Vos premiers pas en Swing

Partie structurelle

Audrey Occello & AM Dery pinna@polytech.unice.fr

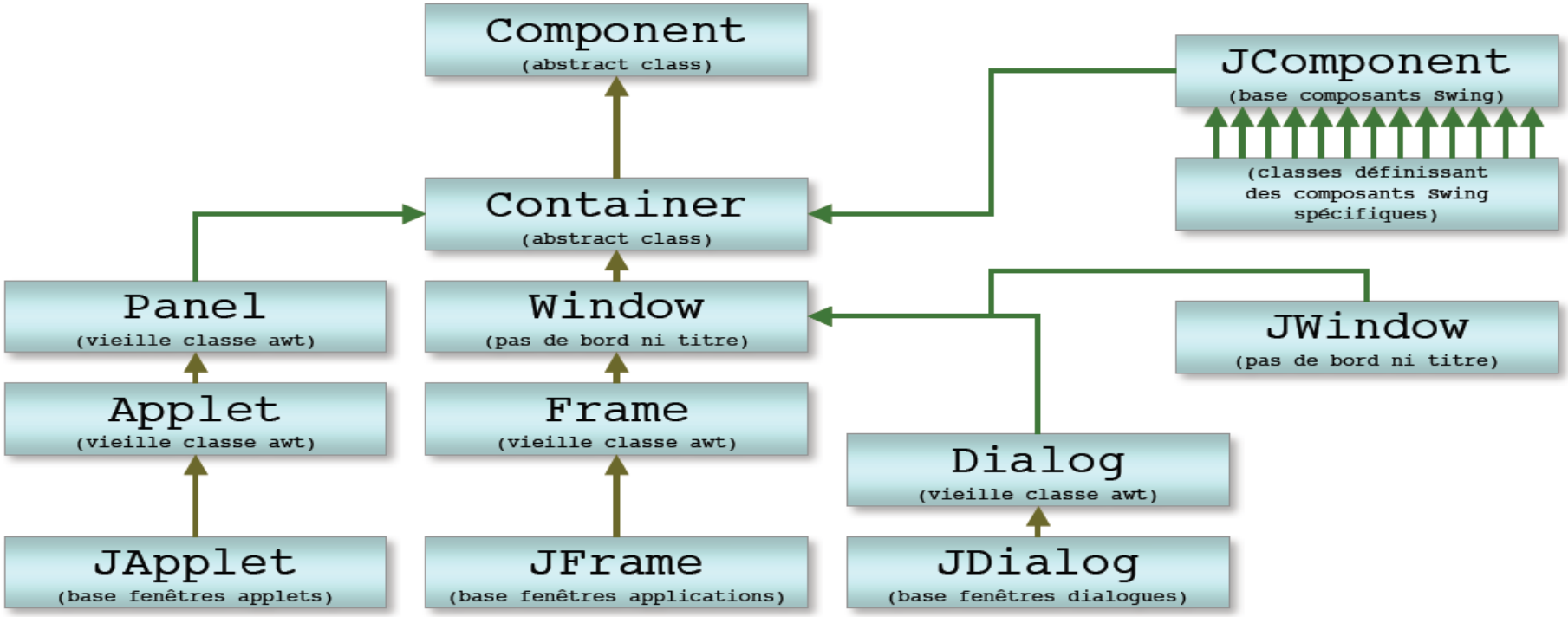
Année 2015 2016

Questions préalables

- Quelles sont les différentes « catégories » de classes dans Swing pour construire l'Interface ?

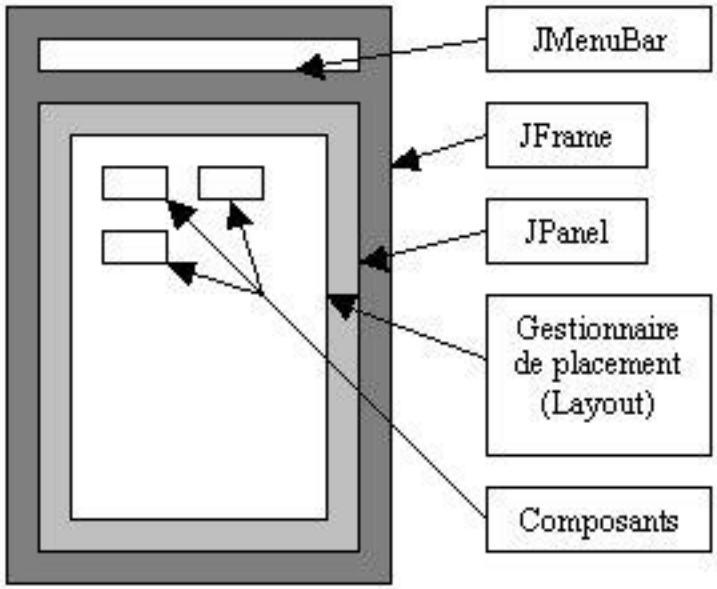
- Quel est le lien entre Swing et AWT ?

HIÉRARCHIE DES CLASSES



- Swing est une boite à outils permettant la création d'interfaces utilisateur en Java
- SWING dépend historiquement de AWT (Abstract Window Toolkit) qui était la boite à outils d'origine tout en s'appuyant dessus.
- Amélioration : Les IHM SWING sont indépendantes de l'OS et de l'environnement graphique sous-jacents

PRINCIPES GÉNÉRAUX



IL PEUT Y AVOIR PLUSIEURS PANNEAUX DANS UNE FENETRE

- **Pour la création des composants :**
  1. Choix du container racine
  2. Choix du panneau dans lequel on dispose les composants (contenu)

- **Pour l'affichage des composants :**
  1. Utilisation du layout par défaut ou choix d'une mise en page des composants du container
  2. Association de l'afficheur (*LayoutManager*) au container

PRINCIPES GÉNÉRAUX

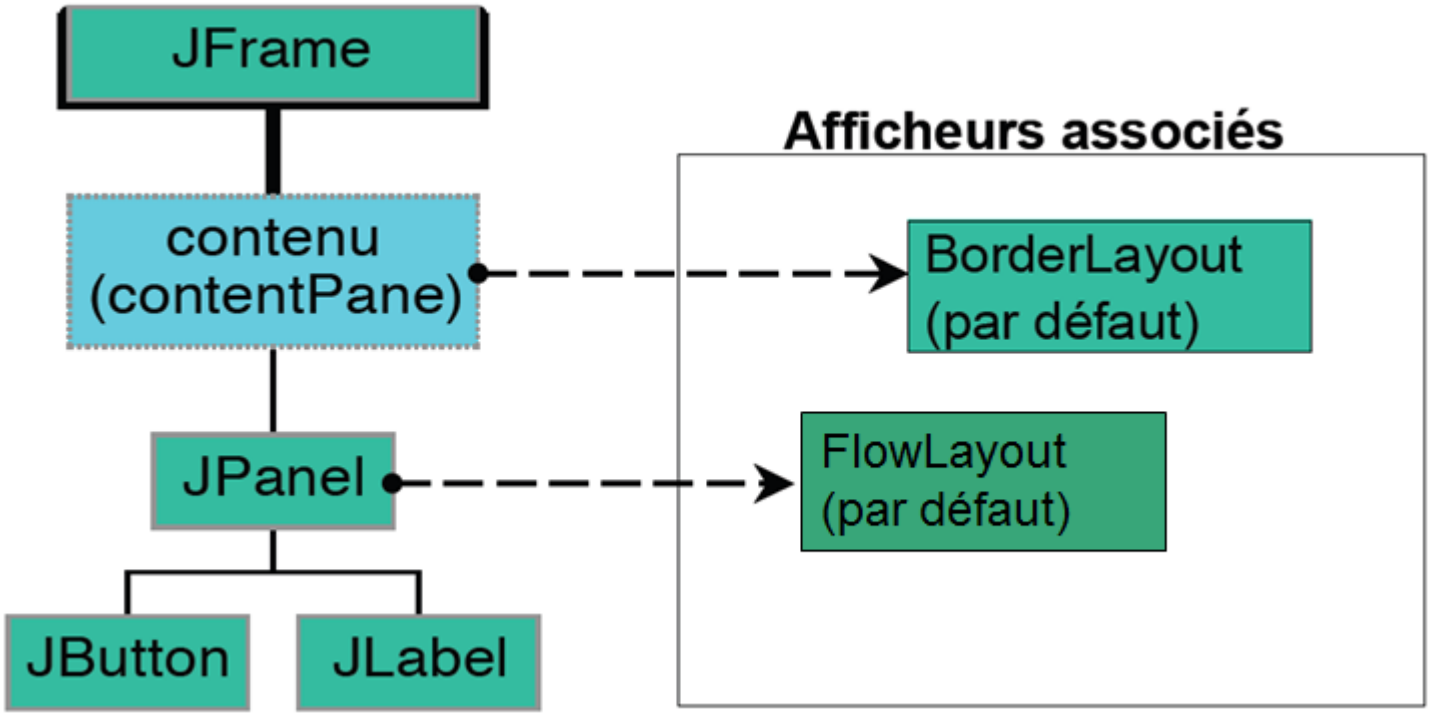
- L'ajout d'un composant à un composant (container) implique la construction d'un sous arbre - le composant ajouté est le fils

- Le parent contient les composants qui lui sont ajoutés

- La méthode *getParent()* sur un composant retourne le parent (de type *Container*)

On a donc une arborescence des composants

Exemple : ARBRE DE COMPOSANTS



Composants : Contrôles

COMMENT SAISIR & AFFICHER DES DONNEES ET PRESENTER DES ACTIONS ?

Je vous écoute….

Aperçu de Swing
Des composants de contrôle

Jcomponent

Méthodes de commodité
- getSize retourne une Dimension
- setSize : une Dimension ou deux entiers

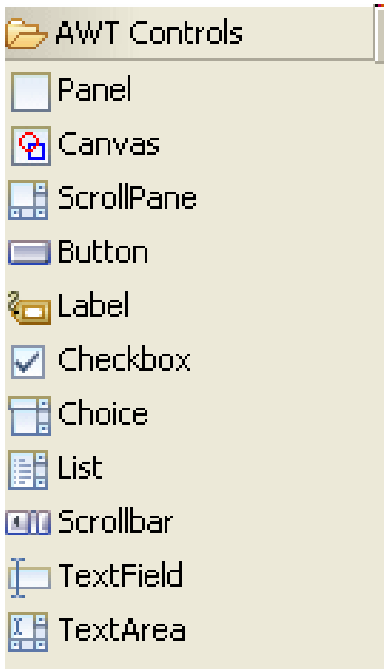
- getLocation retourne un Point
- setLocation avec un Point ou deux entiers

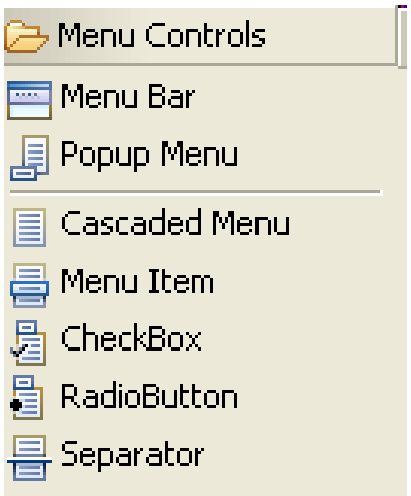
- Origine au coin supérieur gauche
  - x (width) vers la droite
  - y (height) vers le bas

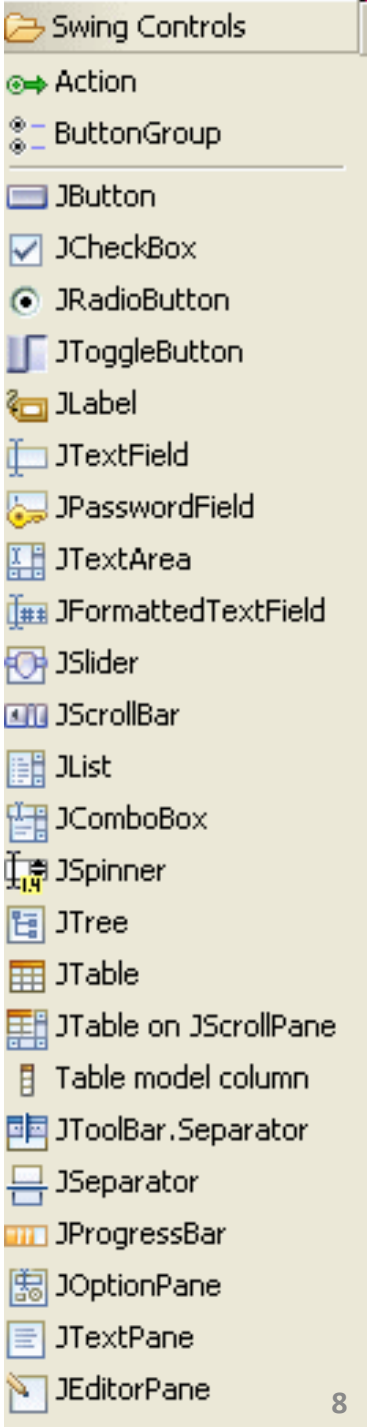
- setPreferredSize : Dimension
- setMinimumSize : Dimension
- setMaximumSize : Dimension

- setBackground : Color

Et bien d'autres !





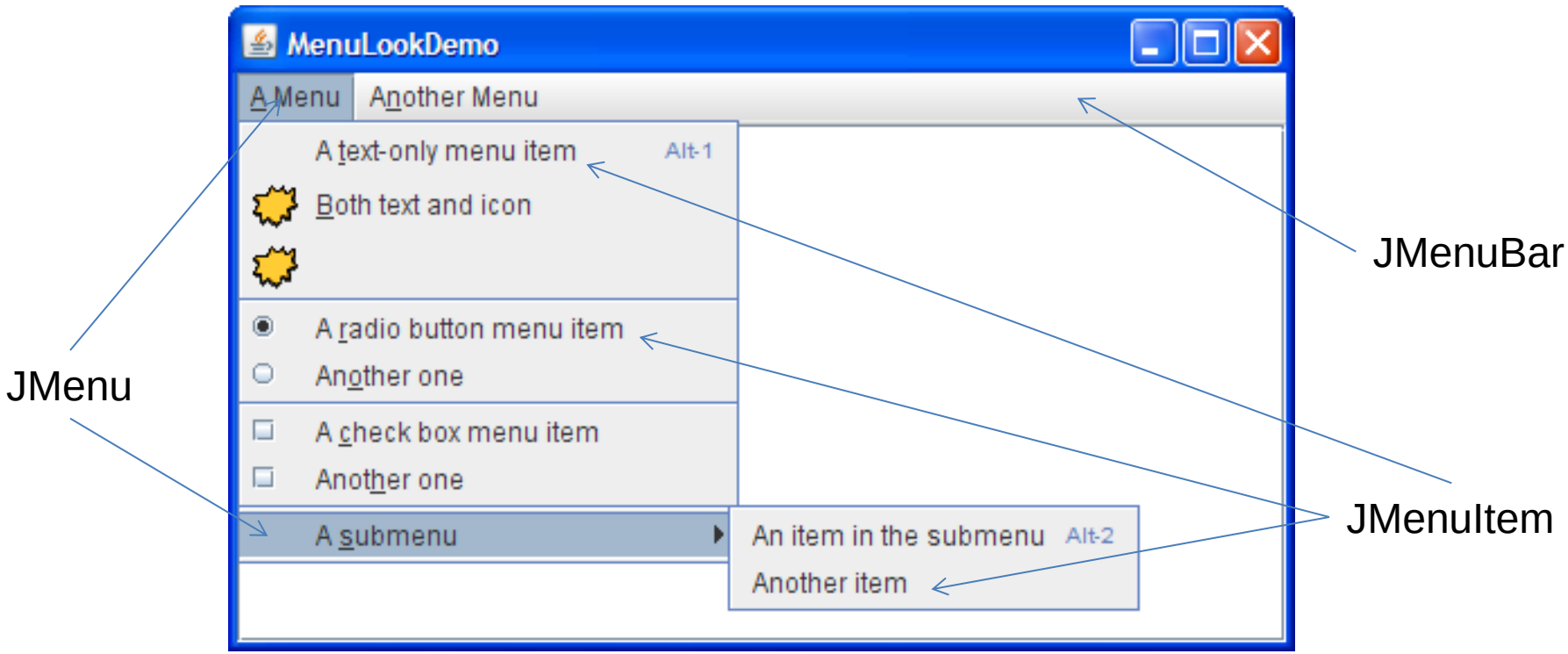


JLABEL

- Permet de définir un label
- Peut comporter : une icône, du texte ou les deux

- Méthodes intéressantes :
  - *setEnabled(boolean)* : pour activer/désactiver la zone (permet de griser le label)
  - *setHorizontalAlignement(constante)* pour contrôler l'alignement du texte

ILLUSTRATION D'UN MENU



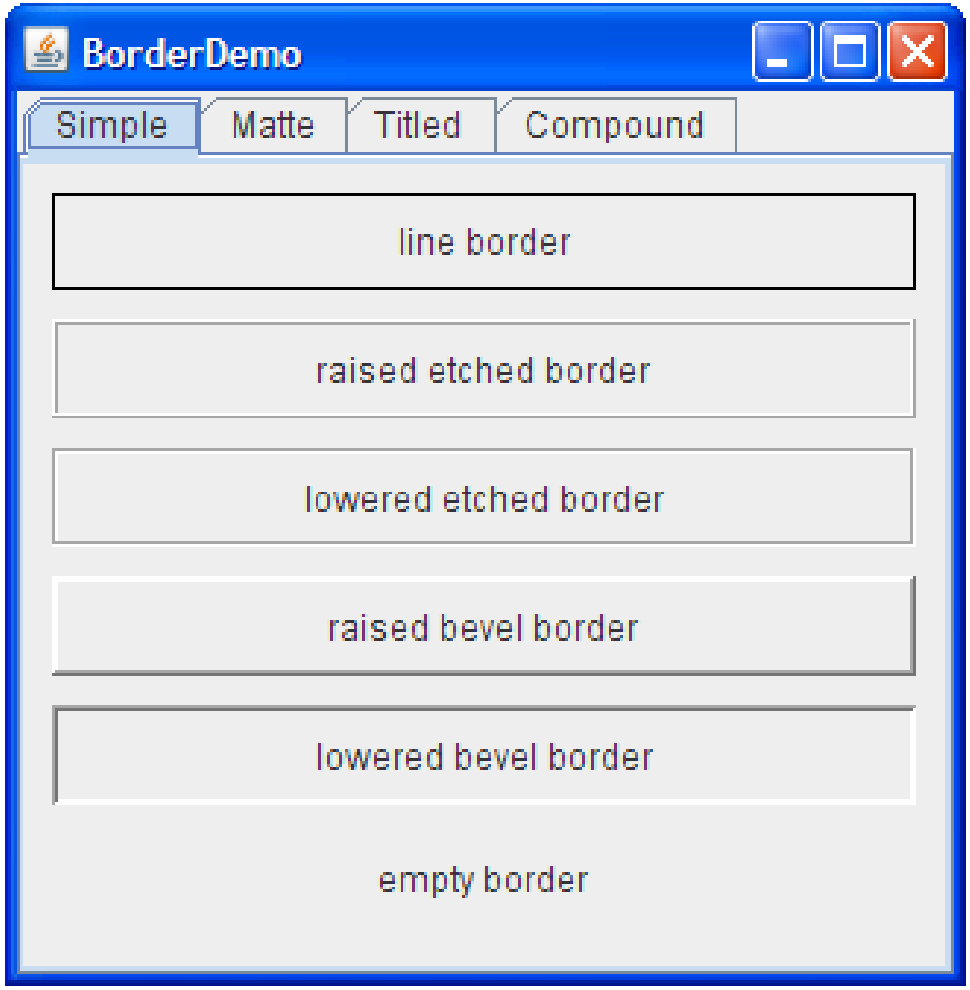
LES BORDURES

- Il est possible d'attribuer une bordure à un *JComponent* :

  setBorder(border)

- Pour obtenir la bordure, on utilise les méthodes statiques de la classe *BorderFactory* :

  Factory.createEtchedBorder()



Containers

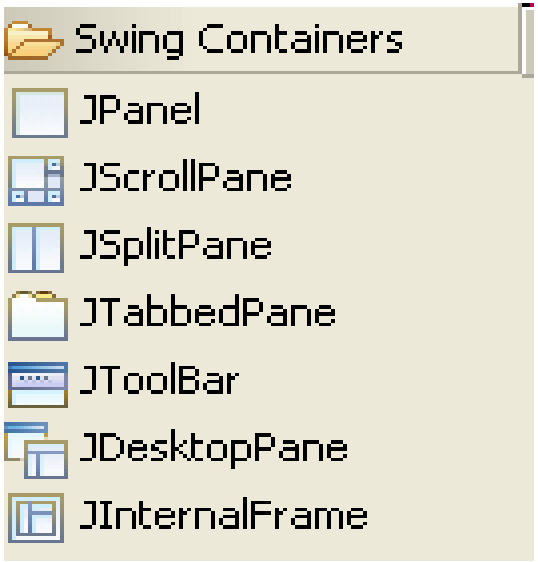
COMMENT GROUPER LES COMPOSANTS D'IHM ?

Je vous écoute…

Aperçu de Swing
Les Containers

Les containers

– add / remove d'un Component
– unicité de lieu
– indice des components
– association à un LayoutManager



Méthodes à connaître

- repaint() / revalidate() : mise à jour de l'IHM
- getBounds / setBounds(x,y, w, h) : positionne et dimensionne
- getWidth() : largeur / getHeight() : hauteur
- getX() et getY() : obtenir une coordonnée (coin haut gauche)
- setVisible(true / false)
- getBackground et setBackground [objet Color, définition RGB]

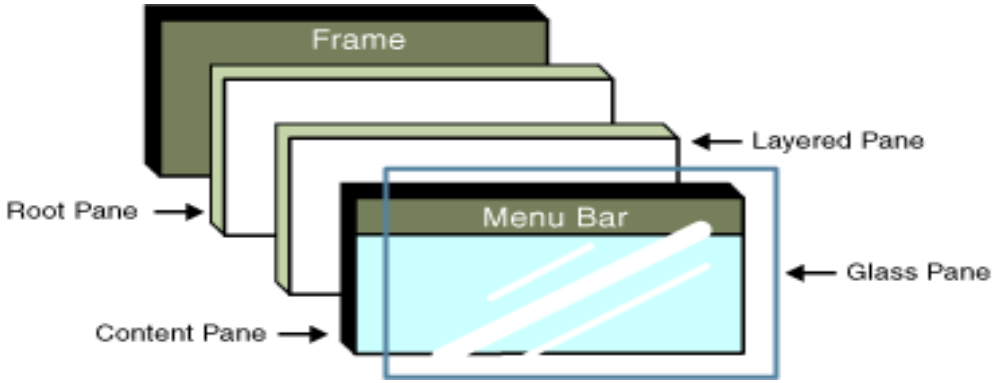
DES CONTAINERS RACINE

- ***JFrame*** : le cadre principal d'une application. Il peut contenir des menus et d'autres composants.

- ***JDialog*** : une fenêtre de dialogue avec l'utilisateur. Son « parent » (owner des constructeurs) sert essentiellement à la placer dans l'écran.

- ***JApplet*** : classe de base pour les applets Java 2. Les applets sont des applications Java pouvant s'executer à l'intérieur d'un navigateur Web.
  Plus vraiment utilisé … Les RIA ont pris le marché !

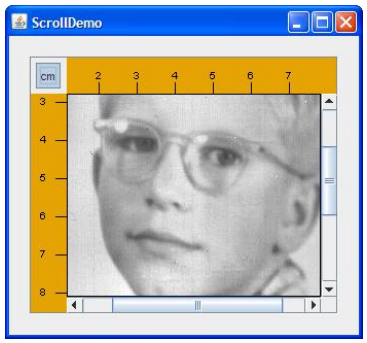
LES DIFFÉRENTS PANNEAUX D'UN CONTAINER RACINE

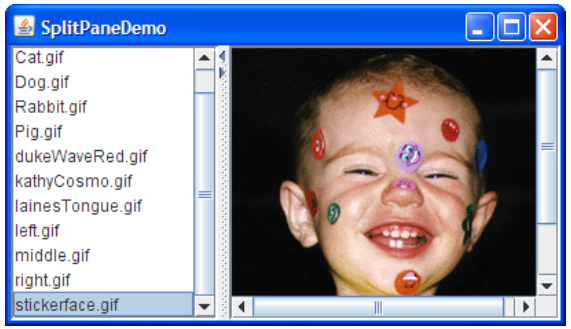


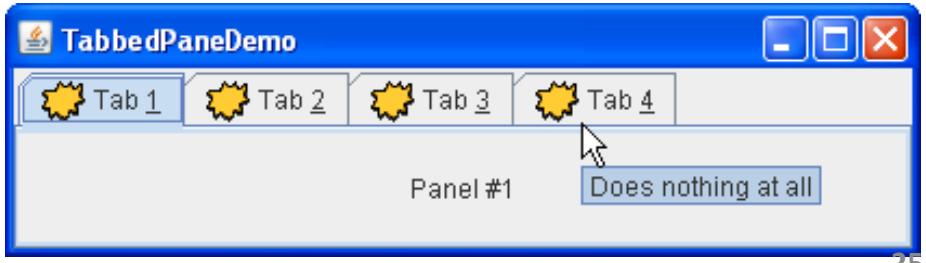
- S'obtiennent par :
  - *getRootPane* () zone de la fenêtre sous le titre
  - *getLayeredPane()* zone où sont cachés les panneaux non visibles
  - *getContentPane* () zone où les éléments sont ajoutés
  - *getGlassPane* () zone transparente dessinée au-dessus du *JRootPane* utilisé pour afficher des pop-up menus

AUTRES CONTAINERS
La famille des panneaux

- Pour structurer les différents éléments graphiques, on utilise des containers qui font partis de la famille des panneaux.
- Quelques panneaux très utiles :
  - Le ***JPanel*** : panneau de base,
  - Le ***JScrollPane*** : qui permet d'obtenir des ascenseurs
  - Le ***JSplitPane*** : qui permet de diviser en 2 (seulement 2)
  - Le ***JTabbedPane*** : qui permet d'avoir différents onglets pour les différents sous-contenus







Afficheurs

COMMENT POSITIONNER LES COMPOSANTS ?

Je vous écoute….

DEUX MANIÈRES

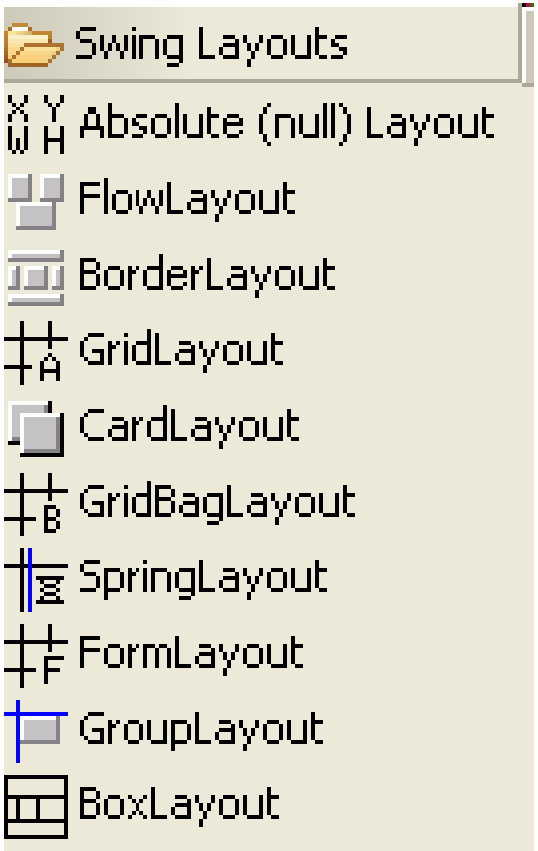
- Positionner de manière absolue, revient à :
  - Retirer tout gestionnaire d'affichage (*setLayout(null)*)
  - Placer un par un les éléments en leur donnant leur position dans la fenêtre (méthode *setBounds*)
  - Du «sur mesure» mais peut s'avérer très complexe à définir !

- Positionner de manière relative en utilisant des gestionnaires d'affichage

MISE EN PAGE RELATIVE

- La mise en page des éléments s'effectue à l'aide d'un gestionnaire d'affichage (*Layout*),
- Tous les *containers* possèdent une méthode *setLayout* qui permet de spécifier le type de gestionnaire d'affichage
- On peut ajouter des contraintes lorsqu'on ajoute un composant
- Avantage : Le redimensionnement est géré automatiquement

Aperçu de Swing Layouts



Les Layout : Basé sur PreferredSize ou une maximisation de l'élément

- BorderLayout
  - par défaut dans une fenêtre
  - ajout en précisant la zone
  - add(comp, "North")
- FlowLayout : en ligne(s)
- *BoxLayout* : séquence verticale ou horizontale
- GridLayout : en tableau
- GridBagLayout : avec des contraintes
- *CardLayout* : empilement
- etc.

UTILISATION D'UN LAYOUT

1. Création du *layout*

   Exemple :

   *BorderLayout bl = new **BorderLayout**()*)

2. Association au container : *panel.setLayout(bl)*

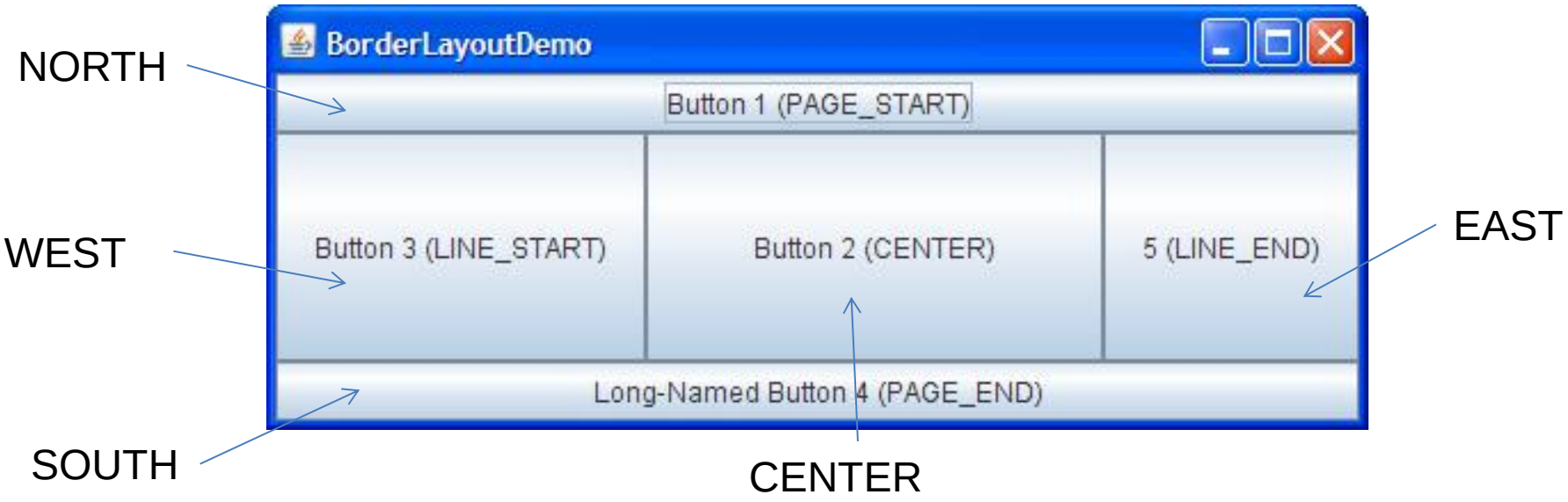
3. Ajout de composants au container, avec ou sans contrainte

   Exemple :

   *panel.add(button, **BorderLayout.EAST**)*

BORDERLAYOUT

- Définit des zones avec différentes proportions dans lesquelles ajouter des éléments
- La zone centrale s'agrandit proportionnellement au container associé
- Gestionnaire d'affichage par défaut des JFrame



FLOWLAYOUT

- On ajoute au fur et à mesure les composants, si cela ne tient pas dans la ligne, on commence une nouvelle ligne
- Gestionnaire d'affichage par défaut des JPanel
- Possibilité de spécifier l'alignement (centré par défaut)

  setAlignement(int) – 0 gauche, 1 centré, 2 droite

Exemple

```
import javax.swing.*;

public class Exemple {
    public static void main(String[] args) {
        JFrame fenetre = new JFrame();
        JPanel panel = new JPanel();

        JLabel etiquette = new JLabel("Aujourd'hui: ");
        JCheckBox premier = new JCheckBox("Lundi");
        JCheckBox deuxieme = new JCheckBox("Mardi");
        JCheckBox troisieme = new JCheckBox("Mercredi");
        JCheckBox quatrieme = new JCheckBox("Jeudi");
        JCheckBox cinquieme = new JCheckBox("Vendredi", true);
        JCheckBox sixieme = new JCheckBox("Samedi");
```

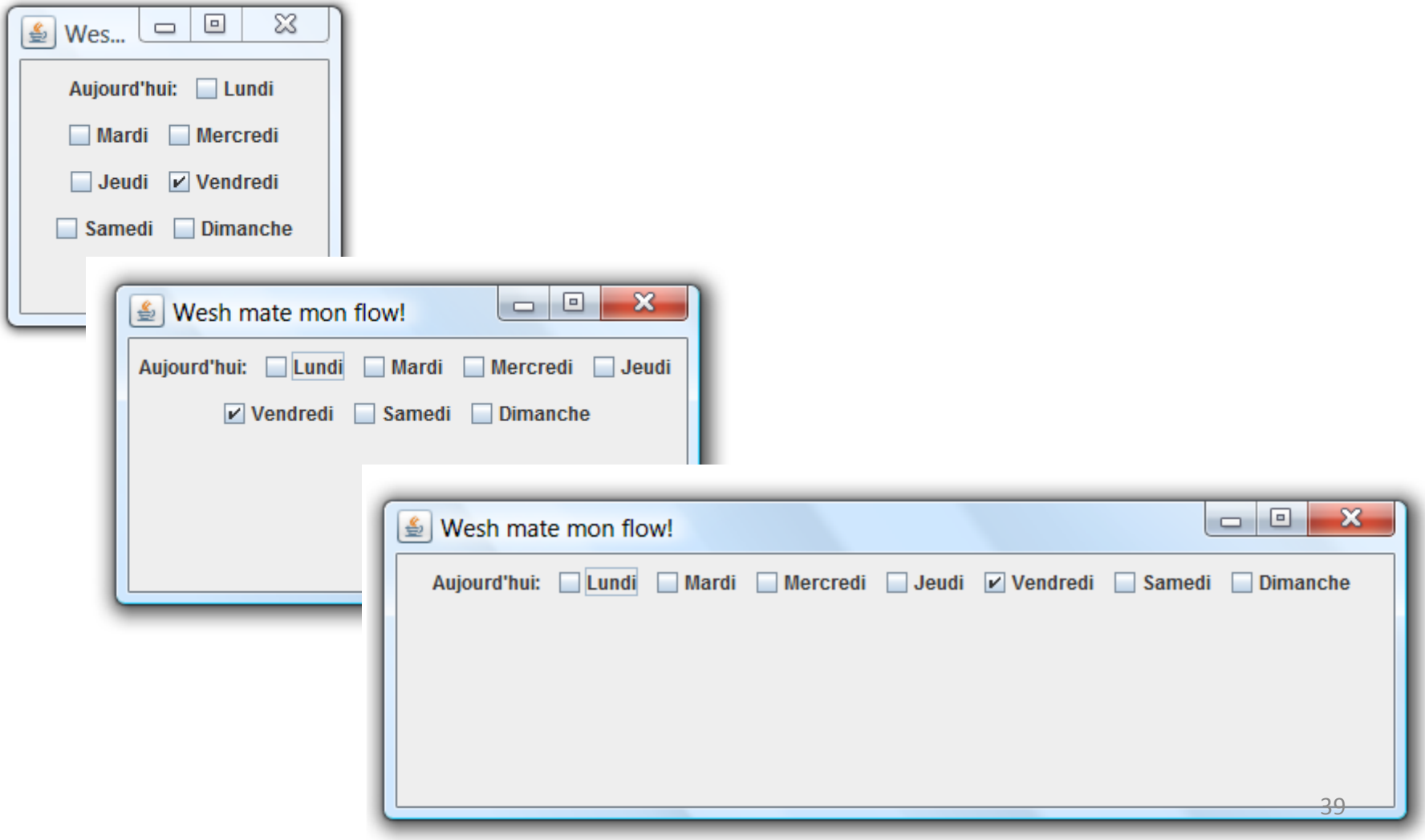
Exemple

```
        JCheckBox septieme = new JCheckBox("Dimanche");
        panel.add(etiquette);
        panel.add(premier); panel.add(deuxieme);
        panel.add(troisieme); panel.add(quatrieme);
        panel.add(cinquieme); panel.add(sixieme);
        panel.add(septieme);

        fenetre.setContentPane(panel);
        fenetre.setTitle("Wesh mate mon flow!");
        fenetre.setBounds(100, 100, 200, 200);
        fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fenetre.setVisible(true);
    }
}
```

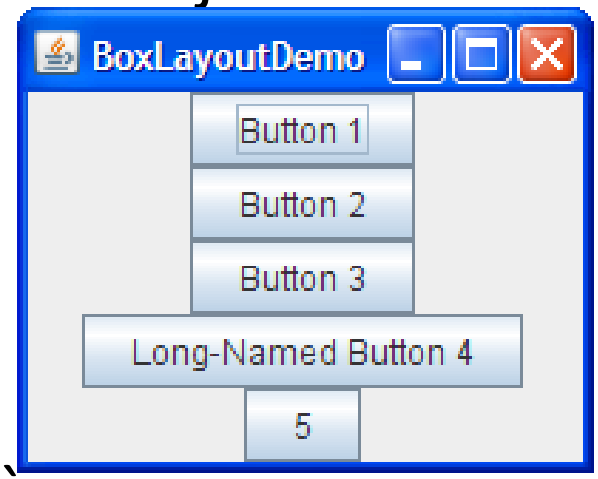
Aperçu du résultat

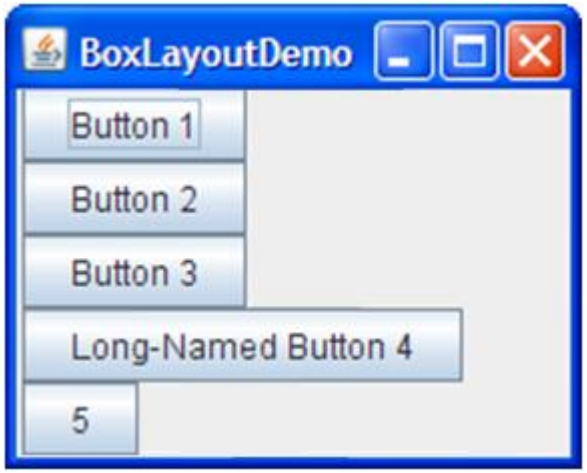
Le placement s'ajuste en fonction de la taille de la fenêtre :



BOXLAYOUT

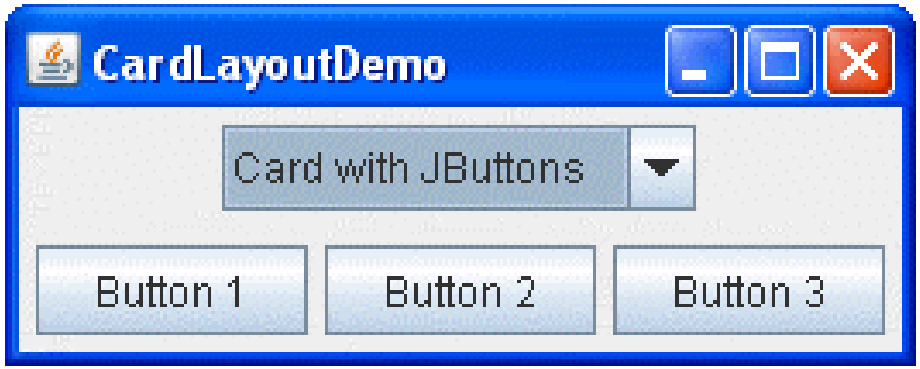
- Définit une boîte dans laquelle seront ajoutés les différents composants
- Cette boîte peut-être :
  - Horizontale : X_AXIS
  - Verticale : Y_AXIS
- On ajoute les éléments au fur et à mesure

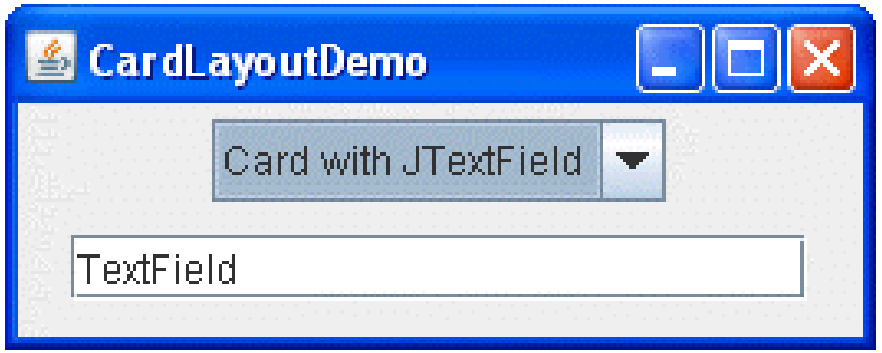




CARDLAYOUT

- Permet d'empiler des composants (suivant des panneaux)
- Le dernier ajouté ou celui spécifié est visible
- Concept de la pile de cartes

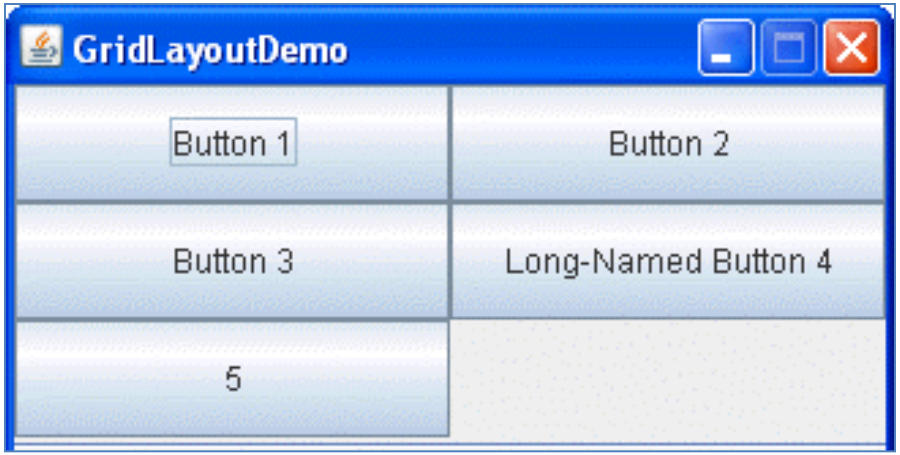




GRIDLAYOUT

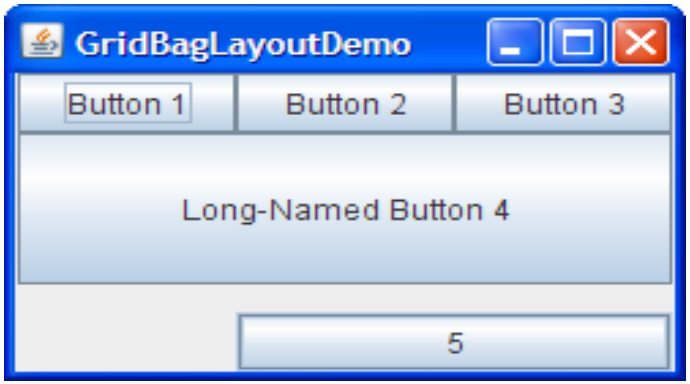
- Définit une grille où ajouter les composants
- L'ajout des composants se fait sans préciser de contraintes
- On déclare le layout de la manière suivante :
  - *new GridLayout(0,2) // row, column*

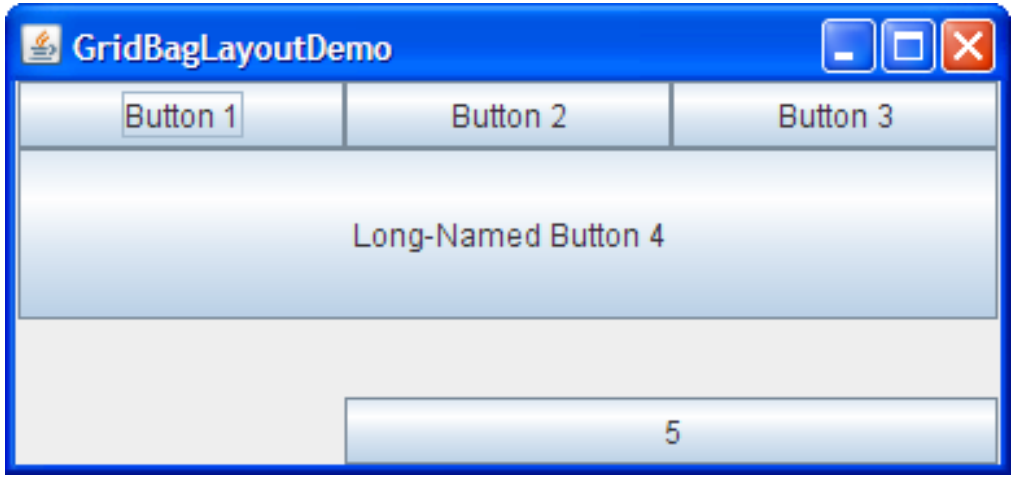
    0 signifie autant qu'on veut



GRIDBAGLAYOUT

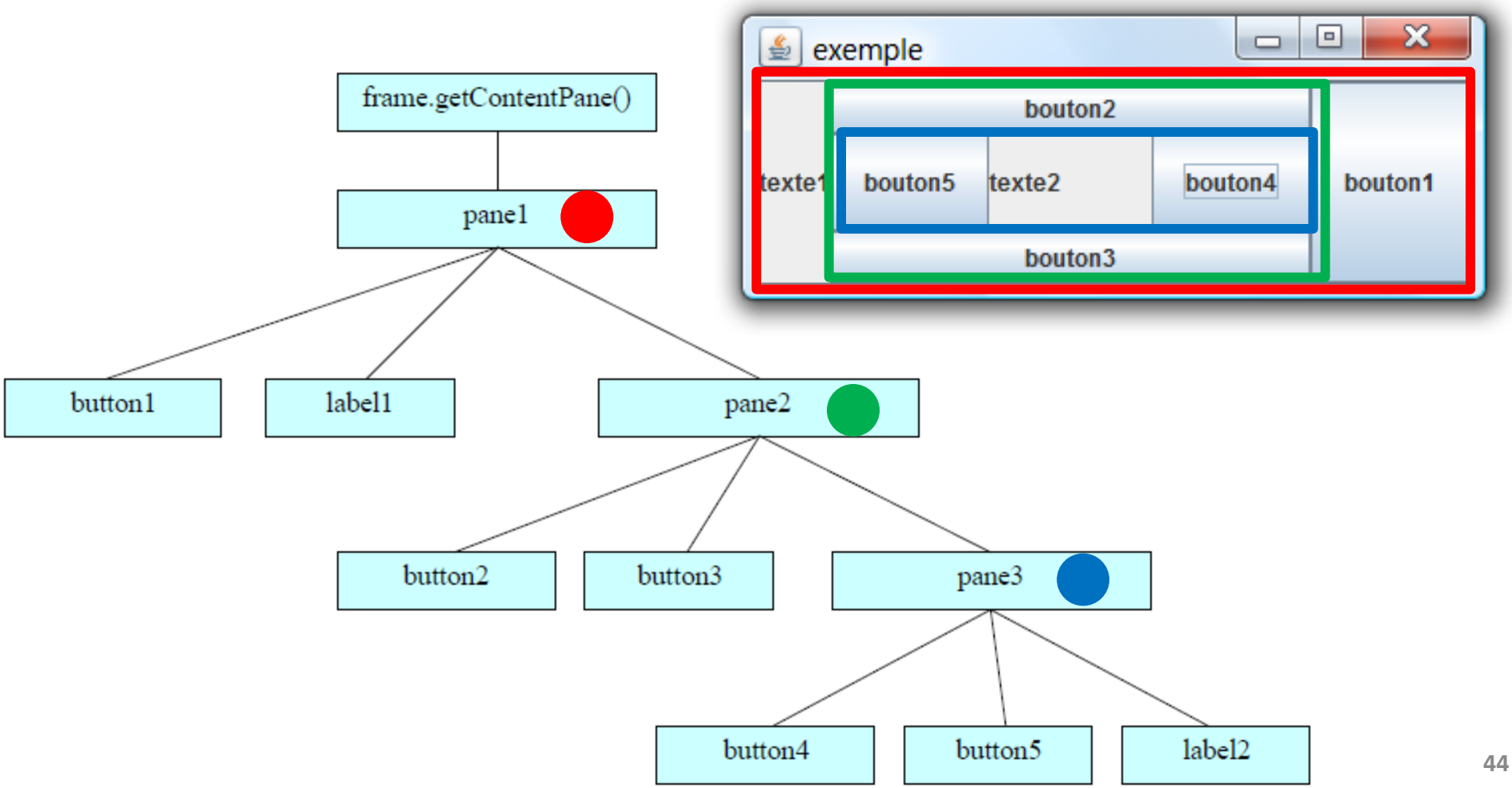
- Place les composants dans une grille modulaire en définissant les contraintes des différents composants à placer
- Pour une mise en page complexe





MISE EN PAGE COMPLEXE
autre solution

- Combinaison de plusieurs panels avec leur propre layout



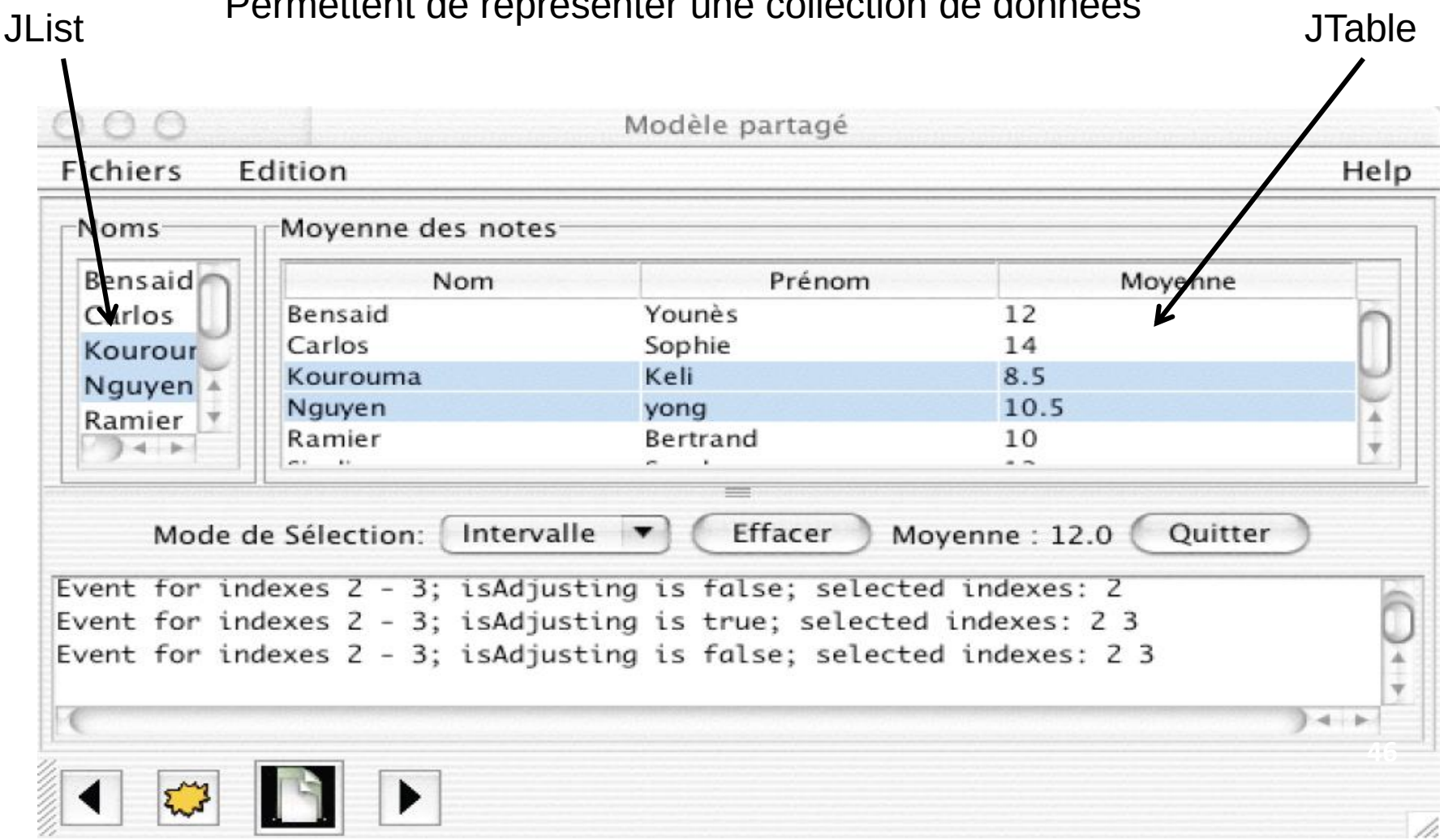
Composants : Contrôles

DES COMPOSANTS « INCLUANT MVC »

Je vous écoute….

JLIST & JTABLE

Permettent de représenter une collection de données



LES *JLIST* ET LES *JTABLE*

- Il est possible de partager des données d'une *JTable* ou d'une *Jlist* avec d'autres composants graphiques considérés comme d'autres vues de la collection de données

- Par exemple : une liste de noms affichée dans
  - une *JList* comme vue (interactive) de la liste de noms permettant la sélection d'éléments
  - une représentation tabulaire de la même liste comme une autre vue
  - un texte indiquant le nombre d'éléments de la liste comme encore une autre vue

- Des méthodes pour gérer les mises à jour des vues utilisant ces données via des évènements (cf prochain cours)

JLIST

- Possède différents constructeurs :
  - Données fixes : JList (Vector listData) ou JList (Object[] listData)
  - Données modifiables : JList (ListModel dm)

- Possède une méthode permettant de spécifier le mode de sélection *setSelectionMode (int mode)*. mode peut valoir :
  - *SINGLE_SELECTION*
  - *SINGLE_INTERVAL_SELECTION*
  - *MULTIPLE_INTERVAL_SELECTION*

- On peut accéder au modèle de données sous-jacent avec la méthode : ListModel getModel()

LISTMODEL

- Pour le modèle de données, utiliser la classe *DefaultListModel*. Ce modèle stocke les objets sous forme de vecteur et fournit les méthodes suivantes :
  - *addElement (Object),*
  - *boolean contains(Object),*
  - *boolean removeElement(Object)*
  - *Object get(index), Object remove(index), int size()*

  - *Plus d'autres méthodes pour gérer les mises à jour des vues utilisants ces données (cf prochain cours)*

JLIST

- Exemple

```
String listData[]= {…,« Carlos »,…, « Ramier»};
DefaultListModel model = new DefaultListModel();
for (int i=0; i<listData.length; i++)
    model.addElement(listData[i]);

JList dataList = new JList(model);

JScrollPane listeScroll = new JScrollPane(dataList);
```

JTABLE

- Un constructeur possible de JTable :
  - JTable (Object[][] rowData, Object[] columnNames)

- Un autre constructeur avec directement le modèle :

```
String nomsCol[]={«Prenom», «Nom»};
String rows[][] = { {«Dinah»,«Cohen»}, … ,
                    {«Said», «Kharrazen»}};
DefaultTableModel model = new DefaultTableModel(rows, nomsCol);
JTable table = new JTable(model);
```

- On peut accéder au modèle sous-jacent avec la méthode :
  - TableModel getModel()

TABLEMODEL

- Il existe 3 différents éléments pour créer un modèle de données :
  - L'interface *TableModel*
  - La classe *AbstractTableModel* qui implémente *TableModel*
  - La classe *DefaultTableModel*
- *DefaultTableModel* est le plus simple à utiliser, quelques constructeurs associés :
  - *DefaultTableModel(int row, int col)*
  - *DefaultTableModel(Object[][] data, Object[] columnNames)*
  - *DefaultTableModel(Vector data, Vector columnNames)*

- *Plus d'autres méthodes pour gérer les mises à jour des vues utilisant ces données (cf prochain cours)*

Je vous écoute….