Swing et Java2D

X.Blanc
Xavier.Blanc@labri.fr

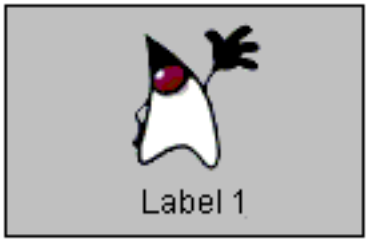
Swing

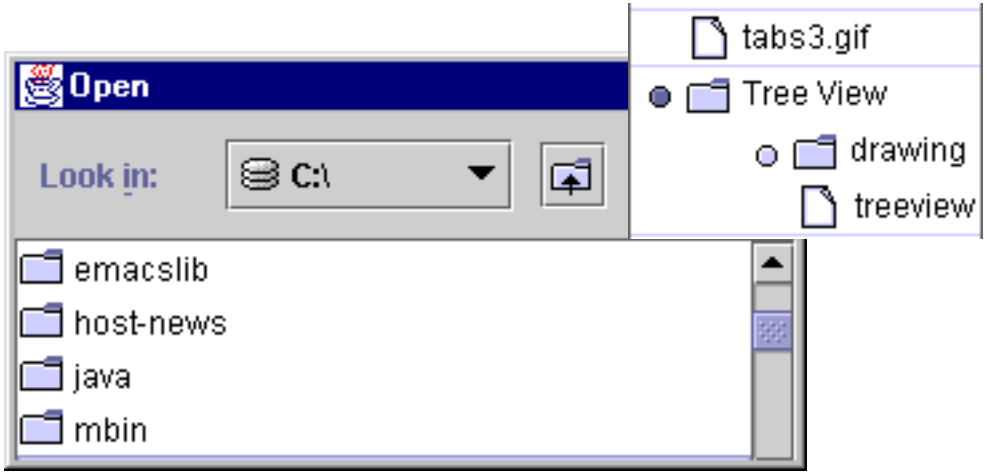
Qui sont les Swing

Définition de Swing

- Les Swing sont utilisés pour faire des interfaces graphiques
- Les Swing sont des composants (bouton, fenêtre, label, zone de texte …)
- Tous ces composants sont regroupés dans le package Swing
- *Les Swing sont des JavaBeans*

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

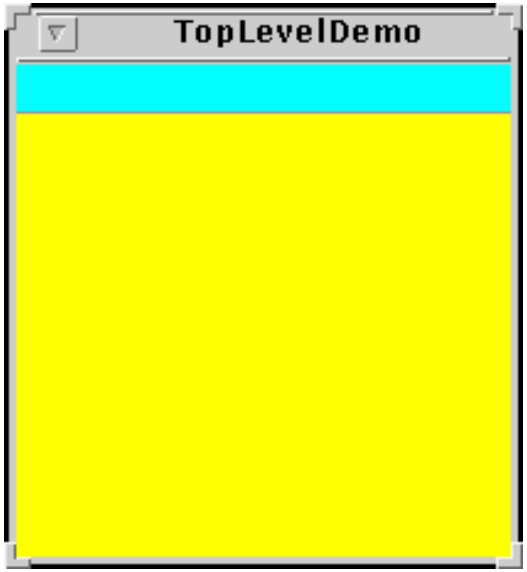
- Tous les composants Swing héritent de JComponent
- Les composants ont des Tool Tips
- Les composants ont des bordures
- Entité graphique la plus abstraite

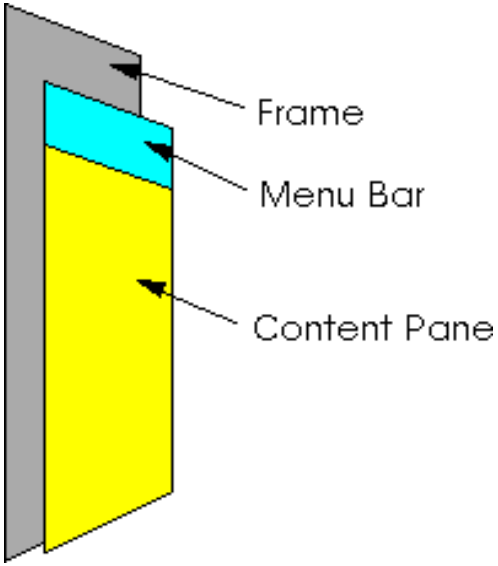
Top-Level

- 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

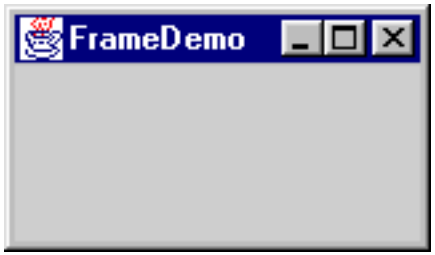
- Les composants top-level possèdent un content pane qui contient tous les composants visibles d'un top-level
- Un composant top-level peut contenir une barre de menu





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);
public setDefaultCloseOperation
                    (JFRAME.EXIT_ON_CLOSE)
```

Initialisation JFrame

```
//1. Create the frame.
JFrame frame = new JFrame("FrameDemo");

//2. Optional: What happens when the frame closes?
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

//3. Create components and put them in the frame.
//...create emptyLabel...
frame.getContentPane().add(emptyLabel, BorderLayout.CENTER);

//4. Size the frame.
frame.pack();

//5. Show it.
frame.setVisible(true);
```

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 JDilaog 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

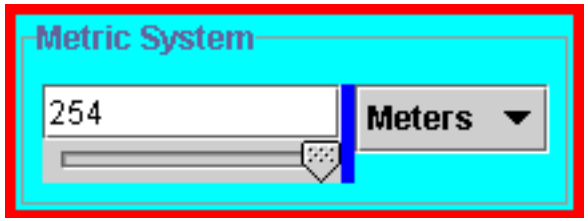
Swing propose plusieurs conteneurs intermédiaire:

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

JPanel

Le JPanel est le conteneur intermédiaire le plus neutre

On ne peut que choisir la couleur de fond

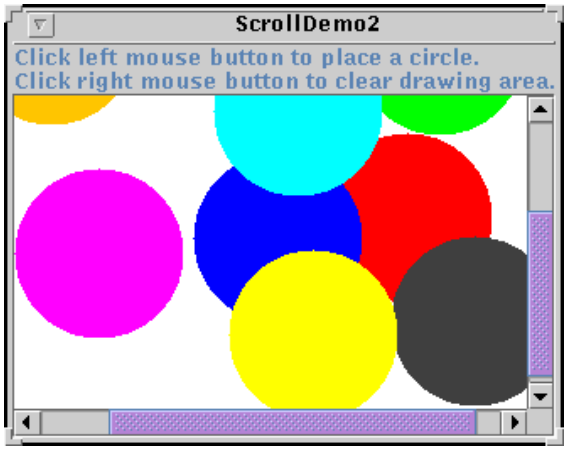


Quelques méthodes de JPanel:

```
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

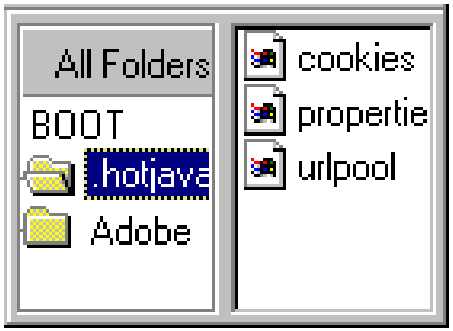


Quelques méthodes:

```
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 accueil deux composants.



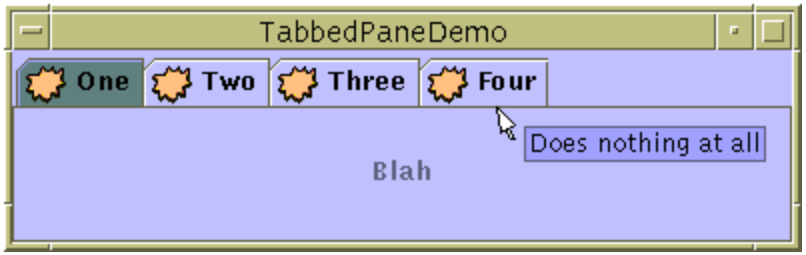
Quelques Méthodes :

public JSplitPane(int ori, Component comp, Component c);

public void setDividerLocation(double pourc);

JTabbedPane

Un JTabbedPane permet d'avoir des onglets

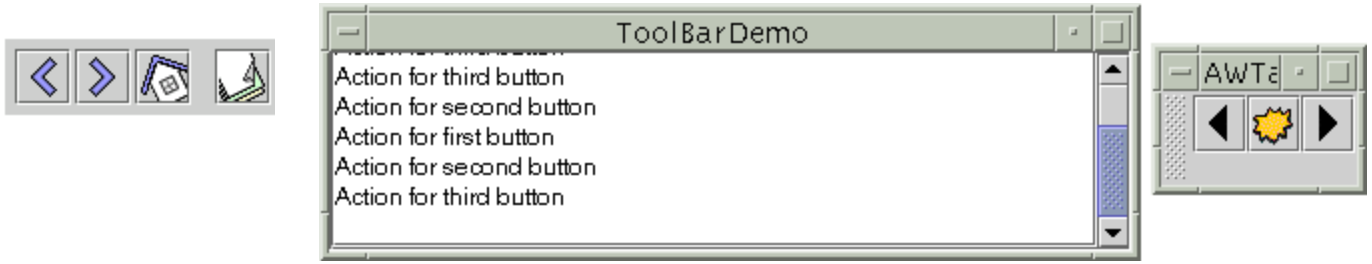


Quelques méthodes :

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

JToolBar

Une JToolBar est une barre de boutons



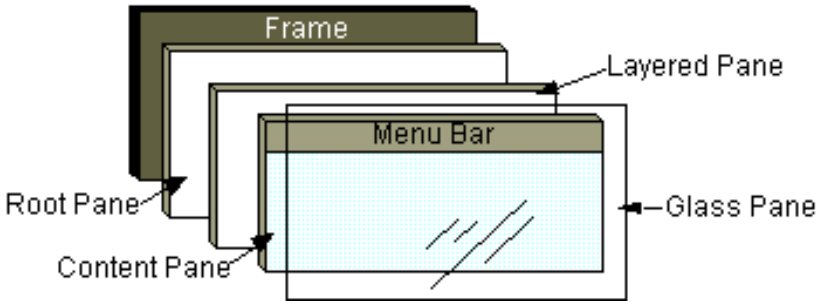
Quelques Méthodes :
public JToolBar();
public Component add(Component c);
public void addSeparator();

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

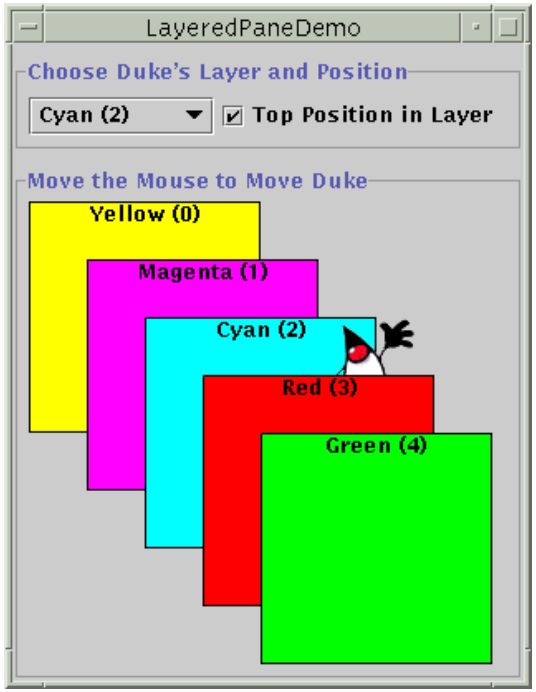


Un JRootPane est composé de

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

JLayeredPane

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



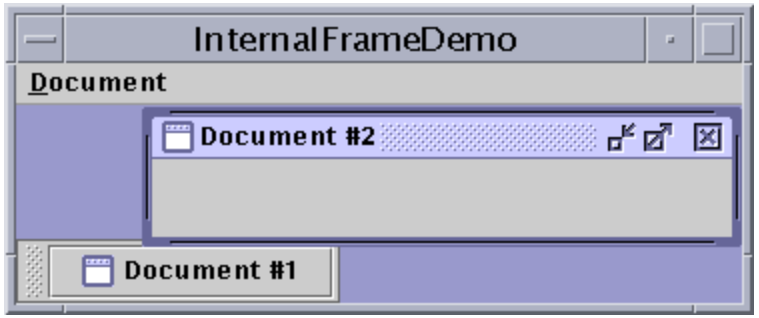
Pour ajouter un Composant:

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

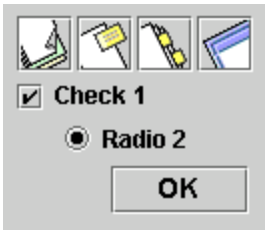


Les composants atomiques

- 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

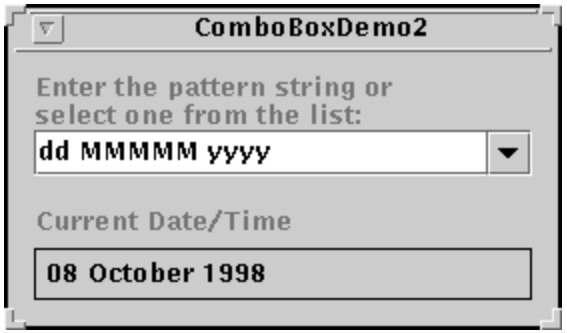
Java propose différent type de boutons:



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

JComboBox

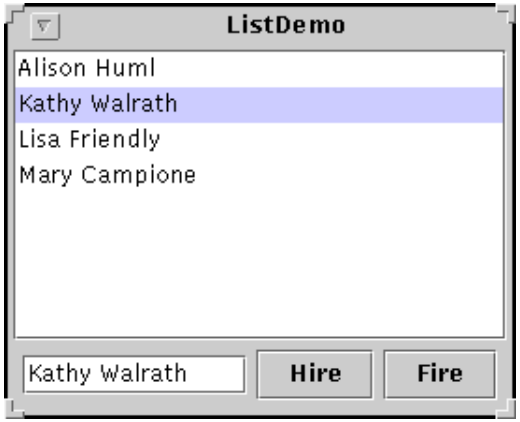
Un JComboBox est un composant permettant de faire un choix parmi plusieurs propositions.



Quelques méthodes:
public JComboBox(Vector v);
public JComboBox(ComboBoxModel c);
Object getSelectedItem();
void addItem(Object o);

JList

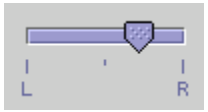
Une JList propose plusieurs éléments rangés en colonne.
Une JList peut proposer une sélection simple ou multiple
Les JList sont souvent contenues dans un srolled pane



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

JSlider

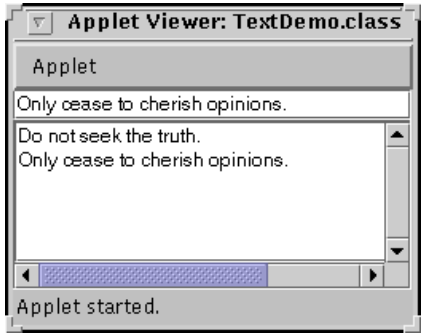
Les JSlider permettent la saisie graphique d'un nombre
Un JSlider doit contenir les bornes max et min



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 seul ligne contrairement au JTextArea
Le JPasswordField permet de cacher ce qui est écrit

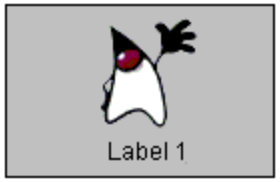


Quelques méthodes:
public JTextField(String s);
public String getText();

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:

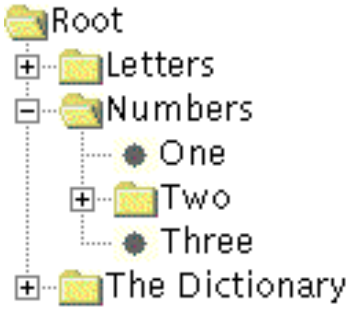
```
menuBar = new JMenuBar();
setJMenuBar(menuBar);
menu = new JMenu("A Menu");
menuBar.add(menu);
menuItem = new JMenuItem("menu item");
menu.add(menuItem);
```

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.

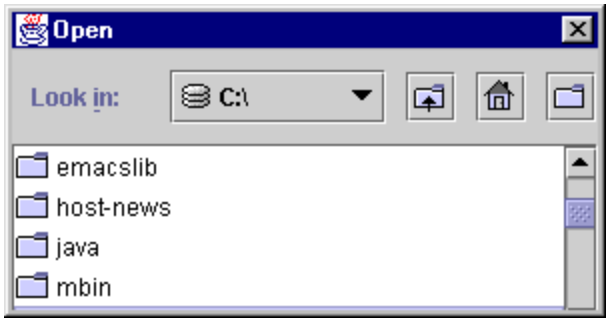
Pour construire un arbre:

```
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.

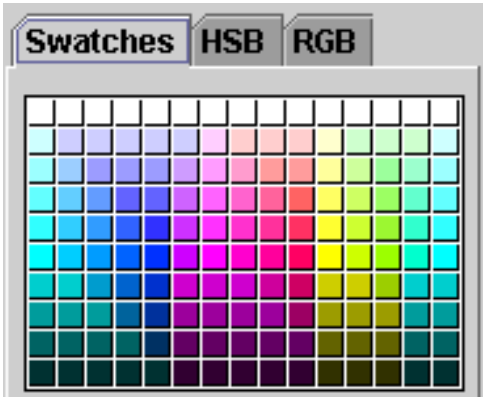


Ex :

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

JColorChooser

Un JColoChooser permet de choisir une couleur

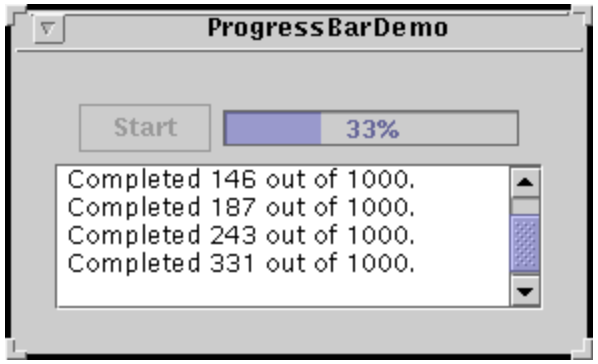


Une méthode :

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

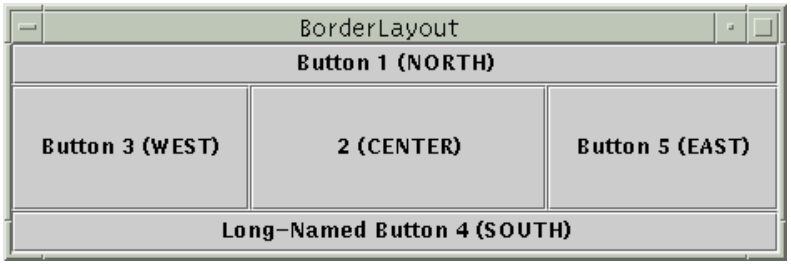
Architecture de Layout

- 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 zone 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);
```

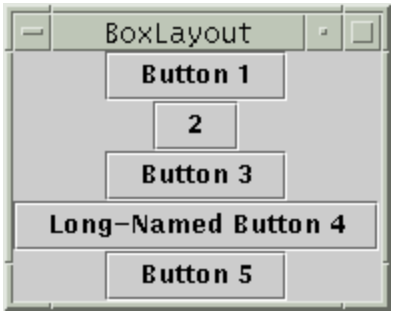
BoxLayout

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 jeux de cartes).

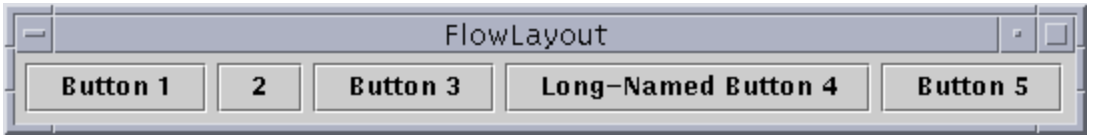
Ex :

```
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



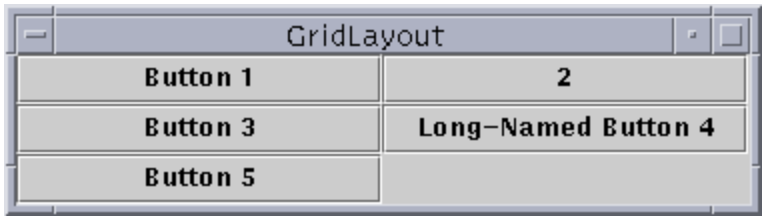
Ex :

```
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 GridBasConstraints 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

Les événements et les Swing

Le comportement de l'interface graphique

Les bases

- Les swing sont des JavaBeans
- Ils communiquent donc grâce aux événements
- Les événements permettent d'associer un traitement à une action

Les événements pour les fenêtre

Un listener d'événement Window permet par exemple de fermer l'application lorsque l'on ferme la fenêtre principale.

Ex :

```
public Monlistener implements java.awt.event.WindowListener {
        public void windowClosing(java.awt.event.WindowEvent e) {
                System.exit(0);
        }
}
```

Pour les boutons

Sur les boutons les événement permettent l'exécution d’un traitement si un click a eu lieux.

Ex :

```
public MonListener implements java.awt.event.ActionListener {
        public void actionPerformed(ActionEvent e) {
                System.out.println(« le bouton a été clické »);
        }
        public MonListener() {
        }
}
```

Dans le programme :

```
JButton b = new JButton(« Click Me ! »);
b.addActionListener(new MonListener());
```

Architecture des Listener

- Centralisée
  - Un seul listener
  - Il répond à tous les
  - évènements
  - Il faut filtrer en fonction de la source et de l'état de l'application
- Décentralisée
  - Un listener par évènement
  - Il répond uniquement à l'évènement

Look & Feel

Paramétrer son application

Look & Feel

- Le Look & Feel permet de d'arranger l'apparence des composants Swing
- Java propose plusieurs look différents dont :
  - Style Windows
  - Style Motif
  - Style Java Swing
- Une fenêtre avec le style Windows ressemblera aux fenêtres Microsoft Windows

Choisir son Look & Feel

try {
UIManager.setLookAndFeel(
UIManager.getSystemLookAndFeelClassName());
}

catch (Exception e) { }

D'autres arguments possibles :

```
UIManager.getCrossPlatformLookAndFeelClassName()
UIManager.getSystemLookAndFeelClassName()
"javax.swing.plaf.metal.MetalLookAndFeel"
"com.sun.java.swing.plaf.motif.MotifLookAndFeel »
"javax.swing.plaf.mac.MacLookAndFeel »
"com.sun.java.swing.plaf.windows.WindowsLookAndFeel"
```

Et en plus

Les bonus des Swing

Les actions

- Action est une interface qui hérite de ActionListener
- En plus d'être un Listener, un action centralise une icône et du texte
- Associer une action à un bouton, permet au bouton d'avoir une icône, du texte et un listener.
- Exemple possible d'action : load, save, cut …
- Certain container (comme le JToolBar) permette directement l'ajout d'action

Les threads

- Un seul Thread gère tout les événements : event-dispatching thread
- Les Swing ne peuvent être modifié en principe que par ce Thread.
- « Once a Swing component has been realized, all code that might affect or depend on the state of that component should be executed in the event-dispatching thread »
- Lorsque l'on veut faire des Threads avec les swing il est conseillé d'utiliser soit le **SwingWorker**, soit un **Timer** soit la classe **SwingUtilities**

Introduction à Java2D

Une API pour dessiner

Objectifs

- Dessiner des points et des lignes
- Remplir des formes
- Ecrire du texte
- Afficher des images
- Réaliser des transformations graphiques

Rendering

- Les objets graphiques sont placés dans un plan 2D abstrait
  - User Space
- Lors de l'affichage, les objets graphiques apparaissent sur un plan concret
  - Device Space

$\Rightarrow$ Le rendering permet l'affichage des objets utilisateur

$\Rightarrow$ De l'User Space vers le Device Space

Graphics2D

- Un objet de type Graphic2D est permet de dessiner toutes les formes
  - draw, fill,
- Il permet de préciser
  - la taille du trait (stroke)
  - La façon dont les traits sont joints
  - La façon dont les formes sont remplies
  - Les fontes
  - Les découpes d'image
  - Les transformations (rotation, etc.)

Comment utiliser Graphics2D

- Faire une classe qui hérite de JComponent
- Polymorphisme sur
  - **protected void paintComponent(Graphics arg0)**
- Caster arg0 en Graphics2D
- Utiliser Graphics2D
- Appeler super.paintComponent(arg0)

Classes de base

- Point
  - setLocation()
- Line2D
- QuadCurve2D, CubicCurve2D
  - Double, Float
- Rectangle2D
- Ellipse2D
- Arc2D

- Toutes ces classes existent en Double et Float Précision
  - Point2D.Double, Line2D.Double

GeneralPath

- Permet de construire des formes complexes
  - moveTo(float x, float y) – Moves the current point of the path to the given point
  - lineTo(float x, float y) – Adds a line segment to the current path
  - quadTo(float ctrlx, float ctrly, float x2, floaty2) – Adds a quadratic curve segment to the current path
  - curveTo(float ctrlx1, float ctrly1, float ctrlx2, float ctrly2, float x3, floaty3) – Adds a cubic curve segment to the current path
  - …
  - closePath() – Closes the current path

Filling and Stroking

- Stroke : permet de spécifier la nature du trait
  - La classe BasicStroke permet de spécifier les informations de base du trait
- Paint : permet de spécifier la nature du remplissage
  - Color, GradientPaint, TexturePaint

- Graphics2D g
- g.setStroke()
- g.setPaint()

Texte

- L'API Graphic permet
  - De spécifier la fonte utilisée
  - De charger des familles de fonte
  - De mesurer la taille d'un texte en fonction de la fonte
  - D'afficher du texte
  - De gérer l'antialiasing
  - …

Image

- L'API Graphics permet
  - De charger des images
  - D'afficher des images
  - D'effectuer des opérations de transformation sur les images
  - De sauvegarder des images

Et encore

- Gestion des imprimantes
- Opération de transformations (rotation, etc.)
- Clipping
- Composition de graphiques
- Interaction utilisateur