



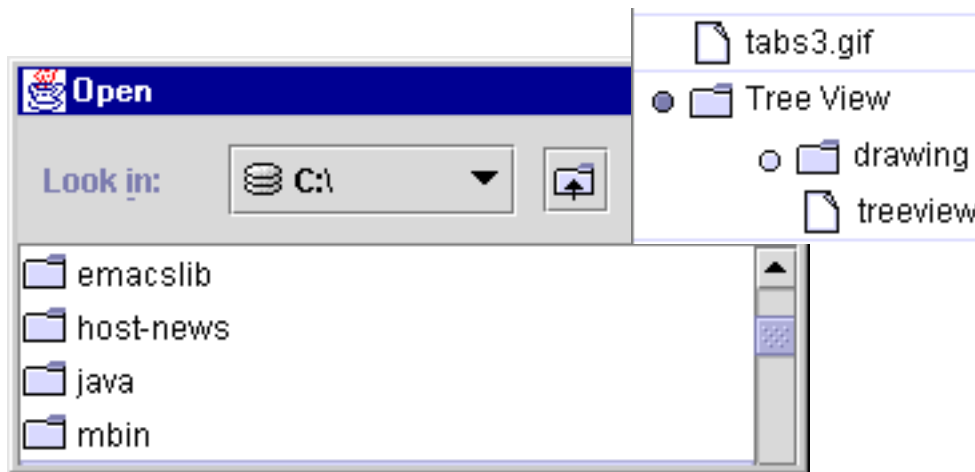
Composants Swing

Xavier Blanc

Xavier.Blanc@lip6.fr



Exemples de Swing





Concepts

Les différents Swing



Architecture Swing

Une application est composée de plusieurs Swing :

- Un composant **top-level**
- Plusieurs composants **conteneur intermédiaire**, ils contiennent d'autre composants
- Des **composants atomiques**



Le composant JComponent

- Tous les composants Swing héritent de JComponent
- Les composants ont des Tool Tips
- Les composants ont des bordures
- Entité graphique la plus abstraite



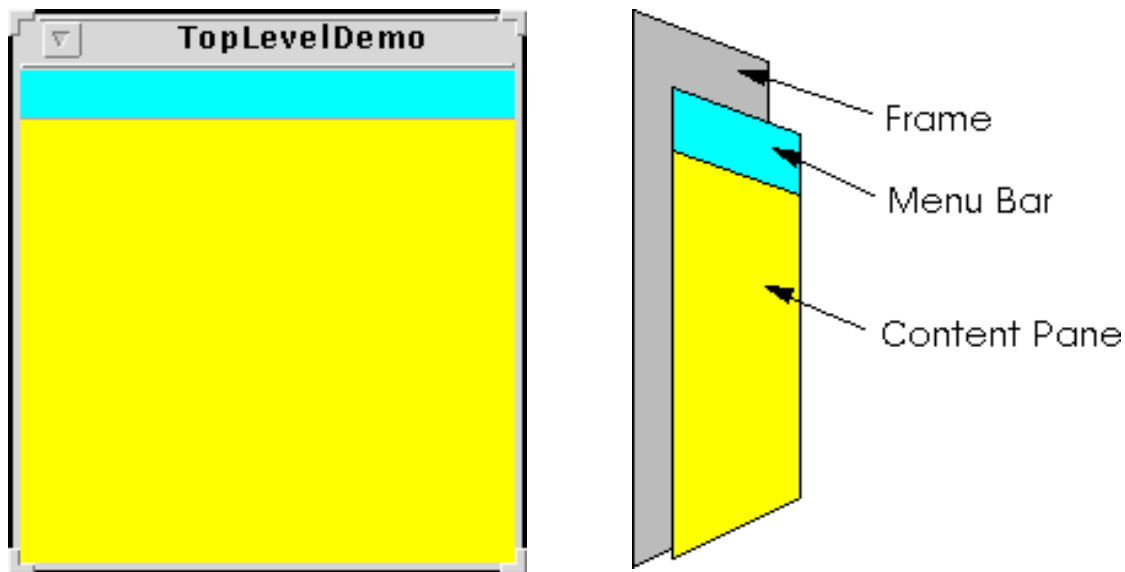
Top-Level

- Swing propose 3 composants top-level: JFrame, JDialog et JApplet
- JWindow est aussi top-level mais il n'est pas utilisé
- JInternalFrame ressemble à un top-level mais il n'en est pas un
- Une application graphique doit avoir un composant top-level comme composant racine (composant qui inclus tous les autres composants)



Top-Level

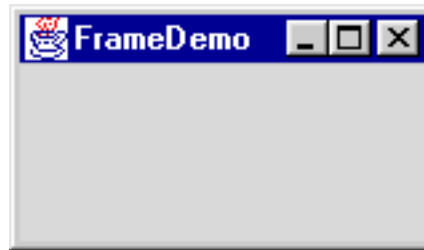
- Les composants top-level possèdent un content pane qui contient tous les composants visibles d'un top-level
- Un composant top-level peut contenir une barre de menu





JFrame

Une JFrame est une fenêtre avec un titre et une bordure



Quelques méthodes :

```
public JFrame();  
public JFrame(String name);  
public Container getContentPane();  
public void setJMenuBar(JMenuBar menu);  
public void setTitle(String title);  
public void setIconImage(Image image);
```




JDialog

Un JDialog est une fenêtre qui a pour but un échange d'information



Un JDialog dépend d'une fenêtre, si celle-ci est détruite, le JDialog l'est aussi

Un JDialog peut aussi être modal, il bloque tout les inputs sur lui



Conteneur Intermédiaire

- Les conteneur intermédiaire sont utilisés pour structurer l'application graphique
- Le composant top-level contient des composants conteneur intermédiaire
- Un conteneur intermédiaire peut contenir d'autre conteneurs intermédiaire



Conteneur Intermédiaire

Swing propose plusieurs conteneurs intermédiaire:

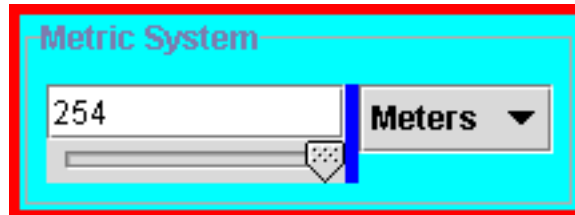
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane
- JToolBar
- ...



JPanel

Le JPanel est le conteneur intermédiaire le plus neutre

On ne peut que choisir la couleur de fond



Quelques méthodes de JPanel:

```
public JPanel();
```

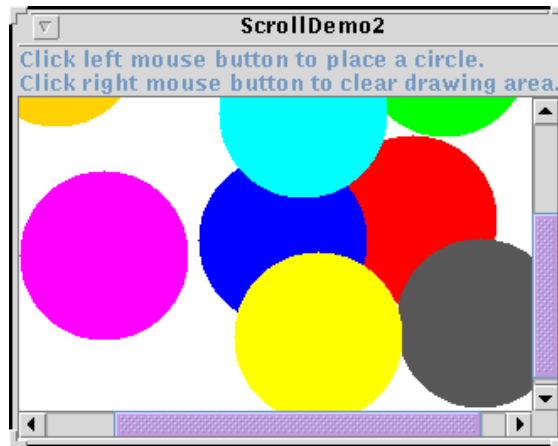
```
public Component add(Component comp);
```

```
public void setLayout(LayoutManager lm);
```



JScrollPane

Un JScrollPane est un conteneur qui offre des ascenseurs, il permet de visionner un composant plus grand que lui



Quelques méthodes:

```
public JScrollPane(Component comp);
```

```
public void setCorner(String key, Component comp);
```



JSplitPane

Un JSplitPane est un panel coupé en deux par une barre de séparation. Un JSplitPane accueille deux composants.



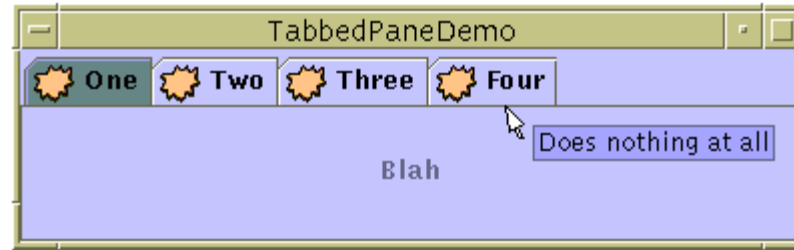
Quelques Méthodes :

```
public JSplitPane(int ori, Component comp, Component c);  
public void setDividerLocation(double pourc);
```



JTabbedPane

Un JTabbedPane permet d'avoir des onglets



Quelques méthodes :

```
public JTabbedPane();
```

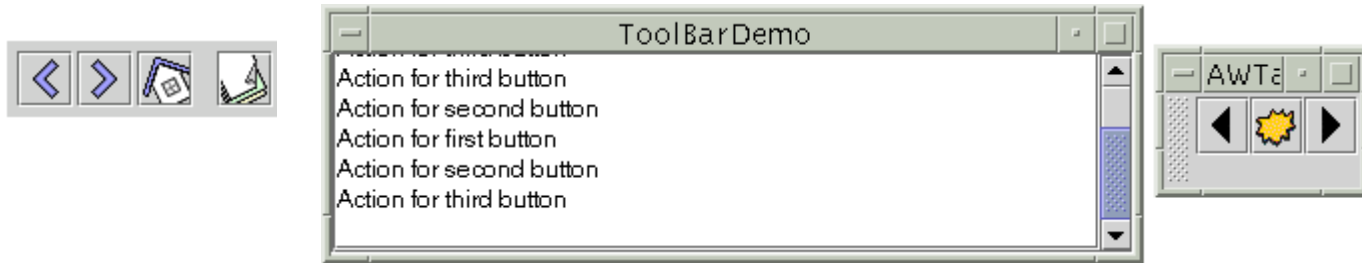
```
public void addTab(String s, Icon i, Component c, String s);
```

```
public Component getSelectedComponent();
```



JToolBar

Une JToolBar est une barre de menu



Quelques Méthodes :

```
public JToolBar();
```

```
public Component add(Component c);
```

```
public void addSeparator();
```



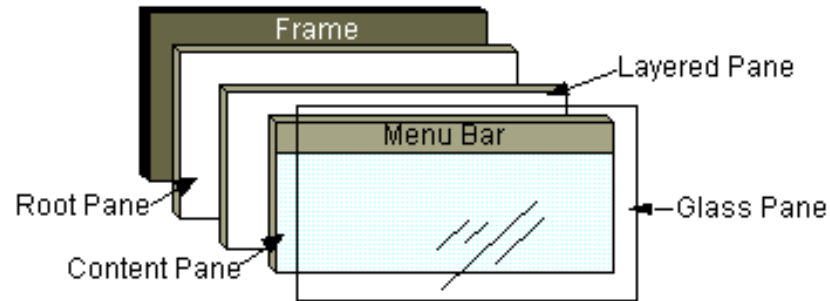

Conteneur Intermédiaire Spécialisé

- Les conteneur Intermédiaire spécialisé sont des conteneurs qui offrent des propriétés particulières aux composants qu'ils accueillent
 - JRootPane
 - JLayeredPane
 - JInternalFrame



JRootPane

En principe, un JRootPane est obtenu à partir d'un top-level ou d'une JInternalFrame



Un JRootPane est composé de

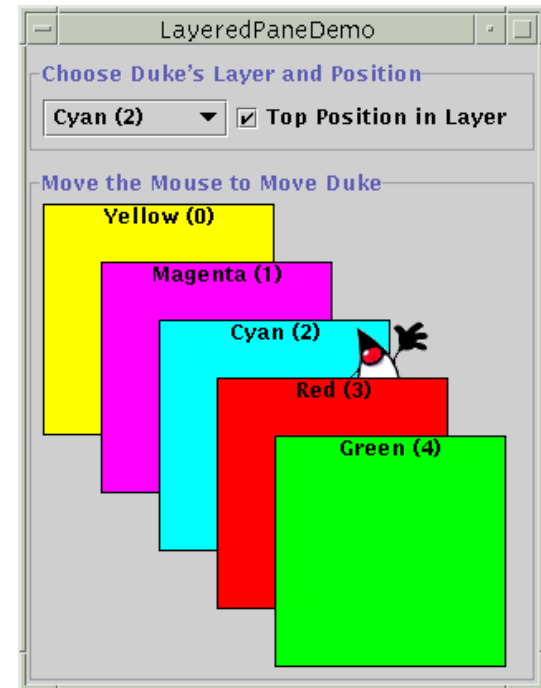
- glass pane
- layered pane
- content pane
- menu bar



JLayeredPane

Un JLayeredPane permet de positionner les composants dans un espace à trois dimensions

Pour ajouter un Composant:
`add(Component c, Integer i);`

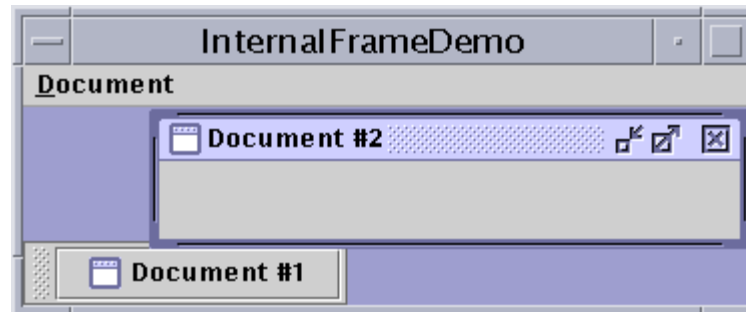




JInternaleFrame

Un JInternaleFrame permet d'avoir des petites fenêtres dans une fenêtre.

Une JInternaleFrame ressemble très fortement à une JFrame mais ce n'est pas un container Top-Level





Les composants atomiques

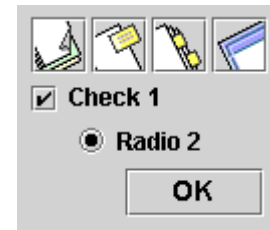
- Un composant atomique est considéré comme étant une entité unique.
- Java propose beaucoup de composants atomiques:
 - boutons, CheckBox, Radio
 - Combo box
 - List, menu
 - TextField, TextArea, Label
 - FileChooser, ColorChooser,
 - ...



Les boutons

Java propose différent type de boutons:

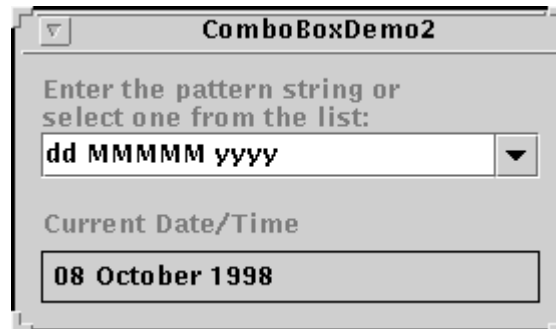
- Le bouton classique est un JButton.
- JCheckBox pour les case à cocher
- JRadioButton pour un ensemble de bouton
- JMenuItem pour un bouton dans un menu
- JCheckBoxMenuItem
- JRadioButtonMenuItem
- JToggleButton Super Classe de CheckBox et Radio





JComboBox

Un JComboBox est un composant permettant de faire un choix parmi plusieurs propositions.



Quelques méthodes:

```
public JComboBox(Vector v);
```

```
public JComboBox(ComboBoxModel c);
```

```
Object getSelectedItem();
```

```
void addItem(Object o);
```



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 srolled pane

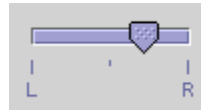
Quelques méthodes:
`public JList(Vector v);`
`public JList(ListModel l);`
`Object[] getSelectedValues();`





JSlider

Les JSlider permettent la saisie graphique d'un nombre
Un JSlider doit contenir les bornes max et min



Quelques méthodes:

```
public JSlider(int min ,int max, int value);  
public void setLabelTable(Dictionary d);
```

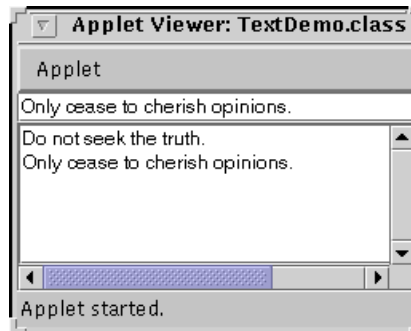


JTextField

Un JTextField est un composant qui permet d'écrire du texte.

Un JTextField a une seule ligne contrairement au JTextArea

Le JPasswordField permet de cacher ce qui est écrit



Quelques méthodes:

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



JLabel

Un JLabel permet d'afficher du texte ou une image.

Un JLabel peut contenir plusieurs lignes et il comprend les tag HTML.



Quelques méthodes:

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



Les menu

Si on a une barre de menu JMenuBar, on ajoute des JMenu dans la barre.

Les JMenu et le JPopupMenu ont le même mode de fonctionnement, créer des JMenuItem et les ajouter.



Ex:

```
menuBar = new JMenuBar();  
setJMenuBar(menuBar);  
menu = new JMenu("A Menu");  
menuBar.add(menu);  
menuItem = new JMenuItem("menu item");  
menu.add(menuItem);
```

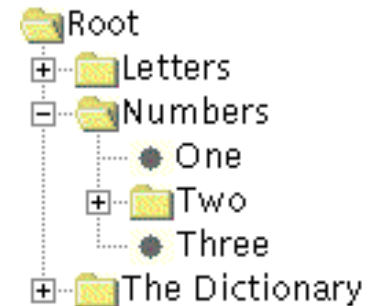


JTree

Un JTree permet d'afficher des informations sous forme d'arbre. Les nœuds de l'arbre sont des objets qui doivent implanter l'interface MutableTreeNode. Une classe de base est proposée pour les nœuds : DefaultMutableTreeNode. Pour construire un arbre il est conseillé de passer par la classe TreeModel qui est la représentation abstraite de l'arbre.

Pour construire un arbre:

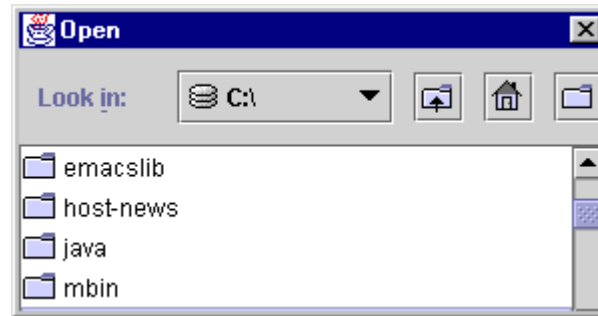
```
rootNode = new DefaultMutableTreeNode("Root");  
treeModel = new DefaultTreeModel(rootNode);  
tree = new JTree(treeModel);  
childNode = new DefaultMutableTreeNode ("Child");  
rootNode.add(childNode);
```





JFileChooser

Un JFileChooser permet de sélectionner un fichier en parcourant l'arborescence du système de fichier.



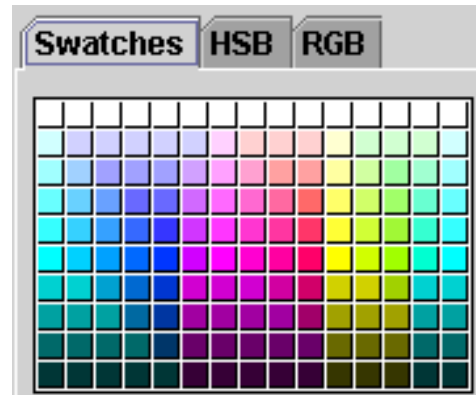
Ex :

```
JFileChooser fc = new JFileChooser();  
int returnVal = fc.showOpenDialog(aFrame);  
if (returnVal == JFileChooser.APPROVE_OPTION) {  
    File file = fc.getSelectedFile();  
}
```



JColorChooser

Un JColorChooser permet de choisir une couleur



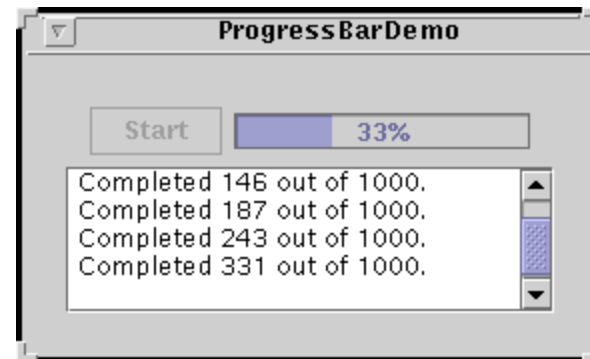
Une méthode :

```
public static Color showDialog(Component c , String title , Color initialColor);
```



JProgressBar

Un JProgressBar permet d'afficher une barre de progression.



Quelques méthodes :

```
public JProgressBar();
```

```
public JProgressBar(int min, int max);
```

```
public void setValue(int i);
```




Positionnement des composants

Les layout manager



Architecture de Layout

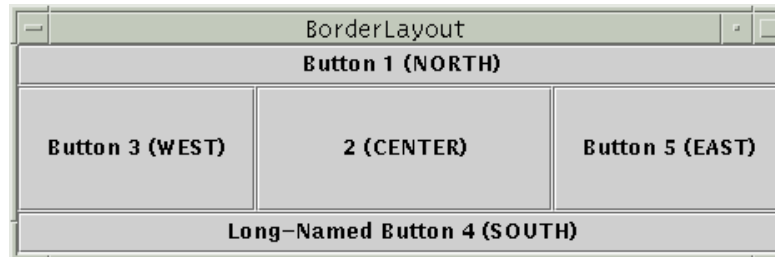
- Pour placer des composants dans un container, Java propose une technique de Layout.
- Un layout est une entité Java qui place les composants les uns par rapport aux autres.
- Le layout s'occupe aussi de réorganiser les composants lorsque la taille du container varie.
- Il y a plusieurs layout : BorderLayout, BoxLayout, CardLayout, FlowLayout, GridLayout, GridBagLayout.
- Un layout n'est pas contenu dans un container, il gère le positionnement.



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 zones prennent uniquement l'espace qui leur est nécessaire.



Ex :

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



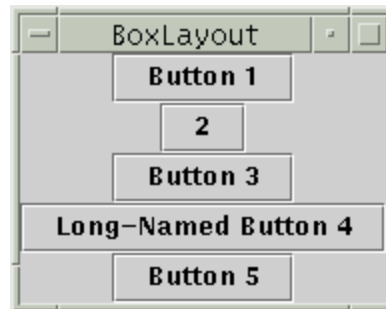
BoxLayout

Un `BoxLayout` permet d'empiler les composants du container (soit de verticalement, soit horizontalement)

Ce layout essaye de donner à chaque composant la place qu'il demande

Il est possible de rajouter des blocs invisible.

Il est possible de spécifier l'alignement des composants (centre, gauche, droite)





CardLayout

Un CardLayout permet d'avoir plusieurs conteneurs ; les uns au dessus des autres (comme un jeu de cartes).



Ex :

```
JPanel cards;
```

```
final static String BUTTONPANEL = "JPanel with JButtons";
```

```
final static String TEXTPANEL = "JPanel with JTextField";
```

```
cards = new JPanel();
```

```
cards.setLayout(new CardLayout());
```

```
cards.add(p1,BUTTONPANEL);
```

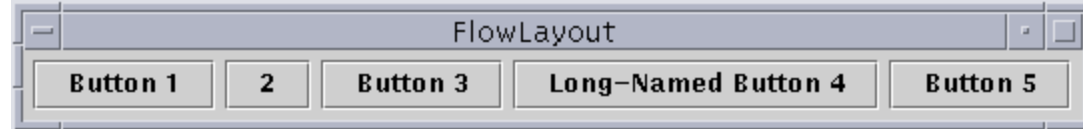
```
cards.add(p2,TEXTPANEL);
```



FlowLayout

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

Le FlowLayout est le layout par défaut des JPanel



Ex :

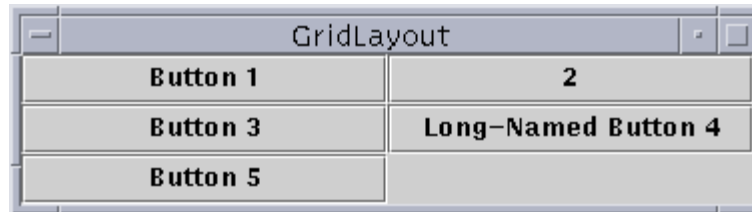
```
Container contentPane = getContentPane();  
contentPane.setLayout(new FlowLayout());
```

```
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"));
```



GridBagLayout

Le GridBagLayout est le layout le plus complexe. Il place les composants sur une grille, mais des composants peuvent être contenus dans plusieurs cases.

Pour exprimer les propriétés des composants dans la grille, on utilise un GridBagConstraints.

Un GridBagConstraints possède :

- gridx, gridy pour spécifier la position
- gridwidth, gridheight pour spécifier la place
- fill pour savoir comment se fait le remplissage
- ...

Ex :

```
GridBagLayout gridbag = new GridBagLayout();
GridBagConstraints c = new GridBagConstraints();
JPanel pane = new JPanel();
pane.setLayout(gridbag);
gridbag.setConstraints(theComponent, c);
pane.add(theComponent);
```





Création de Layout

- Il est possible de construire son propre Layout
- Un layout doit implanter l'interface `java.awt.LayoutManager` ou `java.awt.LayoutManager2`