

# Graphical User Interface (GUI)

- Pour créer une application lourde java graphique, il est nécessaire de créer une **GUI** (Graphical User Interface).
- On utilise principalement deux packages
  - **awt** : abstract window toolkit
  - **swing** : basé sur awt. Remplace progressivement awt.
    - plus performant
    - architecture **MVC** (Modèle Vue Contrôleur)
- Pour créer la GUI on peut :
  - coder l'interface à la main
  - utiliser un **WYSIWYG** (what you see is what you get)
    - eclipse + plugin (**window builder**, jigloo...)
    - netbeans (IDE concurrent d'eclipse)

# Présentation de Swing

- Qu'est-ce que Swing ?
  - C'est une librairie écrite en JAVA
  - Développée par Sun pour remplacer AWT
  - Permet de dessiner des interfaces graphiques complexes
  - Présente de manière native depuis Java 1.2
  - Le but étant d'être complètement détaché de l'OS sur lequel l'application sera lancée. Là où AWT était plus lié au système.

# Pourquoi Swing ?

- Existe depuis de nombreuses années et bénéficie de nombreuses ressources sur Internet
- Propose de multiple Look & Feel disponibles partout sur la toile
- Multiplateforme simple et efficace

# Composants de base : JComponent

- Composant de base pour la quasi-totalité des objets utilisables en Swing à l'exception de :
  - JWindow, JFrame, JDialog, JApplet
- On retrouvera grâce à cela un nombre d'options toujours disponibles comme :
  - La transparence
  - L'état : actif, inactif
  - La bulle d'information
  - La bordure
  - Taille minimale, maximale
  - Alignement
  - ...

# Définir une fenêtre

- Les principales fenêtres existantes :
  - JWindow
  - JFrame
  - JDialog
  - JOptionPane
  - JFileChooser / JColorChooser

# Fenêtre : Jwindow

- Fenêtre simple sans bordures, décorations et boutons
- Utilisé pour les splashscreens ou les notifications
- Exemple:

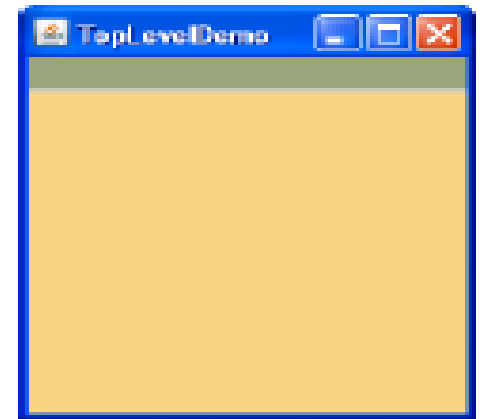


# Fenêtre :JFrame

- Propose la barre de titre et les boutons
- Possibilité d'enlever cette décoration

`setUndecorated(true|false);`

- Fermer l'interface
- Cliquer sur la croix ne suffit pas à arrêter le programme



`setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);`

# Exemple : swing

- Les classes swing sont préfixées J... (JFrame...)

- [http://fr.wikipedia.org/wiki/Swing\\_%28Java%29](http://fr.wikipedia.org/wiki/Swing_%28Java%29)

```
import javax.swing.JFrame;
import javax.swing.JLabel;

public class HelloWorld {
    public static void main(String[] args) {
        // On crée une fenêtre dont le titre est "Hello World!"
        JFrame frame = new JFrame("Hello World!");

        // La fenêtre doit se fermer quand on clique sur la croix rouge
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        // On ajoute le texte "Hello, World!" dans la fenêtre
        frame.getContentPane().add(new JLabel("Hello, World!"));

        // On demande d'attribuer une taille minimale à la fenêtre
        // (juste assez pour voir tous les composants)
        frame.pack();

        frame.setLocationRelativeTo(null); // Centrer la fenêtre
        frame.setVisible(true);           // Rendre la fenêtre visible
    }
}
```



# La classe JFrame

- Une JFrame de base n'est pas utilisable/paramétrée.
  - On peut préciser si la fenêtre est redimensionnable, toujours au premier plan, avec ou sans contours / boutons, sa taille, sa position, son comportement quand on clique sur fermer, si elle est visible...
  - On fait un **héritage** sur JFrame pour chacune de nos fenêtres

```
import javax.swing.JFrame;

public class Fenetre extends JFrame{

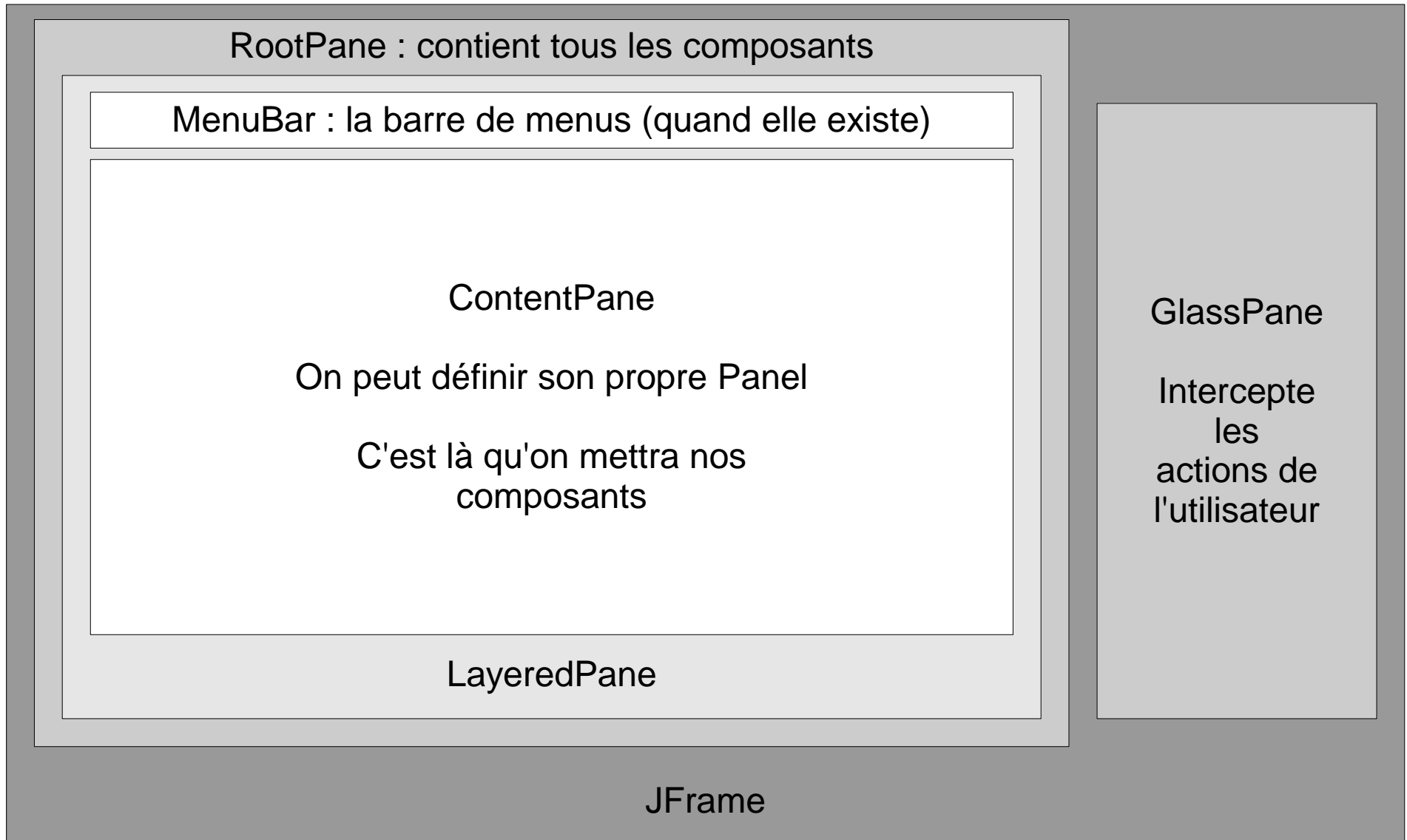
    public Fenetre(){
        this.setTitle("Titre de la fenêtre");
        this.setSize(400, 500);

        // Centrer à l'écran
        this.setLocationRelativeTo(null);

        // Fermer lorsqu'on clique sur "Fermer"
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

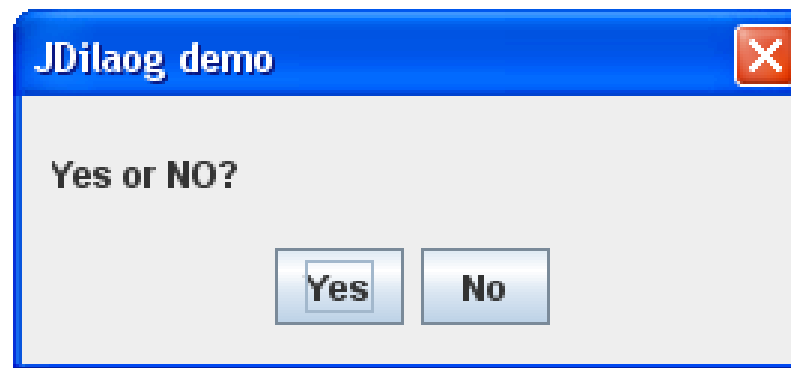
        // Par défaut un élément n'est pas visible !
        this.setVisible(true);
    }
}
```

# La classe JFrame : structure



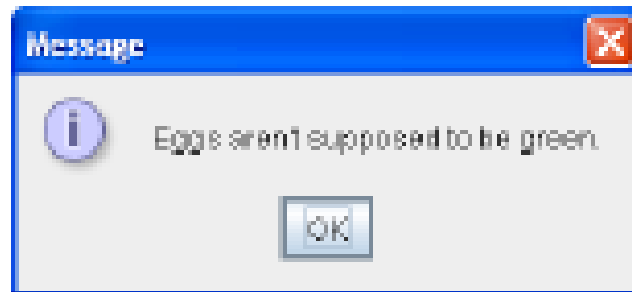
# Fenêtre : JDialog

- Fenêtre fille d'une application
- Apparaît au dessus d'une autre fenêtre
- Possibilité de la rendre modale ou non
- Modale signifie ne pas pouvoir interagir avec la fenêtre mère tant que la boîte de dialogue est ouverte



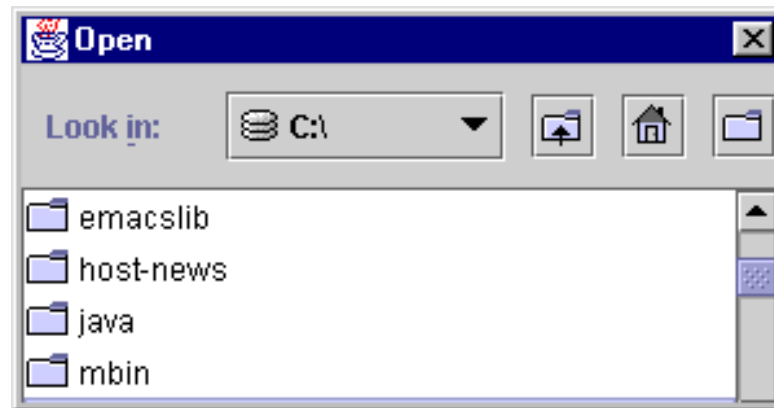
# Fenêtre : JOptionPane

- Ce sont des boîtes de dialogues prédéfinies
- 3 types de boites :
  - Message d'information  
`JOptionPane.showMessageDialog(...)`
  - Message contenant une question  
`JOptionPane.showConfirmDialog(...)`
  - Message de saisie de données  
`JOptionPane.showInputDialog(...)`



# Fenêtre : JFileChooser

- Un JFileChooser permet de sélectionner un fichier en parcourant l'arborescence du système de fichier.
- Ex :
  - ```
JFileChooser fc = new JFileChooser();  
int returnVal = fc.showOpenDialog(aFrame);  
if (returnVal == JFileChooser.APPROVE_OPTION)  
{  
    File file = fc.getSelectedFile();  
}
```



# Fenêtre : JColorChooser

- Un JColorChooser permet de choisir une couleur



- Une méthode :  
`public static Color showDialog(Component c , String title , Color initialColor);`

# Contenaire

- Les conteneurs contiennent et gèrent des composants graphiques.
- Ils dérivent de `java.awt.Container`.
- A l'exécution, ils apparaissent généralement sous forme de panneaux, de fenêtres ou de boîtes de dialogues.
- La totalité de votre travail de conception d'interfaces utilisateurs se fait dans des conteneurs.
- Les conteneurs sont aussi des composants. En tant que tels, ils vous laissent interagir avec eux, c'est-à-dire définir leurs propriétés, appeler leurs méthodes et répondre à leurs événements.

# Conteneur

- Composants qui ont pour but principal de contenir d'autres composants
- Les principaux conteneurs :
  - JPanel
  - JScrollPane
  - JTabbedPane
  - ...



# JPanel

- Le container le plus simple
- N'a quasiment aucun impact sur l'aspect graphique d'une interface
- Sert surtout à positionner de nouveaux composants
- Principales méthodes :
  - `setLayout(...)`
  - `add(...)`
- Un JPanel peut également jouer le rôle de surface graphique dans laquelle le développeur peut dessiner ou afficher des images. Il suffit pour cela de redéfinir la méthode `paintComponent(Graphics)`.

# La classe JPanel

- Contient le fond + les contrôles de la fenêtre
  - Agence les composants, redessine en cas de besoin...
  - On peut obtenir les dimensions du `JPanel`
  - On peut personnaliser n'importe quel composant (bouton, panel...) en réimplémentant `paintComponent(Graphics)` : son fond, un dessin
  - L'objet `Graphic` permet de dessiner des formes, des lignes, du texte, une image...
    - Pour aller plus loin caster `Graphics` en `Graphics2D`

```
import java.awt.Graphics;
import javax.swing.JPanel;

public class Panneau extends JPanel {
    public void paintComponent(Graphics g) {
        g.setColor(Color.ORANGE);
        g.fillOval(20, 20, 75, 75);
    }
}
```

```
// Dans le constructeur Fenêtre() on ajoute :
this.setContentPane(new Panneau());
```

# JScrollPane

- Permet d'avoir des barres de défilement lorsque le composant qu'il contient est trop grand par rapport à la taille qui lui est allouée.



# JTabbedPane

- Reprend le système des onglets
- Permet d'avoir plusieurs panneaux sur la même surface

...

```
JTabbedPane tab = new JTabbedPane();  
tab.addTab("onglet 1", new JLabel("Panneau 1"));  
JTabbedPane tab = new JTabbedPane();  
tab.addTab("onglet 2", new JLabel("Panneau 2"));
```

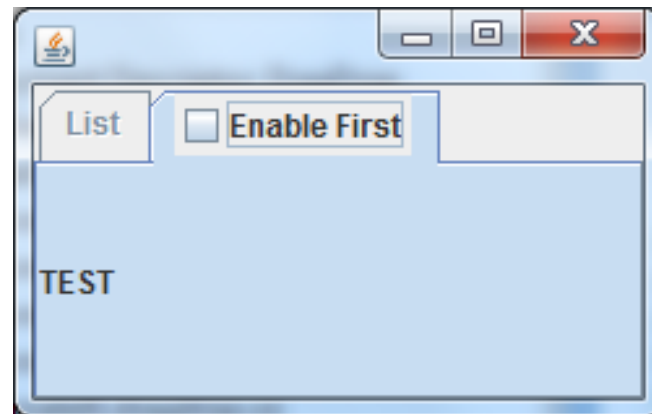
...



# JTabbedPane

- Il est possible d'ajouter un composant à la place du titre

```
JTabbedPane tab = new JTabbedPane();  
tab.addTab("List" , new JLabel("Panneau 1"));  
tab.addTab(null, new JLabel("TEST"));  
tab.setTabComponentAt(1, new JCheckBox("Enable First"));
```

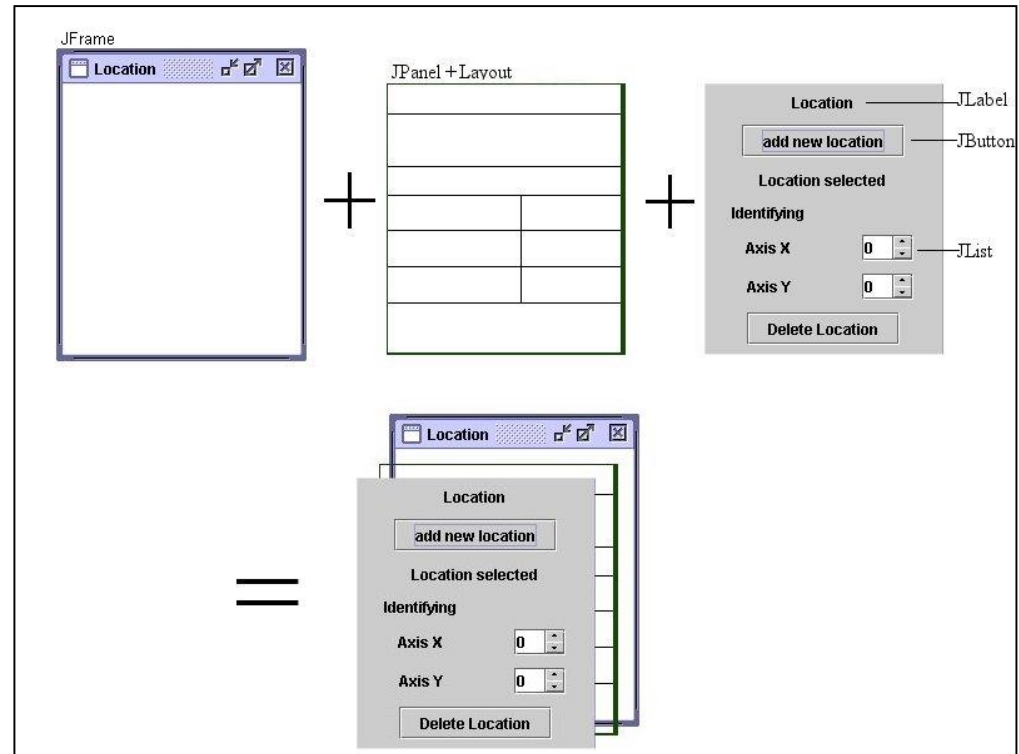


# Les composants

- Les composants sont les blocs de construction utilisés par les outils de conception visuelle de Netbeans ou autres pour construire un programme. (bouton, textbox....)
- Chaque composant représente un élément de programme, tel un objet de l'interface utilisateur, une base de données ou un utilitaire système. Vous construisez un programme en choisissant et **reliant ces éléments**.

# Schéma complet de développement d'une fenêtre Swing

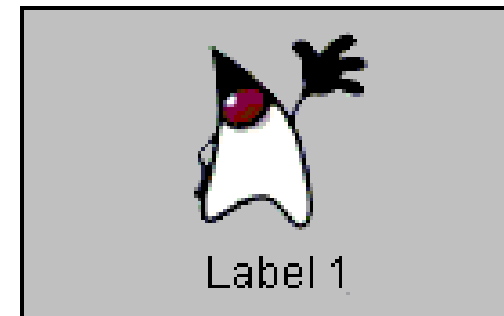
- 1- création d'une fenêtre.(Jframe..)
- 2- Intégration d'un JPanel dans la fenêtre (sert à intégrer des composants)
- 3- utilisation d'un layout à l'intérieur du JPanel pour ordonnancer les composants.
- 4- intégration des composants dans le JPanel en utilisant son layout.



# JLabel

- Un JLabel permet d'afficher du texte ou une image.
- Un JLabel peut contenir plusieurs lignes et il comprend les balises HTML.

```
public JLabel(String s);  
public JLabel(Icon i);
```

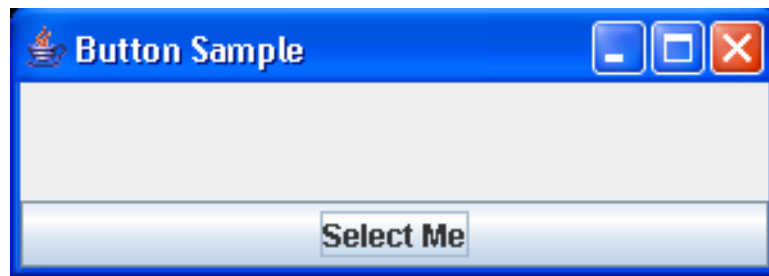




# JButton

- Un JButton nous permet d'avoir un bouton sur notre interface.

```
JButton bouton = new JButton("Select Me");  
panel.add(bouton);
```



# La classe JButton

```
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Fenetre extends JFrame{
    private JPanel pan = new JPanel();
    private JButton bouton = new JButton("Mon Bouton");

    public Fenetre(){
        this.setTitle("Mon titre");
        this.setSize(300, 300);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setLocationRelativeTo(null);

        pan.add(bouton); // sera centré par le JPanel
        // Si on n'utilise pas de JPanel, le panel ne remet pas
        // en forme le bouton qui occupera toute la JFrame !

        this.setContentPane(pan);
        this.setVisible(true);
    }
}
```

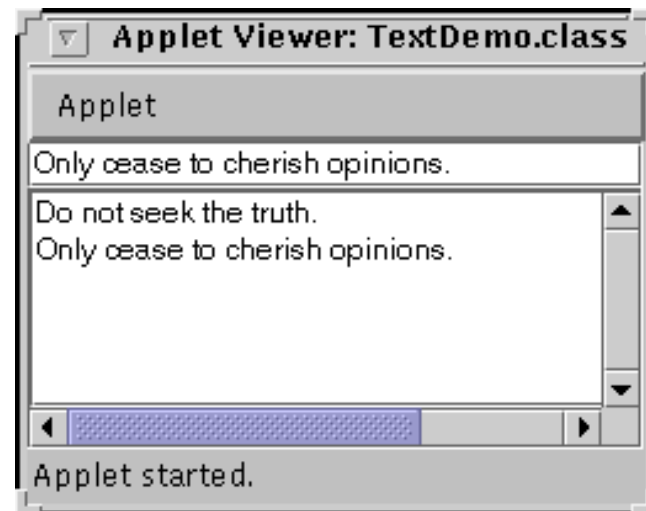
# JTextField

- Un JTextField est un composant qui permet d'écrire du texte. Il est composé d'une seule ligne contrairement au JTextArea
- Le JPasswordField permet de cacher ce qui est écrit
- Quelques méthodes:

```
public JTextField(String s);
```

```
public String getText();
```

```
public String setText();
```



# JList

- Une JList propose plusieurs éléments rangés en colonne.  
Une JList peut proposer une sélection simple ou multiple  
Les JList sont souvent contenues dans un scrolled pane

- Quelques méthodes:

```
public JList(Vector v);  
public JList( ListModel l);  
Object[] getSelectedValues();
```



# Les Layouts

- Un Layout permet de découper un Panel en zones
  - BorderLayout : *CENTER, NORTH, SOUTH, EAST, WEST*

```
// Dans le constructeur Fenetre
this.setLayout(new BorderLayout());

//On ajoute le bouton au contentPane de la JFrame
this.getContentPane().add(new JButton("CENTER"), BorderLayout.CENTER);
this.getContentPane().add(new JButton("NORTH"), BorderLayout.NORTH);
this.getContentPane().add(new JButton("SOUTH"), BorderLayout.SOUTH);
this.getContentPane().add(new JButton("WEST"), BorderLayout.WEST);
this.getContentPane().add(new JButton("EAST"), BorderLayout.EAST);
```

- GridLayout : découpe le panneau en une grille dont le nombre de lignes et colonnes peut être précisé
- FlowLayout : éléments placés de gauche à droite, passe à la ligne si besoin
- Il en existe plein d'autres : CardLayout, GridBagLayout, ...
  - [http://www.siteduzero.com/tutoriel-3-10480-votre-premier-bouton.html#ss\\_part\\_2](http://www.siteduzero.com/tutoriel-3-10480-votre-premier-bouton.html#ss_part_2)

# BorderLayout

- Le BorderLayout sépare un container en cinq zones: NORTH, SOUTH, EAST, WEST et CENTER
- Lorsque l'on agrandit le container, le centre s'agrandit. Les autres zone prennent uniquement l'espace qui leur est nécessaire.



- ```
Container contentPane = getContentPane();
contentPane.setLayout(new BorderLayout());
contentPane.add(new JButton("Button 1 (NORTH)"),
BorderLayout.NORTH);
```

# FlowLayout

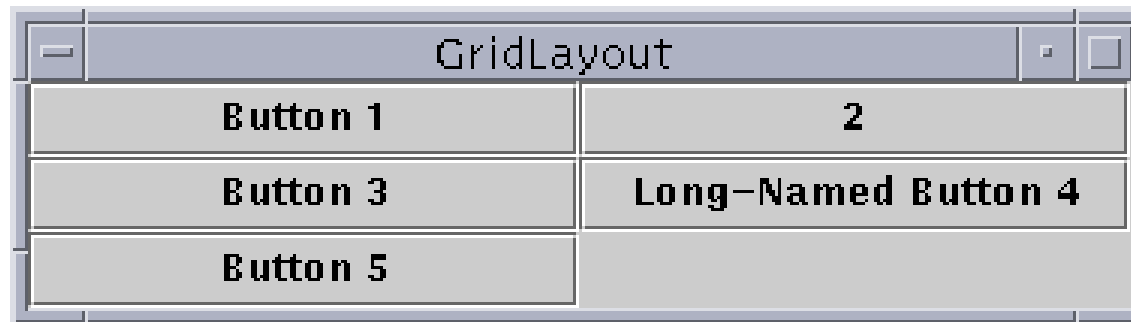
- Un FlowLayout permet de ranger les composants dans une ligne. Si l'espace est trop petit, une autre ligne est créée.



```
Container contentPane = getContentPane();
contentPane.setLayout(new FlowLayout());
contentPane.add(new JButton("Button 1"));
contentPane.add(new JButton("2"));
contentPane.add(new JButton("Button 3"));
contentPane.add(new JButton("Long-Named Button 4"));
contentPane.add(new JButton("Button 5"));
```

# GridLayout

- Un GridLayout permet de positionner les composants sur une grille.



- Ex:

```
Container contentPane = getContentPane();
contentPane.setLayout(new GridLayout(0,2));
contentPane.add(new JButton("Button 1"));
contentPane.add(new JButton("2"));
contentPane.add(new JButton("Button 3"));
contentPane.add(new JButton("Long-Named Button 4"));
contentPane.add(new JButton("Button 5"));
```



# Gestion des événements

- Les composants Swing créent des événements, soit directement, soit par une action de l'utilisateur sur le composant. Ces événements peuvent déclencher une action exécutée par d'autre(s) composant(s).
- Un composant qui crée des événements est appelé **source**. Le composant source délègue le traitement de l'événement au composant auditeur.
- Un composant qui traite un événement est appelé **auditeur** (listener)

# Gestion des événements

- Les événements que nous venons de décrire sont occasionnés par l'utilisateur. on dira que ces événements sont générés par les composants graphiques eux-mêmes. Ces objets sont appelés des sources d'événements.
- Ex: JButton

Action	Lorsque le bouton a été actionné par l'utilisateur (appuyé et relâché)
MouseMotion	Lorsque la souris est déplacée ou draguée sur la surface du bouton
Mouse	Lorsque le bouton de la souris est appuyé, relâché, ou cliqué, ou encore lorsque la souris sort ou entre dans la surface du bouton.
Focus	Lorsque le bouton obtient ou perd le focus (suite par exemple à la pression de la touche TAB)
Component	Lorsque le bouton a été déplacé ou caché
Key	Lorsqu'une touche du clavier est pressée alors que le bouton possède le focus.

# PROGRAMMATION: COMMENT GÉRER UN ÉVÉNEMENT ?

- **1re étape:** choisir une catégorie d'événements
- Voici un tableau de tous les événements pour chaque composant, suivi de la liste des classes associées pour chaque événement.

Component	Listener							
	<a href="#">action</a>	<a href="#">caret</a>	<a href="#">change</a>	<a href="#">document, undoable edit</a>	<a href="#">item</a>	<a href="#">list selection</a>	<a href="#">window</a>	other
<a href="#">button</a>	x		x		x			
<a href="#">check box</a>	x		x		x			
<a href="#">color chooser</a>			x					
<a href="#">combo box</a>	x				x			
<a href="#">dialog</a>								
<a href="#">editor pane</a>		x		x			x	<a href="#">hyperlink</a>
<a href="#">file chooser</a>	x							
<a href="#">formatted text field</a>	x	x		x				
<a href="#">frame</a>						x		
<a href="#">internal frame</a>								<a href="#">internal frame</a>
<a href="#">list</a>						x		<a href="#">list data</a>
<a href="#">menu</a>								<a href="#">menu</a>
<a href="#">menu item</a>	x		x		x			<a href="#">menu key</a> <a href="#">menu drag mouse</a>

# PROGRAMMATION: COMMENT GÉRER UN ÉVÉNEMENT ?

Component	Listener							
	<a href="#">action</a>	<a href="#">caret</a>	<a href="#">change</a>	<a href="#">document, undoable edit</a>	<a href="#">item</a>	<a href="#">list selection</a>	<a href="#">window</a>	other
<a href="#">option pane</a>								
<a href="#">password field</a>	x	x		x				
<a href="#">popup menu</a>								<a href="#">popup menu</a>
<a href="#">progress bar</a>			x					
<a href="#">radio button</a>	x		x		x			
<a href="#">slider</a>			x					
<a href="#">spinner</a>			x					
<a href="#">tabbed pane</a>			x					
<a href="#">table</a>						x		<a href="#">table model</a> <a href="#">table column model</a> <a href="#">cell editor</a>
<a href="#">text area</a>		x		x				
<a href="#">text field</a>	x	x		x				
<a href="#">text pane</a>		x		x				<a href="#">hyperlink</a>
<a href="#">toggle button</a>	x		x		x			
<a href="#">tree</a>								<a href="#">tree expansion</a> <a href="#">tree will expand</a> <a href="#">tree model</a> <a href="#">tree selection</a>
viewport (used by <a href="#">scrollpane</a> )			x					

# PROGRAMMATION: COMMENT GÉRER UN ÉVÉNEMENT ?

- **2ème étape:** inscrire un auditeur pour une certaine catégorie d'événements
  - Le Composant.add**XXXListener** (*auditeur*);
  - Le Composant désigne le composant graphique auprès duquel on désire s'abonner
  - XXX désigne la catégorie d'événements concernée (Action, MouseMotion, ..)
  - auditeur désigne l'objet qui se met à l'écoute des événements

# PROGRAMMATION: COMMENT GÉRER UN ÉVÉNEMENT ?

- **3<sup>ème</sup> étape:** écrire les gestionnaires d'événements dans la classe de l'auditeur

public void add**XXXListener**(**XXXListener**)

public void add**ActionListener**(**ActionListener**)

public void

add**MouseMotionListener**(**MouseMotionListe  
ner**)

# Les listeners

- **EventListener** : interface qui indique quels événements on rattrape pour un composant :
  - Exemple **interface** `MouseListener`, ...

```
public class Bouton extends JButton implements MouseListener {  
    // Constructeurs, membres, méthodes...  
    // Listeners  
    void mouseEntered(MouseEvent e) {}  
    void mouseClicked(MouseEvent e) {}  
    // ... et les autres Listeners  
}
```

- Il ne reste plus qu'à implémenter les méthodes qui en découlent
  - Exemple : changer le style du composant
  - Si on n'a pas de code à mettre, on laisse les { } vides

# Les listeners

- Parfois le contrôle n'a assez d'information pour faire le traitement.
  - On peut le délèguer à sa fenêtre qui peut stocker l'état du logiciel.
  - Exemple : **interface** ActionListener

```
public class Fenetre extends JFrame implements ActionListener{
    private JButton bouton1 = new JButton("mon bouton 1");
    private JButton bouton2 = new JButton("mon bouton 2");
    private JLabel label = new JLabel("mon label");

    // Listeners
    public void actionPerformed(ActionEvent arg0) {
        if(arg0.getSource() == bouton1){
            label.setText("Action sur bouton 1");
        } else if(arg0.getSource() == bouton2){
            label.setText("Action sur bouton 2");
        }
    }
}
```

Pas lisible  
Pas découpé  
Pas pratique  
Pas modulaire



# Les listeners


## ■ Pour plus de lisibilité :

- on crée `Listener` personnalisé par contrôle (au lieu de Fenetre)
- on attache le `Listener` personnalisé avec `addListener`
- `addComponent` avec le `Listener` personnalisé correspondant

```
public class Fenetre extends JFrame{
    private JButton bouton1 = new JButton1("mon bouton 1");

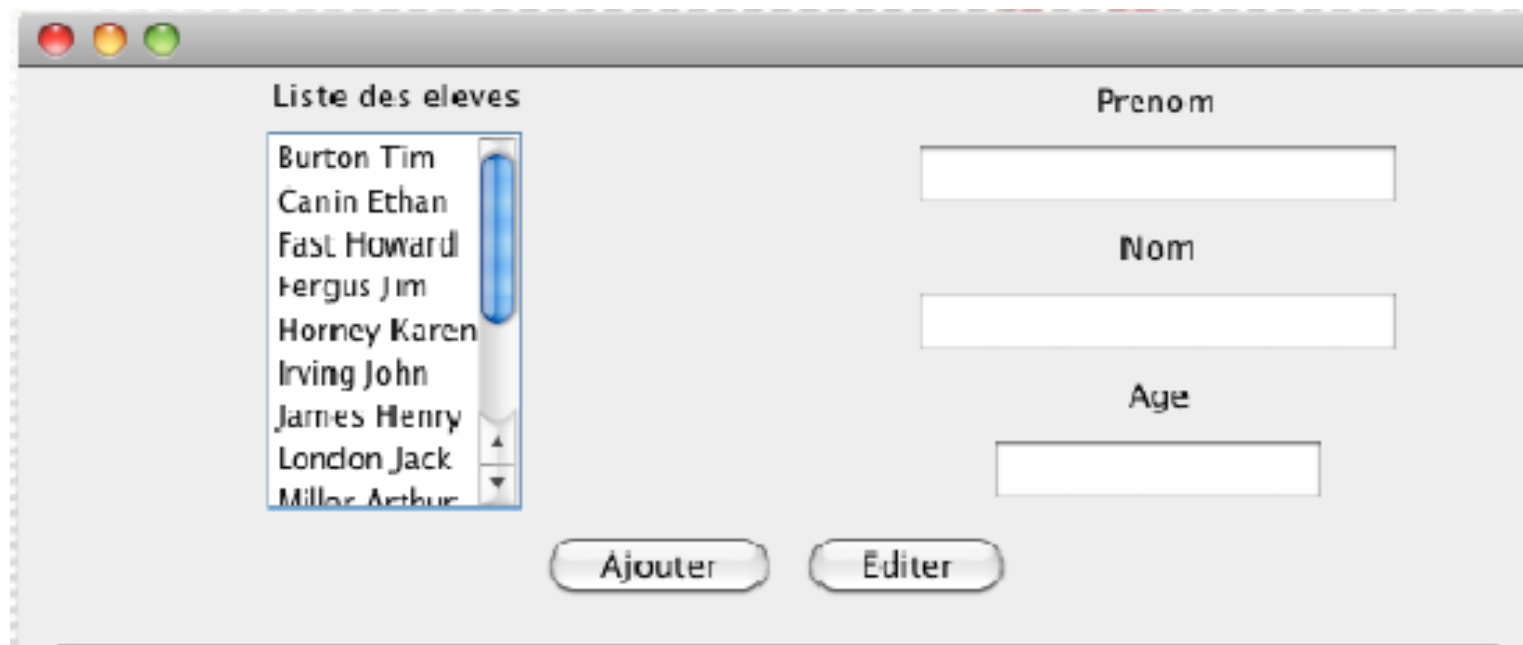
    // Constructeur
    public Fenetre(ActionEvent arg0) {
        //...
        bouton1.addActionListener(new Bouton1Listener());
        // idem avec les autres composants
    }

    public class Bouton1Listener implements ActionListener{
        public void actionPerformed(ActionEvent arg0) {}
    } // idem avec les autres composants
}
```



# TP: interface graphique

- Coder une interface pour gérer une liste des élèves
  - La liste nous permettra de voir tous les noms
  - Si on sélectionne un élément, ses informations seront affichées.111



# Window Builder

## ■ Installer Window Builder

- Vérifiez votre version d'Eclipse (Help, About Eclipse...)
- Ajouter le repository Window Builder (Help, Install new software
  - <http://code.google.com/intl/fr/webtoolkit/tools/download-wbpro.html>
- Cocher la case permettant d'installer Swing Builder