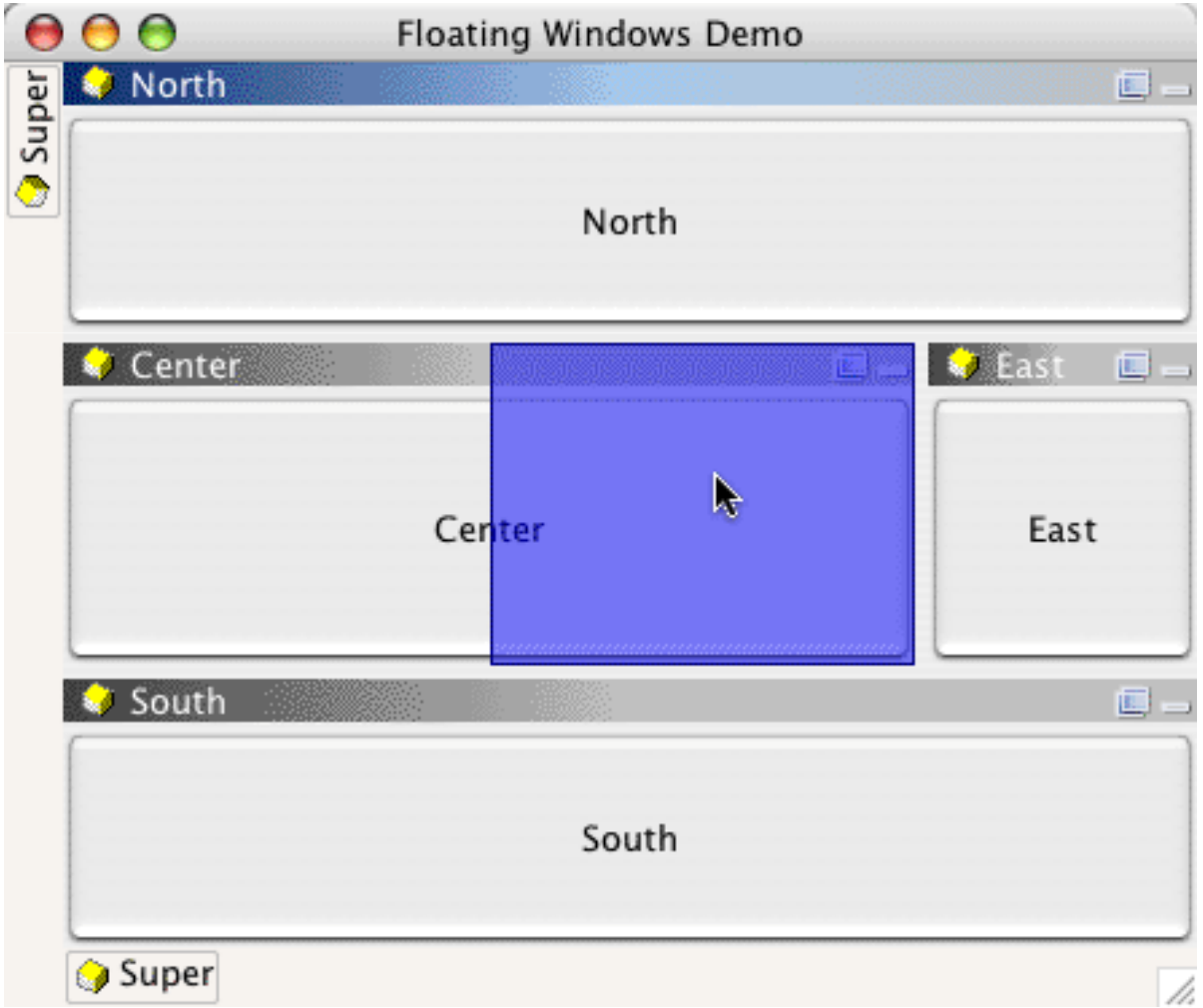


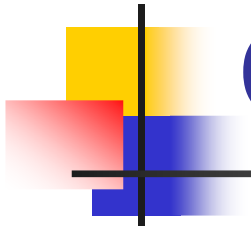
FlexDock

- FlexDock è Java *docking* framework per la realizzazione di applicazioni Swing cross-platform. Per docking si intende la possibilità di **spostare** un elemento grafico all'interno dell'interfaccia.

- Offre diverse funzionalità quali:
  - Tabbed and Split Layouts
  - Drag-n-Drop capability
  - Floating windows
  - Collapsible Containers
  - Layout Persistence

Cosa vuol dire “dockable”?

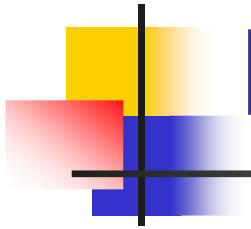




Concetti base

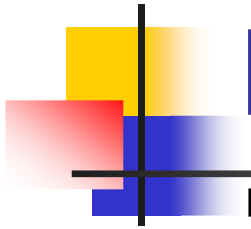
- Il concetto base è che qualsiasi elemento a cui si voglia dare la possibilità di essere *draggato o arrangiato* "DEVE" implementare l'interfaccia *Dockable*

- Tale elemento dovrà poi essere registrato presso il *DockingManager*, sarà esso infatti a gestire gli spostamenti degli elementi attraverso l'interfaccia



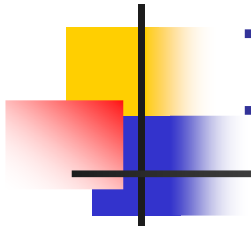
DockingPort

- Una DockingPort è un'opportuna area della Docking UI ove un componente può essere alloggiato
  - Normalmente l'alloggiamento significa che il componente può essere spostato da un'area (opportunamente specificata) ad un'altra area della stessa DockingPort
- Le docking ports si possono considerare come degli speciali containers dell'interfaccia grafica
- Sono molto semplici da usare ma permettono di muovere o arrangiare le aree in qualsiasi momento



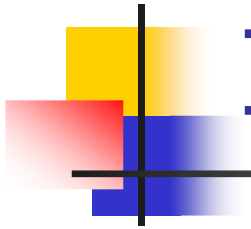
Esempio

```
public class PortDemo extends JFrame {

private DefaultDockingPort port;
...
public PortDemo(String title){
      //Istanzio la DockingPort
      port = new DefaultDockingPort();
      //Creo 2 semplici Pannelli con 2 Label
      JPanel pane1 = new JPanel();
      JLabel lb1=new JLabel("Sono il Panel 1");
      pane1.add(lb1);
      JPanel pane2 = new JPanel();
      JLabel lb2=new JLabel("Sono il Panel 2");
      pane2.add(lb2);
      //Effettuo il Dock tramite il Docking Manager
      DockingManager.dock(pane1, (DockingPort)port);
      DockingManager.dock(pane2, pane1, DockingConstants.NORTH_REGION, 0.3f);
      setContentPane(port);      }
...
}
```



Inserire più Docking Ports [1/2]

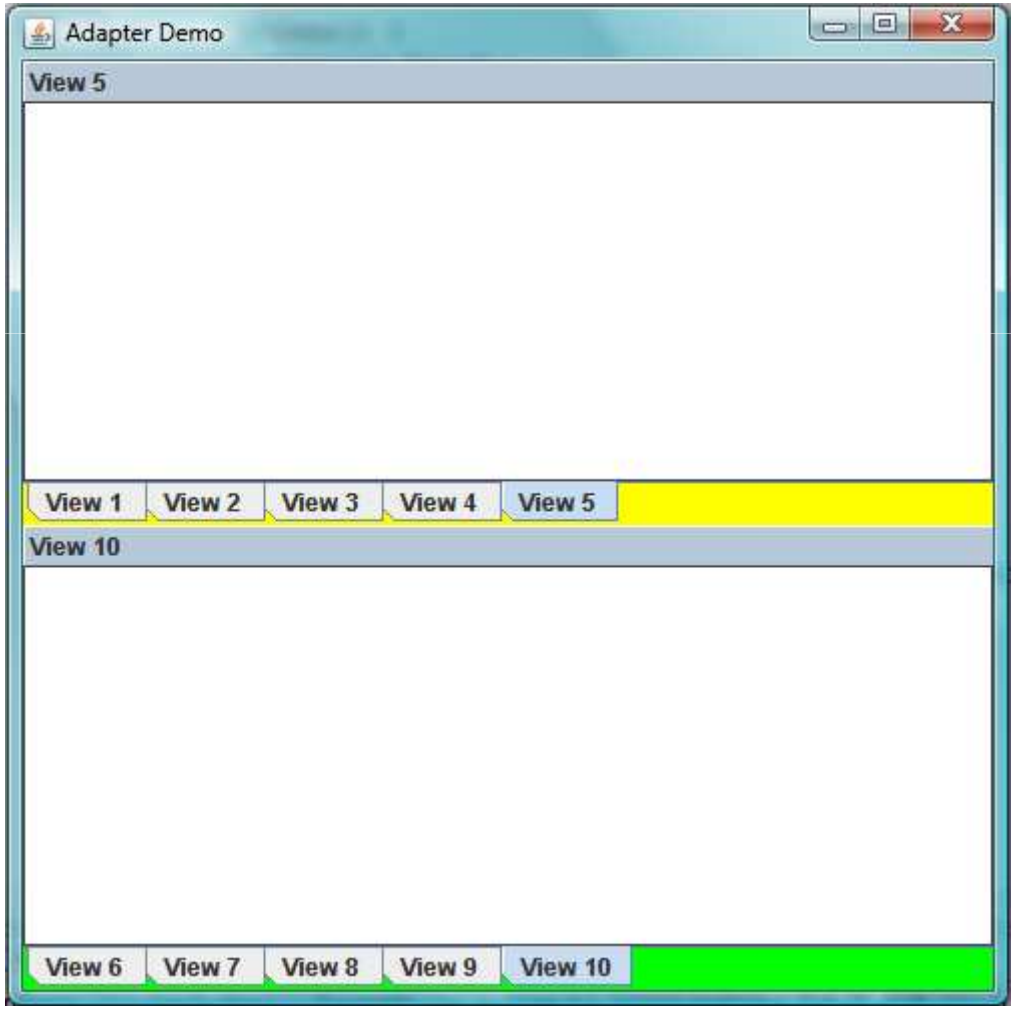
- E’ possibile inserire all’interno dello stesso frame più docking ports
- L’area a disposizione sarà divisa fra le docking ports inserite in base al layout utilizzato
- Componenti dockate su diverse docking port possono essere indifferentemente draggate da una all’altra

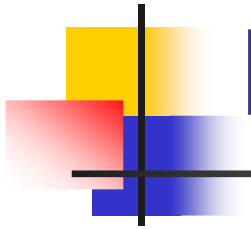


Inserire più Docking Ports [2/2]

- Nell'esempio di seguito sarà utilizzato come layout del frame un gridlayout (2 righe , 1 colonna) ed in ogni cella inserita una docking port con colore di sfondo diverso
- Al posto dei semplici JPanel sono utilizzati dei Titlepane disponibili con la libreria stessa

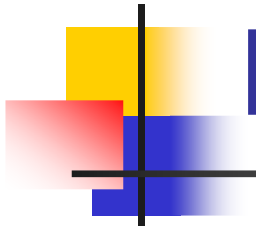
Esempio [1/3]


Adapter Demo
View 5
View 1
View 2
View 3
View 4
View 5
View 10
View 6
View 7
View 8
View 9
View 10



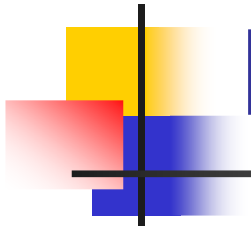
Esempio [2/3]

```
private Container createContentPane() {

    JPanel content = new JPanel(new GridLayout(2,1));
    portN = new DefaultDockingPort();
    portS = new DefaultDockingPort();
    Titlepane pane1 = new Titlepane("View 1");
    Titlepane pane2 = new Titlepane("View 2");
    Titlepane pane3 = new Titlepane("View 3");
    Titlepane pane4 = new Titlepane("View 4");
    Titlepane pane5 = new Titlepane("View 5");


    Titlepane pane6 = new Titlepane("View 6");
    Titlepane pane7 = new Titlepane("View 7");
    Titlepane pane8 = new Titlepane("View 8");
    Titlepane pane9 = new Titlepane("View 9");
    Titlepane pane10 = new Titlepane("View 10");
```



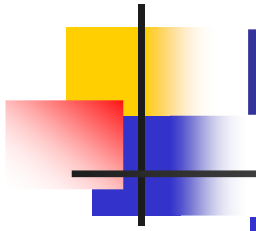
Esempio [3/3]

```
        DockingManager.dock(pane1, (DockingPort)portN);
        DockingManager.dock(pane2, pane1, DockingConstants.NORTH_REGION, 0.3f);
        DockingManager.dock(pane3, pane1, DockingConstants.SOUTH_REGION);
        DockingManager.dock(pane4, pane1, DockingConstants.EAST_REGION, 0.3f);
        DockingManager.dock(pane5, pane1, DockingConstants.WEST_REGION);

        DockingManager.dock(pane6, (DockingPort)portS);
        DockingManager.dock(pane7, pane6, DockingConstants.NORTH_REGION, 0.3f);
        DockingManager.dock(pane8, pane6, DockingConstants.SOUTH_REGION);
        DockingManager.dock(pane9, pane6, DockingConstants.EAST_REGION, 0.3f);
        DockingManager.dock(pane10, pane6, DockingConstants.WEST_REGION);

        portN.setBorder(BorderFactory.createLineBorder(Color.red, 1));
        portS.setBorder(BorderFactory.createLineBorder(Color.pink, 1));
        portN.setBackground(Color.yellow);
        portS.setBackground(Color.green);

        content.add(portN);
        content.add(portS);

        return content;
    }
```



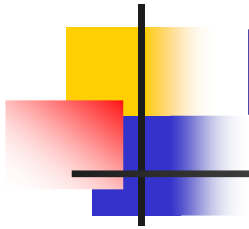
Dockables

- Sono gli oggetti che possono essere effettivamente draggati attraverso l'interfaccia ovvero attraverso le DockingPorts
- Rappresentano delle "Viste" sulla Docking UI
- In molti casi sono normali componenti gestiti internamente dal DockingManager



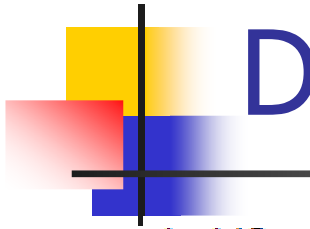
DockingManager

- La classe statica DockingManager è l'end-point finale a cui si deve registrare ogni componente che si voglia sia dockabile all'interno della UI
- Lo sviluppatore non dovrà occuparsi di alcun altro dettaglio oltre alla registrazione
- Il DockingManager fornisce dei metodi statici di utilità, tra cui la massimizzazione dei dockable



Registrazione di un elemento

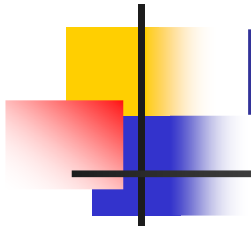
- La registrazione di un elemento presso il DockingManager può avvenire in diverse modalità:
  - L'elemento può essere registrato direttamente utilizzando il metodo statico dock() del Manager. In questo caso possiamo:
    - "Dockare" semplicemente l'elemento ad una porta;
    - Specificare come e dove dockare l'elemento rispetto ad un altro specificando anche per proporzioni di spazio da mantenere.



DockingManager API

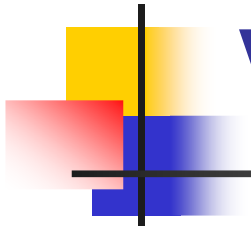
dock(Component dockable, Component parent) boolean - DockingManager
dock(Component dockable, DockingPort port) boolean - DockingManager
dock(Dockable dockable, Dockable parent) boolean - DockingManager
dock(Component dockable, Component parent, String region) boolean - DockingManager
dock(Component dockable, DockingPort port, String region) boolean - DockingManager
dock(Dockable dockable, Dockable parent, String region) boolean - DockingManager
dock(Dockable dockable, DockingPort port, String region) boolean - DockingManager
dock(Component dockable, Component parent, String region, float proportion) boolean - DockingManager
dock(Dockable dockable, Dockable parent, String region, float proportion) boolean - DockingManager

- Esempio:
  - `DockingManager.dock(pane2,pane1);`
- In questo caso pane2 sarà dockato come tab di pane1



Registrazione di un elemento

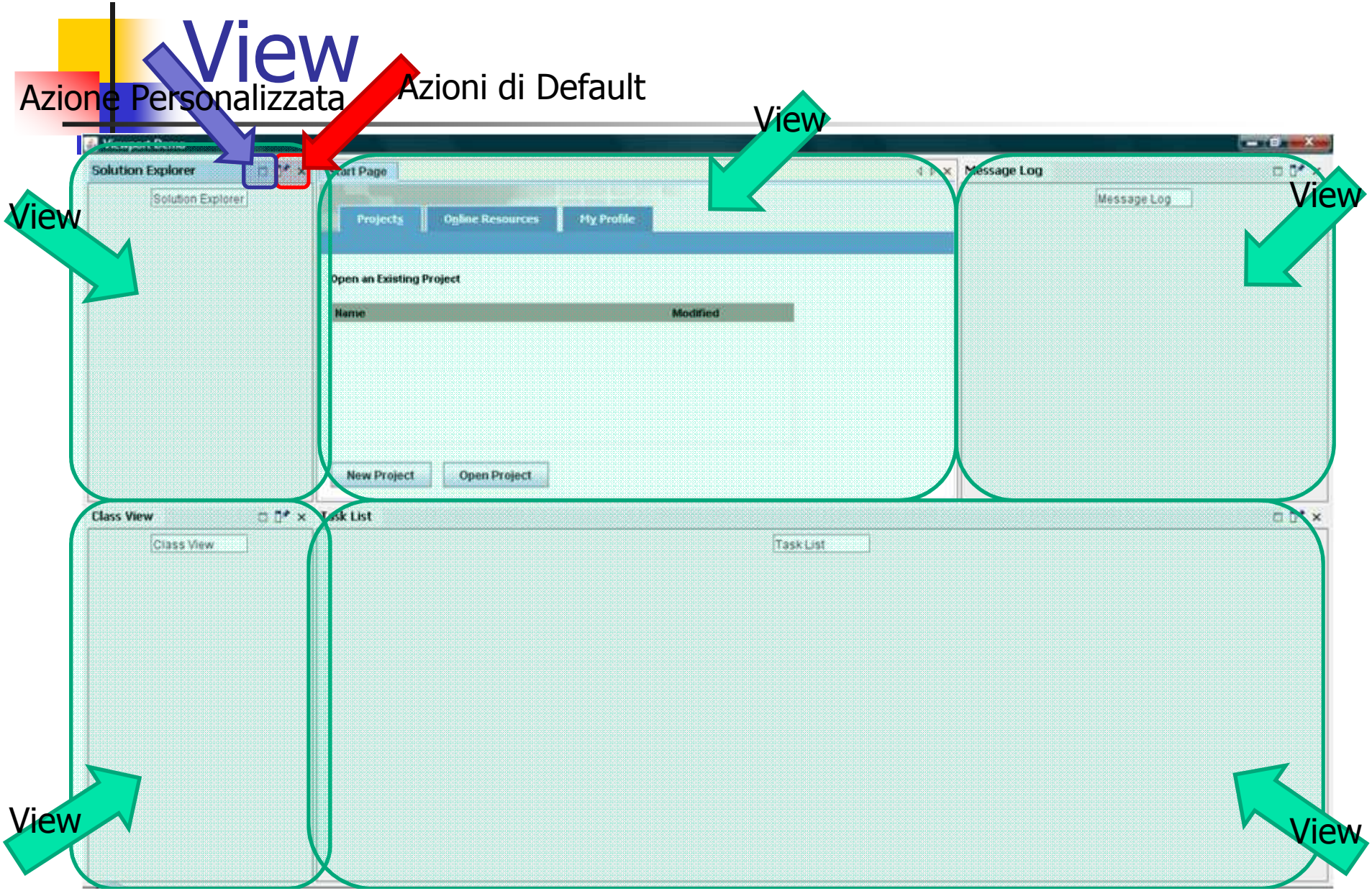
- È possibile registrare un elemento chiamando il metodo dock() su un elemento che a sua volta già “dockato”
  - In questo caso è anche possibile specificare come dividere lo spazio disponibile per quell’elemento fra i 2 componenti.

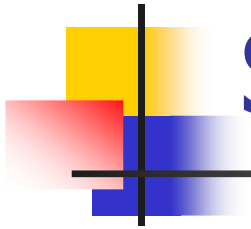


View

- Un particolare elemento dockable è la View che garantisce di default un aspetto molto accattivante.
- Fornisce diverse funzionalità di default come la chiusura e la riduzione ad icona.
- E' possibile aggiungere nuove funzionalità tramite l'aggiunta di Action personalizzate.
- Per utilizzarle, una volta create, basta settarne il contentPane con il pannello che vogliamo sia dockable.

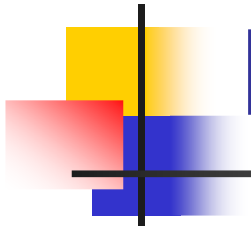
Esempio Completo: Gui con 5 View





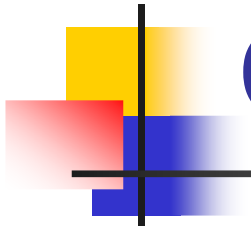
SwingUtilities

- Raggruppa un insieme di funzioni di utilità nella gestione delle Swing.
- E’ disponibile nel package org.flexdock.util.*
- Permette di settare aspetti di default per le view, centrare la GUI all’interno dello schermo, settare il focus, altro.



Metodi Accessori

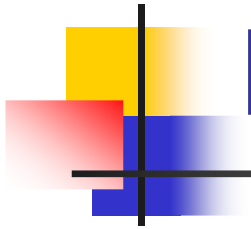
```
//Metodo per la creazione di un bottone a partire dall'icona
private static JButton createButton(Icon icon) {
     JButton button = new JButton(icon);
     button.setFocusable(false);
     return button;
 }
//Metodo per la creazione di un'Icona
 private static Icon createIcon(String icon) {
     return new ImageIcon(createImageImpl(icon));
 }
//Metodo per il recupero di un'immagine da file
 private static Image createImageImpl(String resourceName) {
     URL iconURL = MaximizationDemo.class.getResource(resourceName);
     if (iconURL == null) {
         throw new RuntimeException("Could not find: " +
resourceName);
     }
     return Toolkit.getDefaultToolkit().createImage(iconURL);
 }
```



Creare una View

```
private View createView(String id, String text) {
        final View view = new View(id,text);
   //Aggiungiamo alla View le Action per essere chiusa e ridotta a
   //icona (Già esistenti di default)
        view.addAction(CLOSE_ACTION);
        view.addAction(PIN_ACTION);
  //Creiamo ed Aggiungiamo una Action Personalizzata alla View
        Action maxButton =
         createMaxAction(view,createIcon("maximize.gif"));
        view.addAction(maxButton);

        JPanel p = new JPanel();
        p.setBorder(new LineBorder(Color.GRAY, 1));
        JTextField t = new JTextField(text);
        t.setPreferredSize(new Dimension(100, 20));
        p.add(t);

        view.setContentPane(p);
        return view;}
```

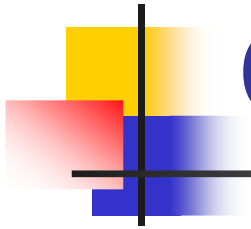
Action personalizzata: Massimizzazione



```
private Action createMaxAction(final View view, Icon icon) {

        Action action = new AbstractAction("Maximize",icon) {

            public void actionPerformed(ActionEvent arg0) {
        //Sfruttiamo il DockingManager per aggiungere la possibilità di
         //Massimizzare la View
                DockingManager.toggleMaximized((Dockable)view);
                getContentPane().repaint();

            }

        };

        return action;

}
```

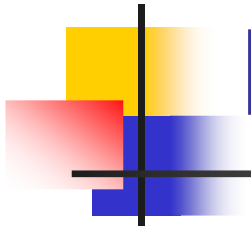
Creazione del pannello principale

```
private JPanel createContentPane() {
    JPanel p = new JPanel(new BorderLayout(0, 0));
    p.setBorder(new EmptyBorder(5, 5, 5, 5));
    //La ViewPort è la speciale DockingPort utilizzata per le View
    //la aggiungiamo poi al pannello principale
    Viewport viewport = new Viewport();
    p.add(viewport, BorderLayout.CENTER);
    //Creaiamo le View
    View startPage = createStartPage();
    View view1 = createView("solution.explorer", "Solution Explorer");
    View view2 = createView("task.list", "Task List");
    View view3 = createView("class.view", "Class View");
    View view4 = createView("message.log", "Message Log");
    //Aggiungiamo alla Viewport le diverse View create
    viewport.dock(startPage);
    startPage.dock(view1, WEST_REGION, .3f);
    startPage.dock(view2, SOUTH_REGION, .3f);
    startPage.dock(view4, EAST_REGION, .3f);
    view1.dock(view3, SOUTH_REGION, .3f);
    return p;}
```



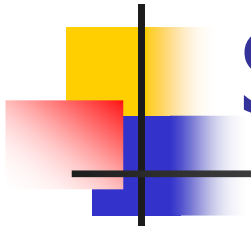
Costruttore e Startup

```
public ViewDemo() {
      super("Viewport Demo");
      setContentPane(createContentPane());
 }

 private static void startup() {
      // abilita il floating support
      DockingManager.setFloatingEnabled(true);

      JFrame f = new ViewDemo();
      f.setSize(800, 600);
      SwingUtility.centerOnScreen(f);
      DemoUtility.setCloseOperation(f);
      f.setVisible(true);
 }
```



Main

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

    SwingUtility.setPlaf("com.sun.java.swing.plaf.gtk.GTKLookAndFeel");
    startup();

}

private View createStartPage() {
        String id = "startPage";
        View view = new View(id, null, null);
        view.setTerritoryBlocked(CENTER_REGION, true);
        view.setTitlebar(null);
        view.setContentPane(new VSNetStartPage());
        return view;
 }
```

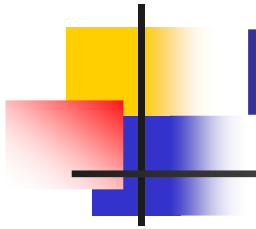


SwingX

- SwingX è un'estensione del Swing GUI toolkit, con nuovi e avanzati componenti, i quali forniscono funzionalità comunemente richieste da applicazioni client "ricche".

- In particolare permette di:

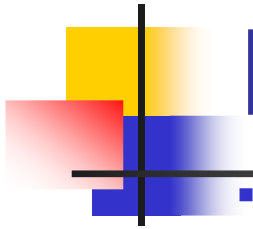
  - Ordinare, filtrare, evidenziare tabelle, alberi, e liste
  - Funzioni di Find/search
  - Auto-completamento
  - Login/authentication framework
  - TreeTable component
  - Date picker component
  - Tip-of-the-Day component



Estensioni

- SwingX estende le principali componenti grafiche Swing quali:
  - JXPanel;
  - JXTree;
  - JXTable;
  - JXList;
  - JXLabel;
  - JXTreeTable;
  - Other…

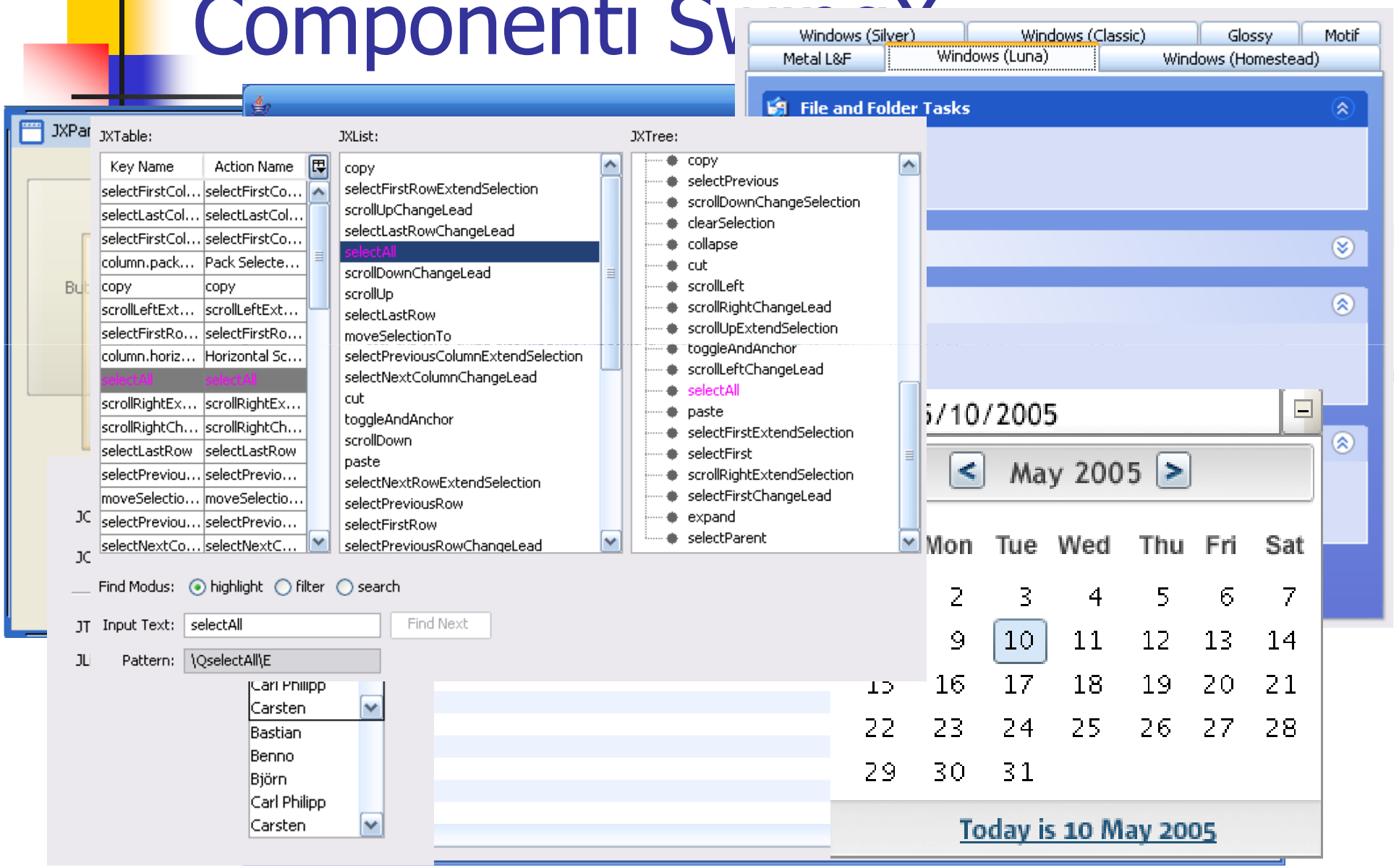
Il vantaggio di usare queste componenti risiede nella possibilità di utilizzare decoratori per arricchire la veste grafica (es. Bordi ad ombreggio, sottolineature) e funzionalità aggiuntive (es. ricerca e filtraggio)



Ricapitoliamo

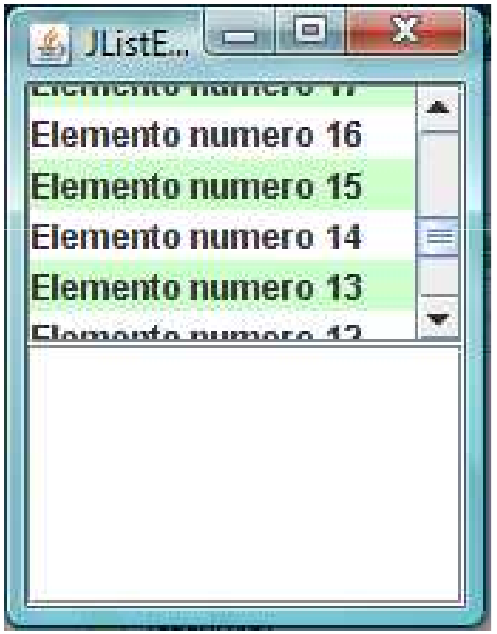
- JXButtonPanel Smart panel for displaying buttons in OS specific & local specific order
- JXComboBox Enhanced JComboBox supporting embedded tables, etc
- JXDatePicker Standard date chooser component
- JXEditorPane Enhancements to JEditorPane
- JXErrorDialog Enhanced standard error dialog
- JXFindDialog Standard find dialog
- JXGlassBox Dismiss on click & drop shadow functionality
- JXHyperlink Extends AbstractButton, adds action listener support
- JXImagePanel Displays an image
- JXList Adds in place editing to JList
- JXLoginDialog Standard Login Dialog
- JXMonthView Displays a month calendar
- JXPanel Adds translucency to the standard JPanel
- JXRadioGroup Simplifies dealing with ButtonGroup & JRadioButton
- JXStatusBar Enhanced status bar functionality
- JXTable Adds filtering/sorting/highlighting/column hiding to JTable
- JXTaskPane Contains actions and hyperlinks, provides "collapsing" functionality
- JXTitledPanel Title bar added to the JXPanel
- JXTree Enhancements to JTree
- JXTreeTable Combination of JTree & JTable

Componenti Sw



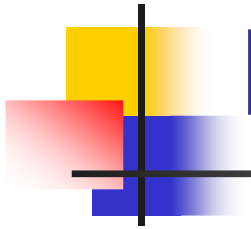
Esempio Highlighting e Filtering: JXList

```
public class JXListExample extends JFrame {

// La JXList che useremo nell'esempio
  private JXList list;
  private JTextArea output;

  public JXListExample() {
    super("JListExample");
    setSize(170,220);
    getContentPane().setLayout(new GridLayout(0,1));

 // Crea 20 elementi
    String[] items = new String[20];
    for(int i=0;i<19;i++)
      items[i]="Elemento numero "+String.valueOf(i);

    items[19]="ciao";
```



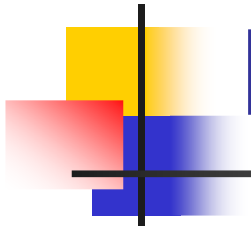
Highlighting e Filtering

```
// Inizializza una JXList
list= new JXList(items);
list.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
ListSelectionListener selectionListener = new SelectionListener();
list.addListSelectionListener(selectionListener);

//Coloro a righe alternate
list.setHighlighters(HighlighterFactory.createSimpleStriping(
HighlighterFactory.CLASSIC_LINE_PRINTER));
//Filtro gli elementi della Lista secondo il pattern passato nel
//PatterFilter
list.setFilterEnabled(true);
list.setFilters(new FilterPipeline(new PatternFilter("El", 0, 0), new
ShuttleSorter(0, false)));

// Crea la TextArea di output
output = new JTextArea();
output.setEditable(false);
```



Highlighting e Filtering

```
// assembla la GUI
   getContentPane().add(new JScrollPane(list));
   getContentPane().add(new JScrollPane(output));
   setVisible(true);
   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 }

  class SelectionListener implements ListSelectionListener {
     public void valueChanged(ListSelectionEvent e) {
       if(!e.getValueIsAdjusting()) {
         JList list = (JList)e.getSource();
         output.setText("");
         Object[] selectedItems = list.getSelectedValues();
         for(int i=0;i<selectedItems.length;i++)
           output.append(selectedItems[i].toString()+"\n");
       }
     }
  }
```

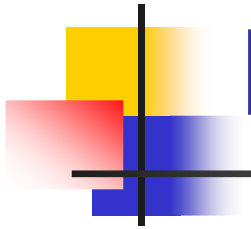


Highlighting e Filtering

```
 public static void main(String argv[])
{

   JXListExample b = new JXListExample();
}
}
```

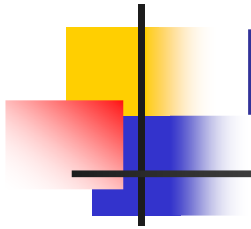
Esempio: JTaskPane

```
public class TaskPaneExample
{
 public static void main(String argv[])
 {
    try
    {
   UIManager.setLookAndFeel(UIManager.getSystemLo
    }
    catch(Exception e) {e.printStackTrace();}

   // Primo Componente
    JTextField textField = new JTextField("Premi C
    textField.setEditable(false);

    // Secondo Componente
    JLabel labelIcon = new JLabel(new ImageIcon("i
    labelIcon.setBorder(BorderFactory.createLineBo
```



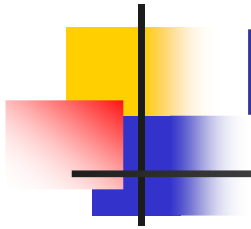
Esempio: JTaskPane

```
// Terzo Componente
   JButton okButton = new JButton("OK");
   okButton.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
       try
       {
         System.exit(0);
       }
       catch (Exception ex)
       { }
     }
   });

   // Quarto Componente
   JButton cancelButton = new JButton("Cancel");
```



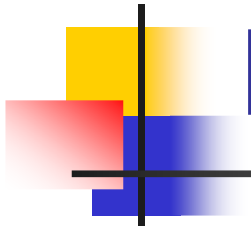
Esempio: JTaskPane

```
// Pannello NORTH
  JPanel northPanel = new JPanel();
  northPanel.setLayout(new GridLayout(1,0));
  northPanel.setBorder(BorderFactory.createEmptyBorder(10,4,10,4));
  northPanel.add(textField);

  // Pannello CENTER
  JPanel centralPanel = new JPanel();
  centralPanel.setLayout(new BorderLayout());
  centralPanel.setBorder(
      BorderFactory.createEmptyBorder(3,4,3,4));
  centralPanel.add(BorderLayout.CENTER,labelIcon);

  // Pannello SOUTH
  JPanel southPanel = new JPanel();
  southPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));

  southPanel.add(cancelButton);
  southPanel.add(okButton);
```



Esempio: JTaskPane

```
//Creo un Task Pane per ognuno dei 3 Pannelli creati in precedenza,
//Setto poi il titolo del task ed aggiungo il pannello
//in particolare per l'ultimo setto il task pane non espanso
JXTaskPane northTask = new JXTaskPane();
 northTask.setTitle("Pannello nord");
 northTask.add(northPanel);

 JXTaskPane centerTask = new JXTaskPane();
 centerTask.setTitle("Pannello centrale");
 centerTask.add(centralPanel);

 JXTaskPane southTask = new JXTaskPane();
 southTask.setTitle("Pannello sud");
 southTask.add(southPanel);
 southTask.setExpanded(false);
```



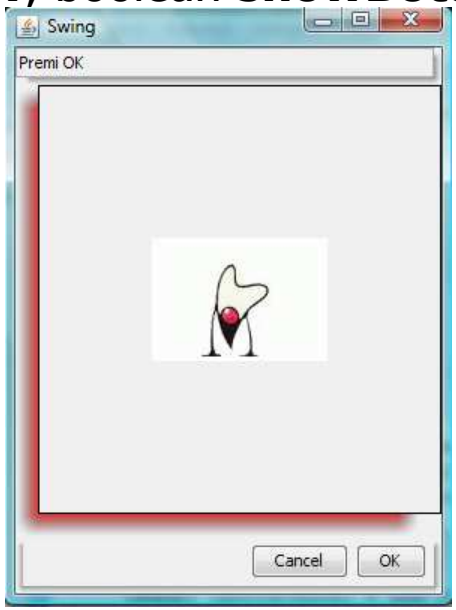
Esempio: JTaskPane

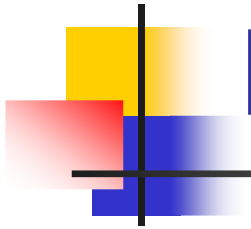
```
        //Ora creo il TaskPaneContainer a cui andrò ad aggiungere i TaskPane
        //creati in precedenza. Il Container sarà settato come contentPane
        //del JFrame
        JXTaskPaneContainer jxtc = new JXTaskPaneContainer();
         jxtc.add(northTask);
         jxtc.add(centerTask);
         jxtc.add(southTask);

         // Top Level Container
         JFrame f = new JFrame("Swing");
         f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
         f.getContentPane().setLayout(new BorderLayout());
         f.getContentPane().add(jxtc);


         f.pack();
         f.setVisible(true);
     }
}
```

Ombreggiatura dei componenti: Costruttori

- DropShadowBorder();
- DropShadowBorder(java.awt.Color **shadowColor**, int **shadowSize**);
- DropShadowBorder(boolean **showLeftShadow**);
- DropShadowBorder(java.awt.Color **shadowColor**, int **shadowSize**, float **shadowOpacity**, int **cornerSize**, boolean **showTopShadow**, boolean **showLeftShadow**, boolean **showBottomShadow**, boolean **showRightShadow**);

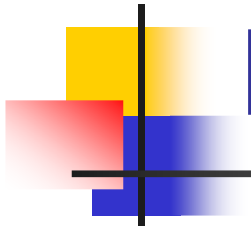
- ESEMPIO





Esempio: DropShadowBorder

```
public class WidgetExample
{
 public static void main(String argv[])
 {
    // imposta il Look&Feel di sistema
    try
    {
   UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {}

    // Primo Componente
    JTextField textField = new JTextField("Premi OK");
    textField.setEditable(false);

    // Secondo Componente
    JLabel labelIcon = new JLabel(new ImageIcon("img/img.jpg"));
    labelIcon.setBorder(BorderFactory.createLineBorder(Color.black));
```

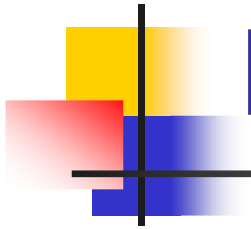


Esempio: DropShadowBorder

```
// Terzo Componente
    JButton okButton = new JButton("OK");
    okButton.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        try
        {
          System.exit(0);
        }
        catch (Exception ex)
        {ex.printStackTrace(); }
      }
    });
    // Quarto Componente
    JButton cancelButton = new JButton("Cancel");
```

Esempio: DropShadowBorder

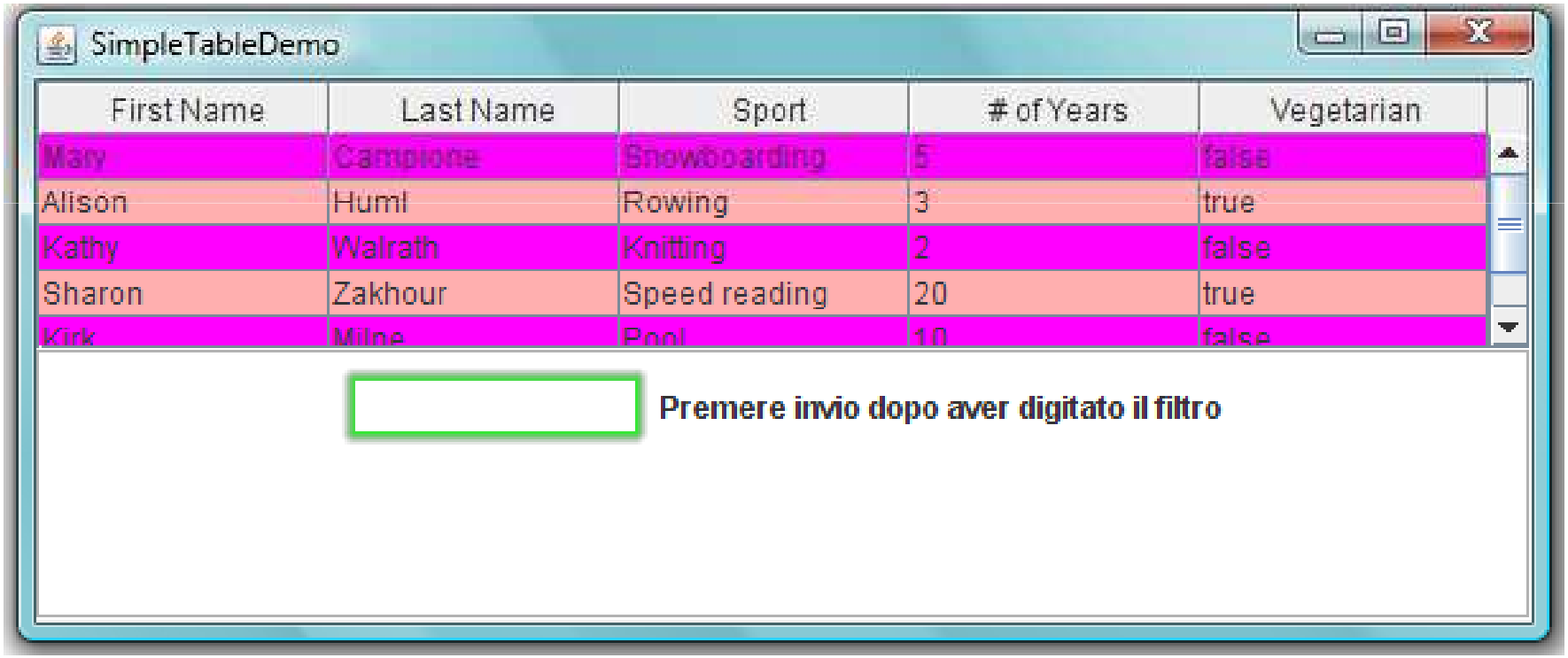
```
// Creo il Pannello NORTH e setto un ombreggiatura di default come border
JPanel northPanel = new JPanel();
northPanel.setLayout(new GridLayout(1,0));
northPanel.setBorder(new DropShadowBorder());
northPanel.add(textField);

// Pannello CENTER con i diversi parametri dell'ombreggiatura settati:
// Colore ombreggiatura, dimensione, opacità, lati su cui ombreggiare
JPanel centralPanel = new JPanel();
centralPanel.setLayout(new BorderLayout());
centralPanel.setBorder(new
DropShadowBorder(Color.RED,15,0.9f,15,false,true,true,false ));
centralPanel.add(BorderLayout.CENTER,labelIcon);

// Pannello SOUTH con ombreggiatura di default a sinistra
JPanel southPanel = new JPanel();
southPanel.setLayout(new FlowLayout(FlowLayout.RIGHT));
southPanel.setBorder(new DropShadowBorder(true));
southPanel.add(cancelButton);
southPanel.add(okButton);
```

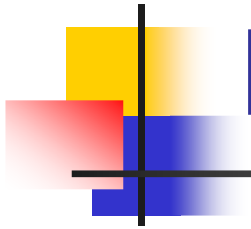


Esempio: DropShadowBorder

```
// Top Level Container
    JFrame f = new JFrame("Swing");
    f.setBounds(15, 15, 300, 400);
    f.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    f.getContentPane().setLayout(new BorderLayout());
    f.getContentPane().add(BorderLayout.NORTH,northPanel);
    f.getContentPane().add(BorderLayout.CENTER,centralPanel);
    f.getContentPane().add(BorderLayout.SOUTH,southPanel);

    f.setVisible(true);
 }
}
```

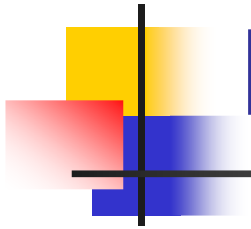
Esempio: JXTable





Esempio: JXTable

```
public class SimpleTableDemo extends JPanel {

    private JTextField filter;

    public SimpleTableDemo() {
        super(new GridLayout(2,0));

        //Definisco l'array del nome delle colonne della tabella
        String[] columnNames = {"First Name",
                                "Last Name",
                                "Sport",
                                "# of Years",
                                "Vegetarian"};
```

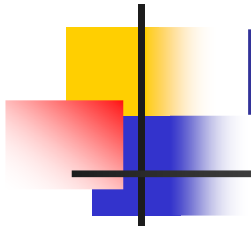


Esempio: JXTable

```
//Preparo la matrice dei valori che andranno a riempire la JXTable
//che andremo a visualizzare
Object[][] data = {
            {"Mary", "Campione",
             "Snowboarding", 5, false},
            {"Alison", "Huml",
             "Rowing", 3, true},
            {"Kathy", "Walrath",
             "Knitting", 2, false},
            {"Sharon", "Zakhour",
             "Speed reading", 20, true},
            {"Kirk", "Milne",
             "Pool", 10, false},
            {"Mark", "Landers",
                 "Pool", 10, false}
        };
```

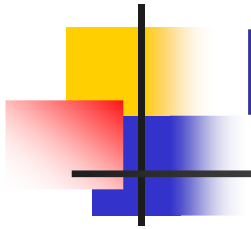
Esempio: JXTable

```
//Istanzio la JXTable
final JXTable table = new JXTable(data, columnNames);
        table.setPreferredScrollableViewportSize(new Dimension(500, 70));
        table.setFillsViewportHeight(true);


   table.setHighlighters(HighlighterFactory.createAlternateStriping(Color.
   MAGENTA, Color.PINK));

        //Creo uno scrollpane a cui aggiungere la tabella
        JScrollPane scrollPane = new JScrollPane(table);

        filter = new JTextField();
        filter.setPreferredSize(new Dimension(120,30));
        JPanel filterPane = new JPanel(new FlowLayout());
        filterPane.setBorder(BorderFactory.createEtchedBorder());
        filter.setBorder(new
         DropShadowBorder(Color.green,5,0.9f,5,true,true,true,true));
```



Esempio: JXTable

```
filter.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {
int col = table.getColumn("First Name").getModelIndex();
 table.setFilters(new FilterPipeline(new PatternFilter("^"+filter.getText(), 0,
    col) ));
}
});
 JLabel suggeestion = new JLabel("Premere invio dopo aver digitato il filtro");
        filterPane.add(filter);
        filterPane.add(suggeestion);
        filterPane.setBackground(Color.white);
        filter.setCaretColor(Color.BLUE);

        //Aggiungo lo scrollpane ed il TextFied per il filtering
        add(scrollPane);
        add(filterPane);
    }
```



Esempio: JXTable

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

                    createAndShowGUI();
    }
}
```