FACULTE DES SCIENCES AIN CHOCK
UNIVERSITE HASSAN II DE CASABLANCA

Département: Mathématiques & Informatique

Interfaces graphiques et Gestion d'événements

Licence Professionnelle : Développement Informatique

Pr: Youssef Ouassit

Plan Chapitre 1: Introduction à Java

1. **Les interfaces graphiques (GUI):**
   a. AWT vs SWING vs SWT vs JavaFX
   b. La structure de Swing
   c. Les « Components »
   d. Les « Containers »
   e. Les « LayoutManagers »
2. **Gestion d'événements**
   a. Mécanismes et structure
   b. Mise en œuvre

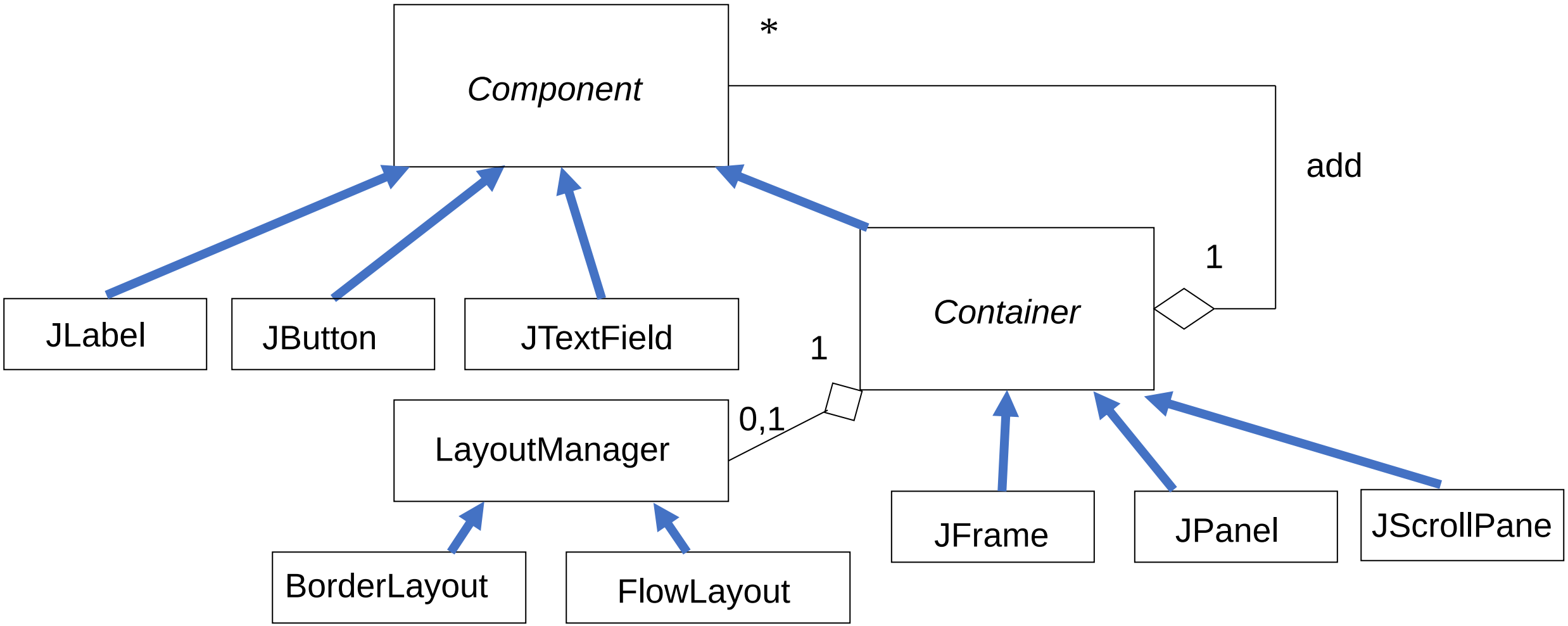
AWT vs SWING vs SWT vs JavaFX

- L'AWT (Abstract Window Toolkit)
  - AWT utilise les composants natifs du système d'exploitation, ce qui rend les interfaces dépendantes de la plateforme.
  - Il est plus limité que Swing ou JavaFX en termes de composants et de personnalisation.
- L'API Swing
  - Offre une large gamme de composants : boutons, boîtes de dialogue, listes déroulantes, tables, arbres, etc.
  - Supporte les interfaces graphiques complexes et personnalisées.
  - Les composants sont lourds (heavyweight), car ils sont dessinés par Java plutôt que par le système d'exploitation.
  - Il est possible de personnaliser les composants Swing avec des LookAndFeel pour changer l'apparence de l'interface.
- JavaFX : la dernière née des API graphiques proposées par le Java SE
  - Supporte les animations, les effets graphiques, et les interfaces utilisateur riches.
  - Utilise le langage FXML pour créer des interfaces graphiques de manière déclarative, semblable à HTML pour les applications web.
  - Supporte les graphiques vectoriels, les vidéos, l'audio, et le rendu 2D/3D.
  - Permet la création d'interfaces graphiques plus fluides et modernes.
- SWT : une autre librairie de mise en œuvre d'interfaces graphiques
  - Utilisé principalement par Eclipse IDE pour ses interfaces graphiques.
  - SWT utilise des composants natifs du système d'exploitation, ce qui donne un rendu plus natif mais peut rendre le déploiement plus complexe.

Structure de Swing (1/2)

- Le Swing offre trois types d'éléments graphiques
  - Les « Containers » (contenants)
  - Les « Components » (composants ou contenus)
  - Les « LayoutManagers » (disposition des objets d'un contenant)
- Les « Containers »
  - Sont destinés à accueillir des composants
  - Gèrent l'affichage des composants
  - *Ex*: JFrame, JPanel,…
- Les « Components »
  - Constituent différents éléments de l'affichage (boutons, barres de menus, etc.)
  - *Ex*: Button, Canvas, Label, Scrollbar, Checkbox
- Les « LayoutManagers »
  - Gèrent la disposition des composants au sein d'un conteneur

Structure de Swing (2/2)



Les « Components »

Les composantes de Java Swing peuvent être classés en trois classes :

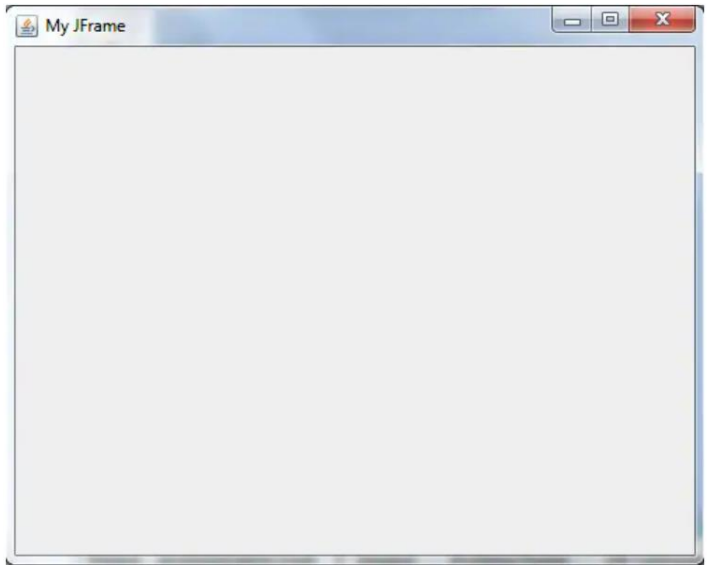
- **Composants conteneurs** : Les conteneurs sont des composants qui contiennent d'autres composants Swing.
- **Composants de contenu** : Ces composants sont utilisés pour afficher du contenu ou interagir avec l'utilisateur.
  - **Composants de fenêtre :** Ce sont des composantes pour les conteneurs de type JFrame.

Les « Components »

Composants conteneurs :

- **JFrame :** C'est une fenêtre principale avec des décorations standard comme une barre de titre, des bordures et des boutons de contrôle (minimiser, agrandir, fermer).

```
import javax.swing.JFrame;
public class MyFrame extends JFrame {

   public MyFrame() {
        this.setTitle("MyFrame");



        this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        this.setSize(400, 400);
        this.setLocationRelativeTo(null);
        this.setVisible(true);
   }
}
```

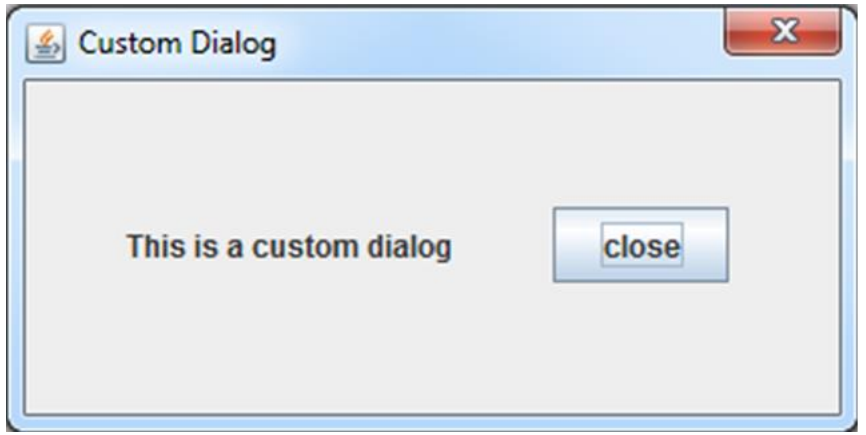


Les « Components »

Composants conteneurs :

- **JDialog** : Utilisé pour créer des boîtes de dialogue. Un JDialog est une fenêtre modale ou non modale qui peut être utilisée pour obtenir des informations de l'utilisateur ou afficher des messages.

```
JDialog dialog = new JDialog(parent, "Mon JDialog", true);
dialog.setSize(250, 150);
dialog.setLocationRelativeTo(parent);
```

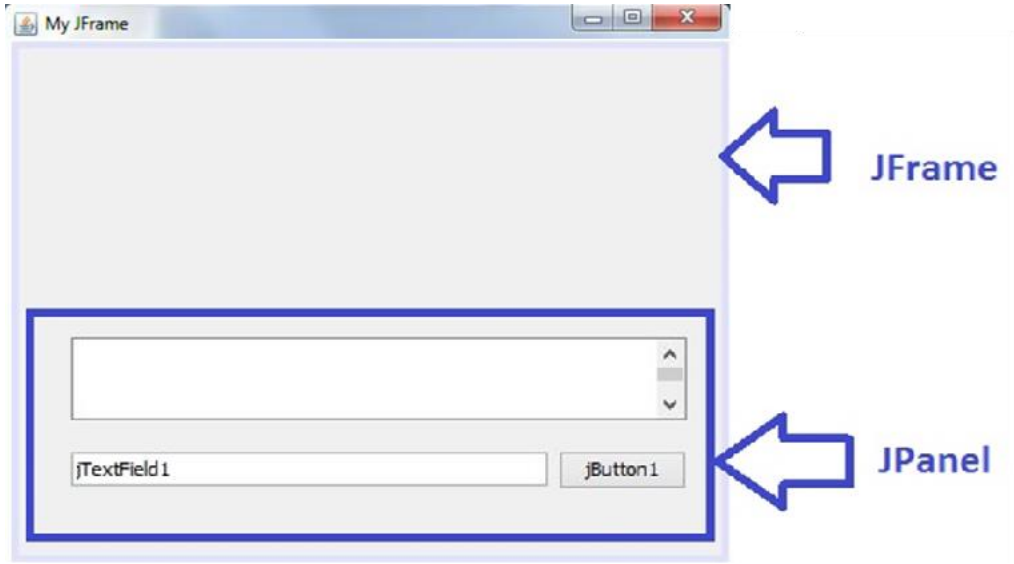


Les « Components »

Composants conteneurs :

- **JPanel** : Un conteneur général qui permet de regrouper des composants sous un même panneau. On peut l'utiliser pour diviser une interface en sections.

```
JPanel p = new JPanel();
p.setLyout(new BorderLayout());
```

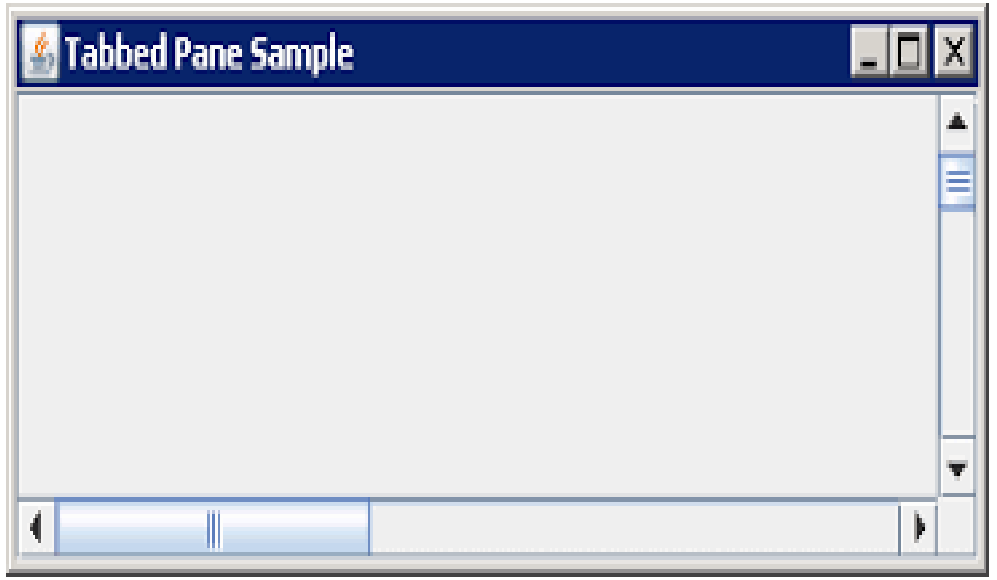


Les « Components »

Composants conteneurs :

- **JScrollPane** : Permet de rendre un autre composant, comme un JTextArea ou JList, défilable en cas de contenu trop grand.

```
JScrollPane p = new JScollPane();
p.setLyout(new BorderLayout());
```



Les « Components »

Composants de contenu:

- **JLabel :** Affiche du texte ou une image.

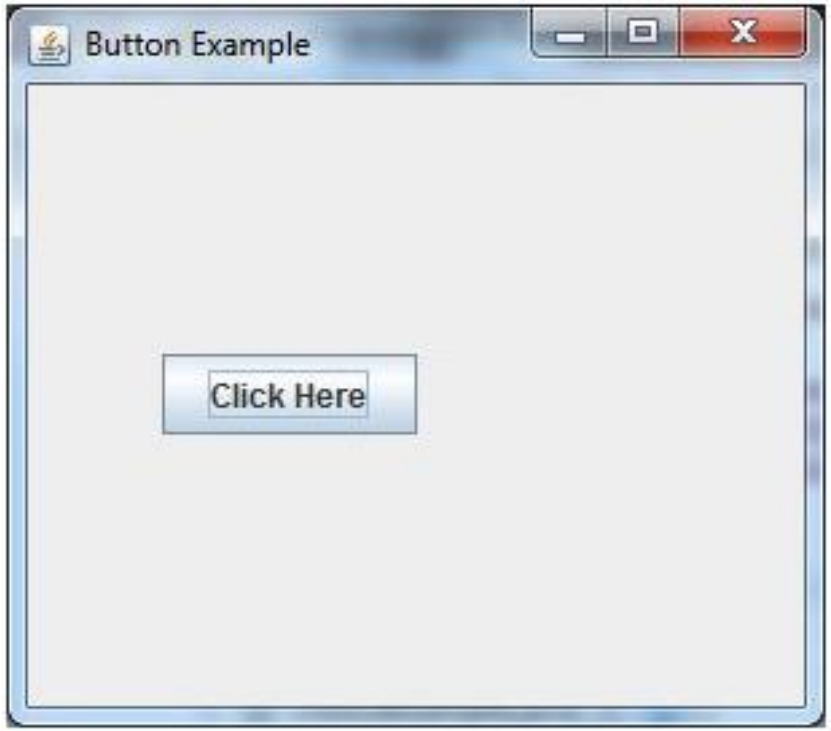
```
JLabel p = new JLabel("First Label");
```

Les « Components »

Composants de contenu:

- **JButton :** Un bouton sur lequel l'utilisateur peut cliquer pour effectuer une action.

```
JButton p = new JButton("Click Here");
```

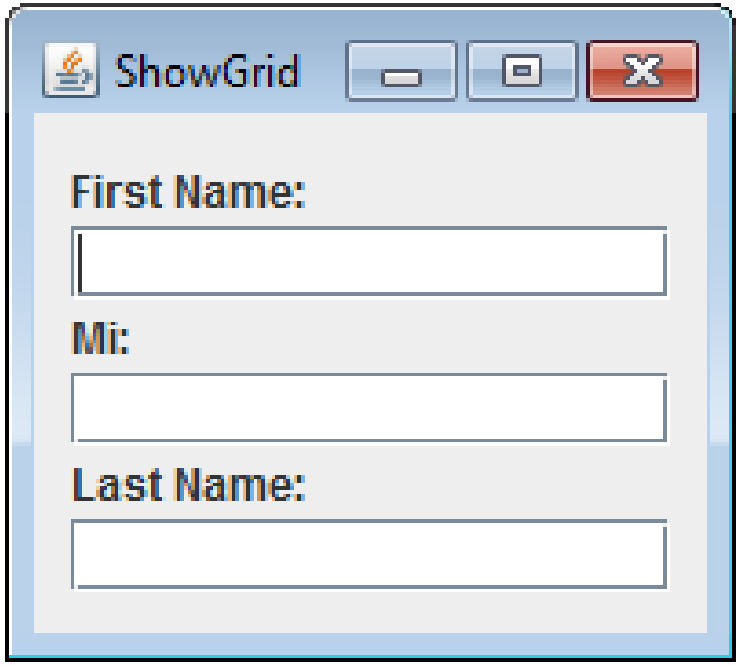


Les « Components »

Composants de contenu:

- **JTextField :** Un champ de texte permettant à l'utilisateur de saisir une ligne de texte.

```
JTextField nameTextField = new JTextField();
```

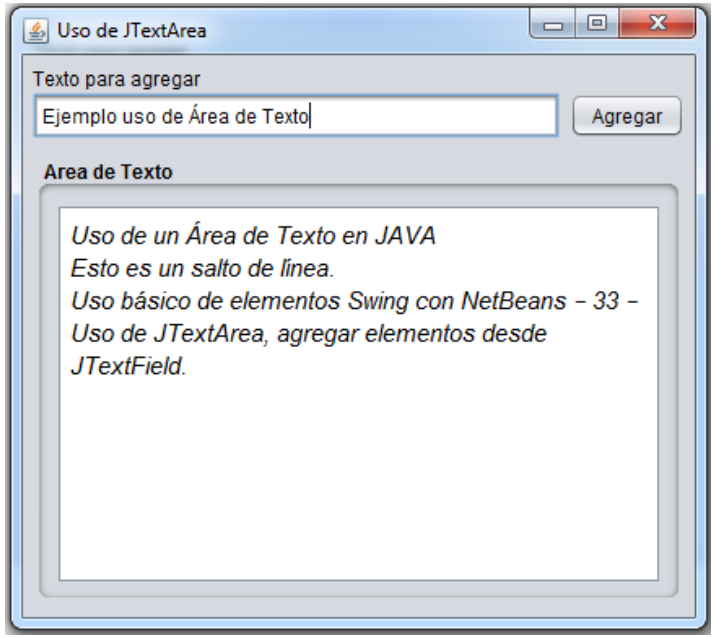


Les « Components »

Composants de contenu:

- **JTextArea :** Une zone de texte qui permet de saisir un long texte sur plusieurs lignes.

```
JTextArea nameTextField = new JTextArea();

// Spécifier le nombre de lignes et de colonnes
JTextArea nameTextField = new JTextArea(10, 30);
```

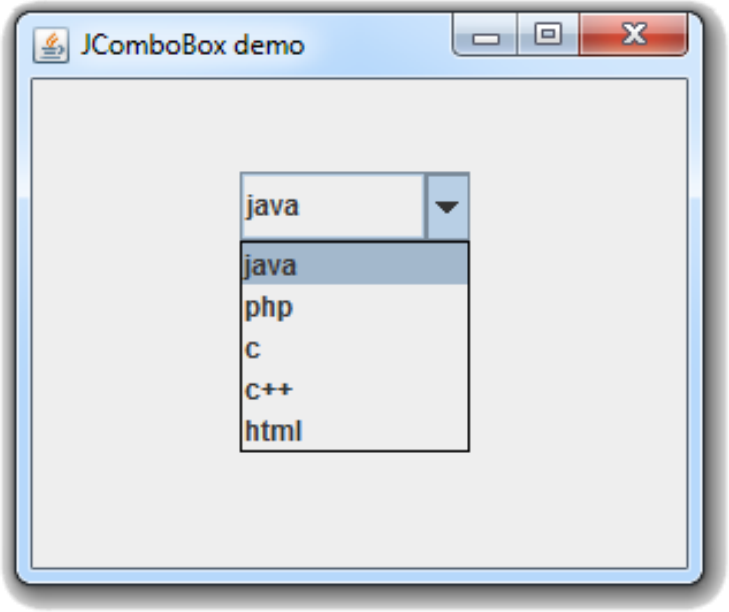


Les « Components »

Composants de contenu:

- **JComboBox :** Un composant de liste déroulante permettant à l'utilisateur de sélectionner une option parmi plusieurs.

```
String[] options = {"Java", "php", "c", "c++"};
JComboBox p = new JComboBox(options);

// Ajouter d'autres items
p.addItem("html");
```

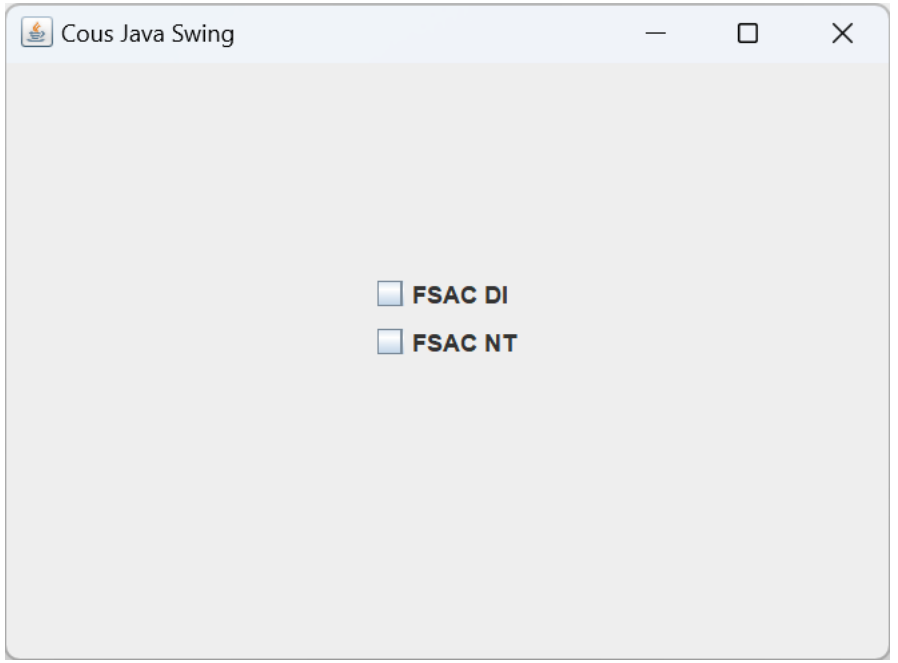


Les « Components »

Composants de contenu:

- **JCheckBox** : un élément qui peut être sélectionné ou désélectionné et qui affiche son état à l'utilisateur

```
JCheckBox p = new JCheckBox("FSAC DI");

// Pour vérifier si sélectionnée
p.isSelected();
```

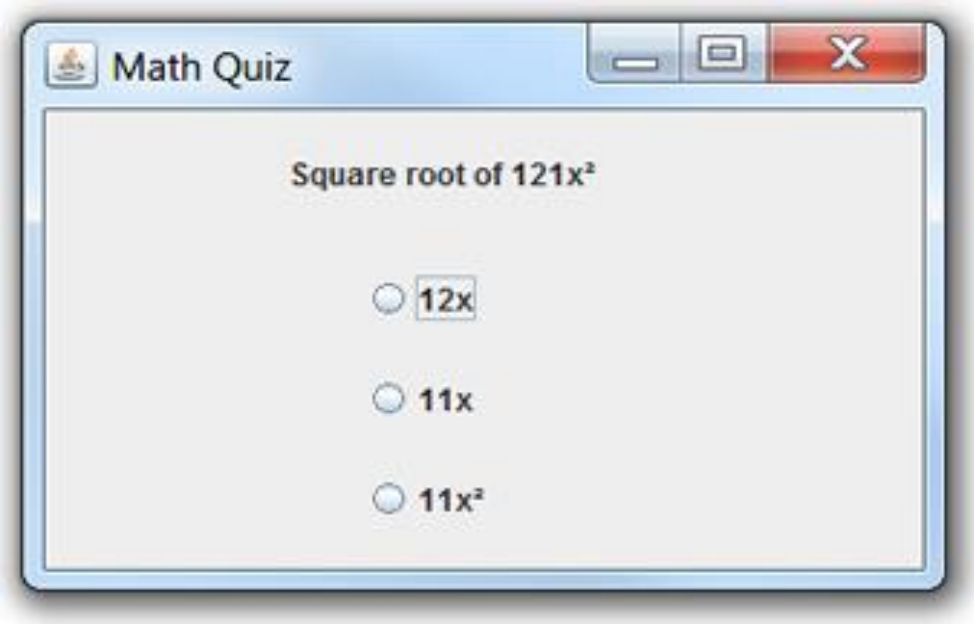


Les « Components »

Composants de contenu:

- **JRadioButton** : Un bouton radio qui fait partie d'un groupe où une seule option peut être sélectionnée à la fois.

```
JRadioButton p = new JRadioButton("12x");

// Pour vérifier si sélectionnée
p.isSelected();

ButtonGroup group = new ButtonGroup();
group.add(option1);
group.add(option2);
group.add(option3);
```

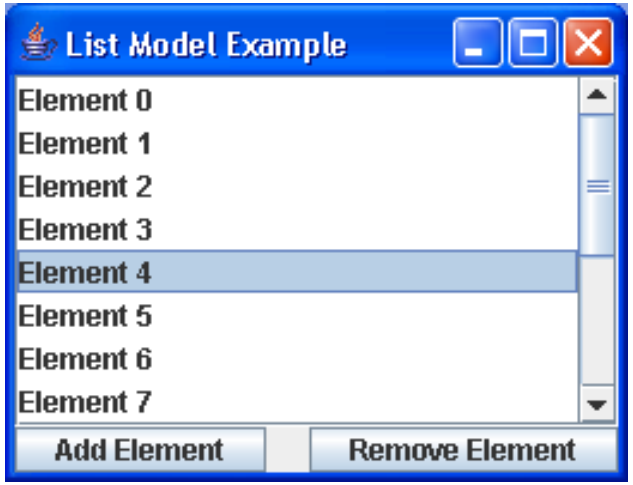


Les « Components »

Composants de contenu:

- **JList** : Affiche une liste d'éléments parmi lesquels l'utilisateur peut sélectionner

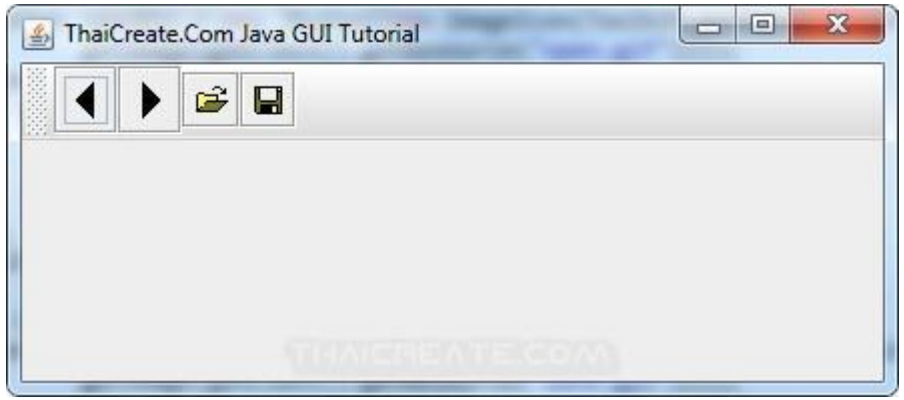
```
DefaultListModel<String> listModel = new DefaultListModel<>();
for(int i=0; i<8; i++)
     listModel.addElement("Element "+i);
JList<String> list = new JList<>(listModel);

JScrollPane p = new JScrollPane(list);
contentPane.add(p);

// Ajouter un élément
listModel.addElement("Element 8");
```

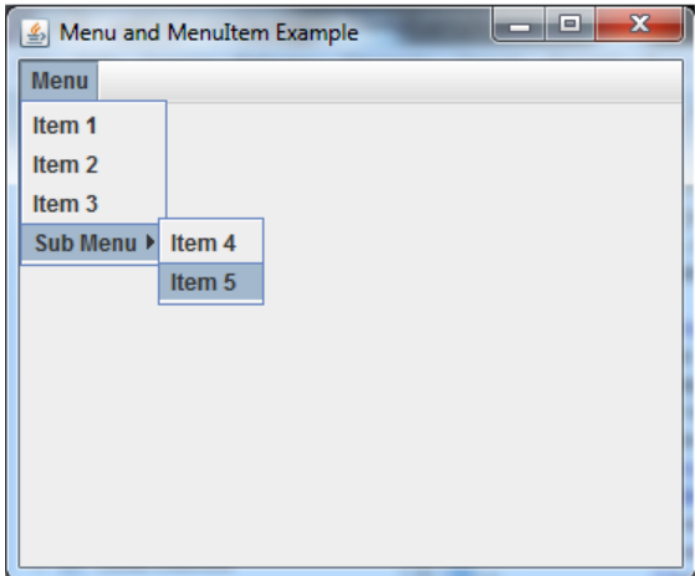


Les « Components »

Composants de fenêtre:

- **JToolBar** : Une barre d'outils qui contient des boutons pour des actions courantes dans l'application.
- **JMenuBar, JMenu, JMenuItem** : Composants pour créer des menus.





Les « LayoutManagers » (1/11)

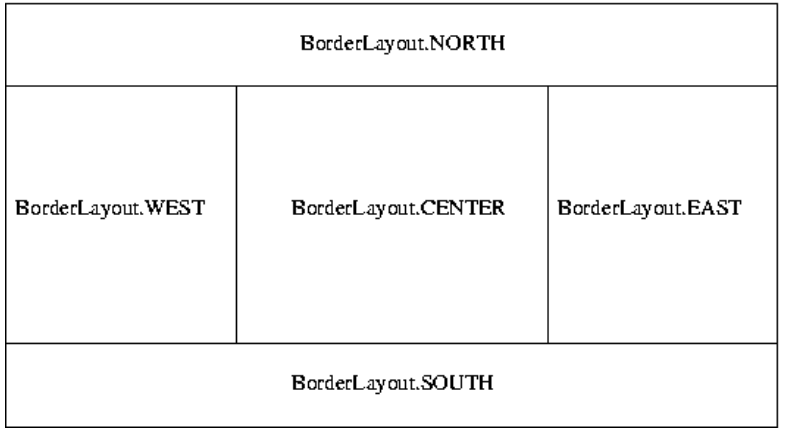
- Rôle :
  - Gérer la disposition des composants au sein d'un conteneur

- Types principaux:
  - BorderLayout: divise le conteneur en 5 zones
  - FlowLayout: rajoute les composants au fur et à mesure
  - GridLayout: applique une grille au conteneur pour aligner les composants
  - CardLayout: pour un conteneur qui contient plusieurs cartes
  - GridBagLayout: grille de cellules élémentaires

Les « LayoutManagers » (2/11)

BorderLayout

- Principe
  - Positionne les composants suivants les points cardinaux
  - Layout par défaut des « Frame »
- Répartition
  - La largeur l'emporte pour le Nord et le Sud
  - La hauteur l'emporte pour l'Est et l'Ouest
  - Le centre occupe tout ce qui reste
- Utilisation
  - add(unBouton, BorderLayout.NORTH)
  - new BorderLayout(int, int) : Le constructeurs accepte les intervalles horizontal et vertical entre les éléments



Les « LayoutManagers » (3/11)

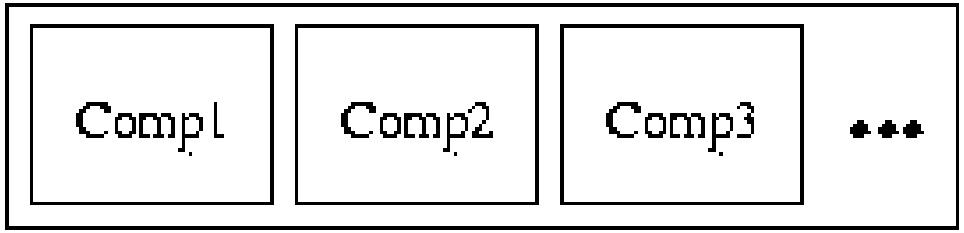
BorderLayout :

```
JButton btn1 = new JButton("Nord");
JButton btn2 = new JButton("Sud");
JButton btn3 = new JButton("Est");
JButton btn4 = new JButton("Ouest");
JButton btn5 = new JButton("Centre");
contentPane.add(btn1, BorderLayout.NORTH);
contentPane.add(btn2, BorderLayout.SOUTH);
contentPane.add(btn3, BorderLayout.EAST);
contentPane.add(btn4, BorderLayout.WEST);
contentPane.add(btn5, BorderLayout.CENTER);
```

Les « LayoutManagers » (4/11)

FlowLayout :

- Principe
  - Rajoute les composants au fur et à mesure
  - La taille des composants l'emporte

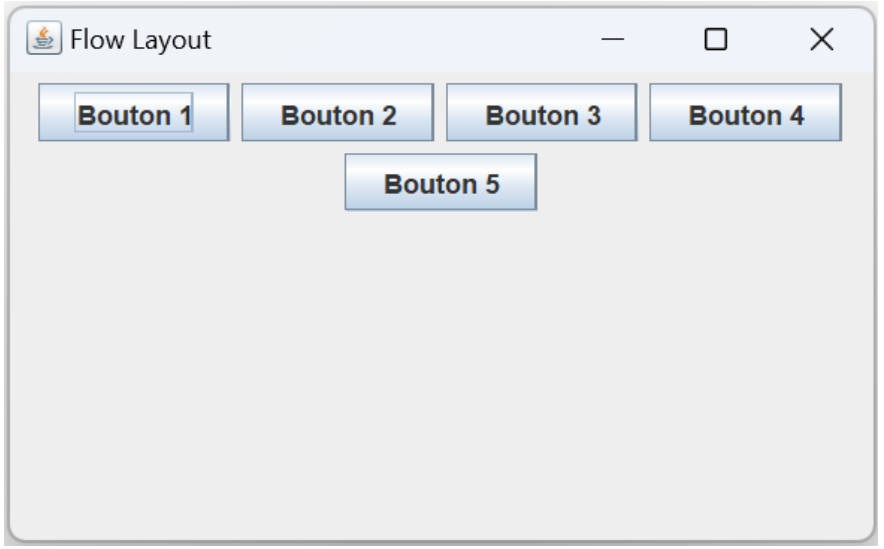


- Utilisation
  - new FlowLayout (int alignment)
    - alignment: LEFT, CENTER, RIGHT
  - new FlowLayout (int alignment, int hintervalle, int vintervalle)
    - alignment: LEFT, CENTER, RIGHT
    - hintervalle: L'intervalle horizontal entre les composants
    - vintervalle: L'intervalle vertical entre les composants

Les « LayoutManagers » (5/11)

FlowLayout :

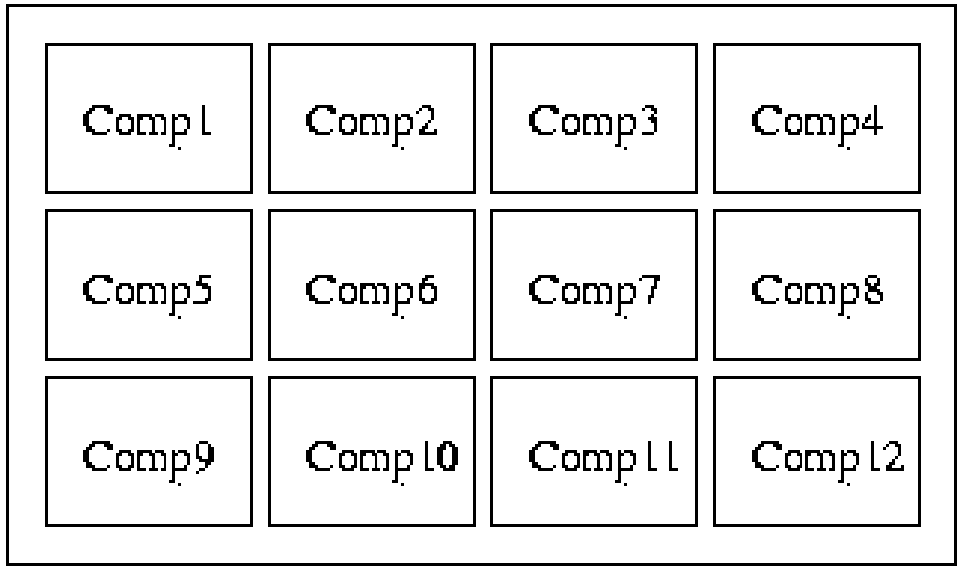
```
this.setLayout(new FlowLayout());
this.add(new JButton("Bouton 1"));
this.add(new JButton("Bouton 2"));
this.add(new JButton("Bouton 3"));
this.add(new JButton("Bouton 4"));
this.add(new JButton("Bouton 5"));
```



Les « LayoutManagers » (6/11)

GridLayout :

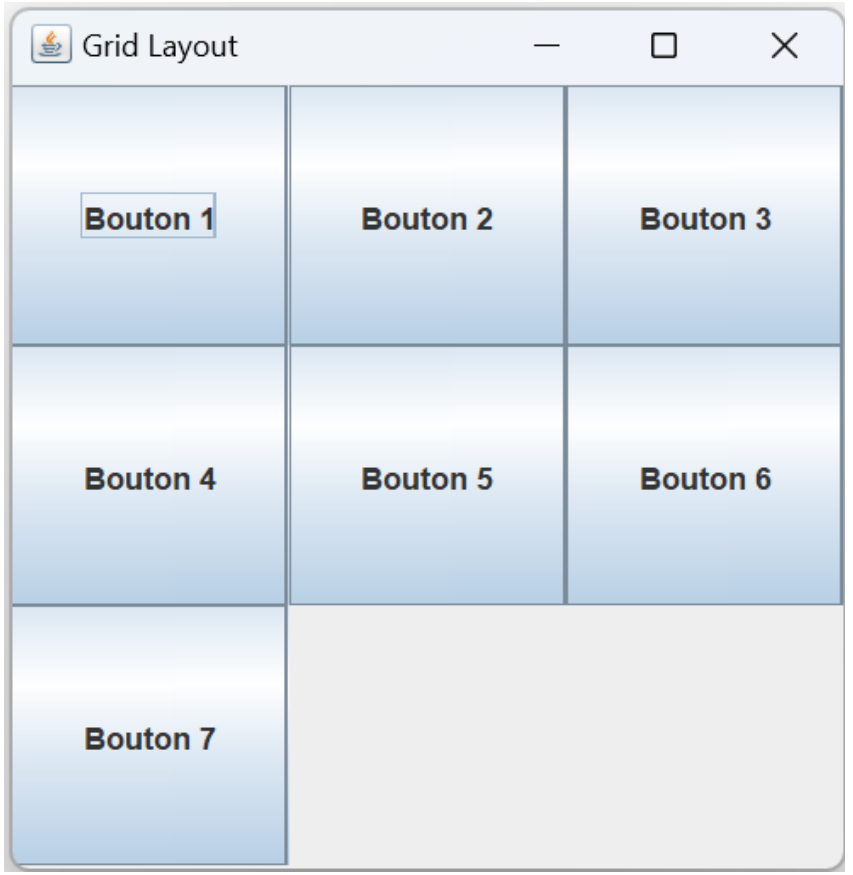
- Principe
  - Définit une grille a 2 dimensions
- Utilisation
  - new GridLayout (int, int)
    - Définit le nombre de lignes et de colonnes
  - new GridLayout (int, int, int, int)
    - Définit le nombre de lignes et de colonnes
    - Définit l'alignement horizontal et vertical



Les « LayoutManagers » (7/11)

GridLayout :

```
this.setLayout(new GridLayout(3, 3));
this.add(new JButton("Bouton 1"));
this.add(new JButton("Bouton 2"));
this.add(new JButton("Bouton 3"));
this.add(new JButton("Bouton 4"));
this.add(new JButton("Bouton 5"));
this.add(new JButton("Bouton 6"));
this.add(new JButton("Bouton 7"));
```



Les « LayoutManagers » (8/11)

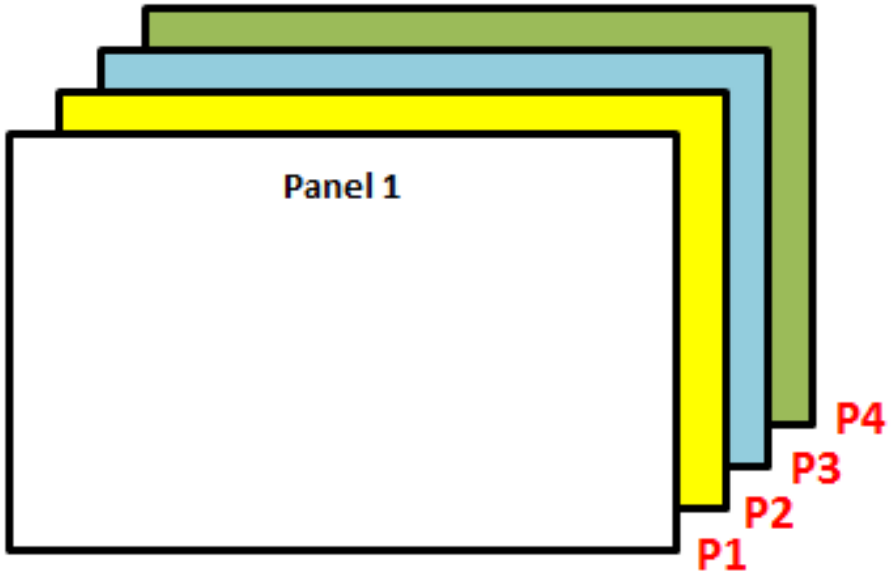
CardLayout :

- Principe
  - Pour un container qui contient plusieurs cartes
  - Seule une carte est visible à chaque fois
  - L'utilisateur peut passer d'une carte à l'autre

Les « LayoutManagers » (9/11)

CardLayout :

```
CardLayout cardLayout = new CardLayout()
contentPane.setLayout(cardLayout);
JPanel card1 = new JPanel();
JPanel card2 = new JPanel();
JPanel card3 = new JPanel();
contentPane.add(card1);
contentPane.add(card2);
contentPane.add(card3);

cardLayout.next(contentPane);
```



Les « LayoutManagers » (10/11)

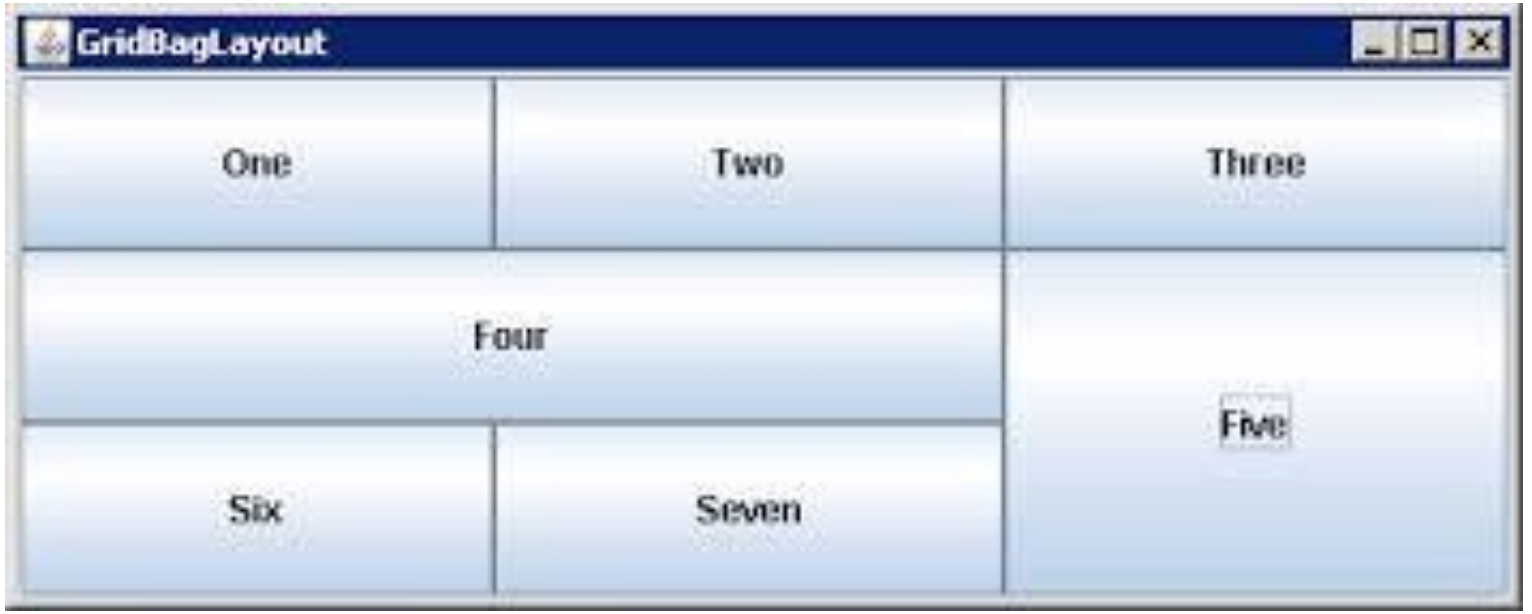
GridBagLayout :

- Principe
  - Le plus compliqué et le plus flexible
  - Une grille de cellules élémentaires
  - Les composants graphiques peuvent s'étaler indifféremment sur ces cellules
  - Pour positionner un composant, il faut utiliser un objet GridBagConstraints
  - On peut spécifier le comportement des composants quand on étire la fenêtre

- Utilisation
  - new GridBagLayout()
  - new GridBagConstraints();

Les « LayoutManagers » (11/11)

GridBagLayout :



Gestion d'événements : Mécanismes et structure (1/4)

- Une source d'événements
  - Génère des objets événements
  - Les fait écouter par un ensemble d'écouteurs d'événements
  - En général: un composant ou conteneur graphique
- Les objets événements
  - xxxEvent
  - Contiennent la description et les caractéristiques d'un événement
- Les objets écouteurs
  - xxxListener ou xxxAdapter
  - Concrétisent des méthodes définies dans les Interfaces
  - Indiquent leur réaction en réponse aux événements
  - Sont des interfaces implémentables dans les classes
  - Peuvent être implémentés par les sources d'événements elles-mêmes (Une source d'événements peut « s'écouter » elle-même)

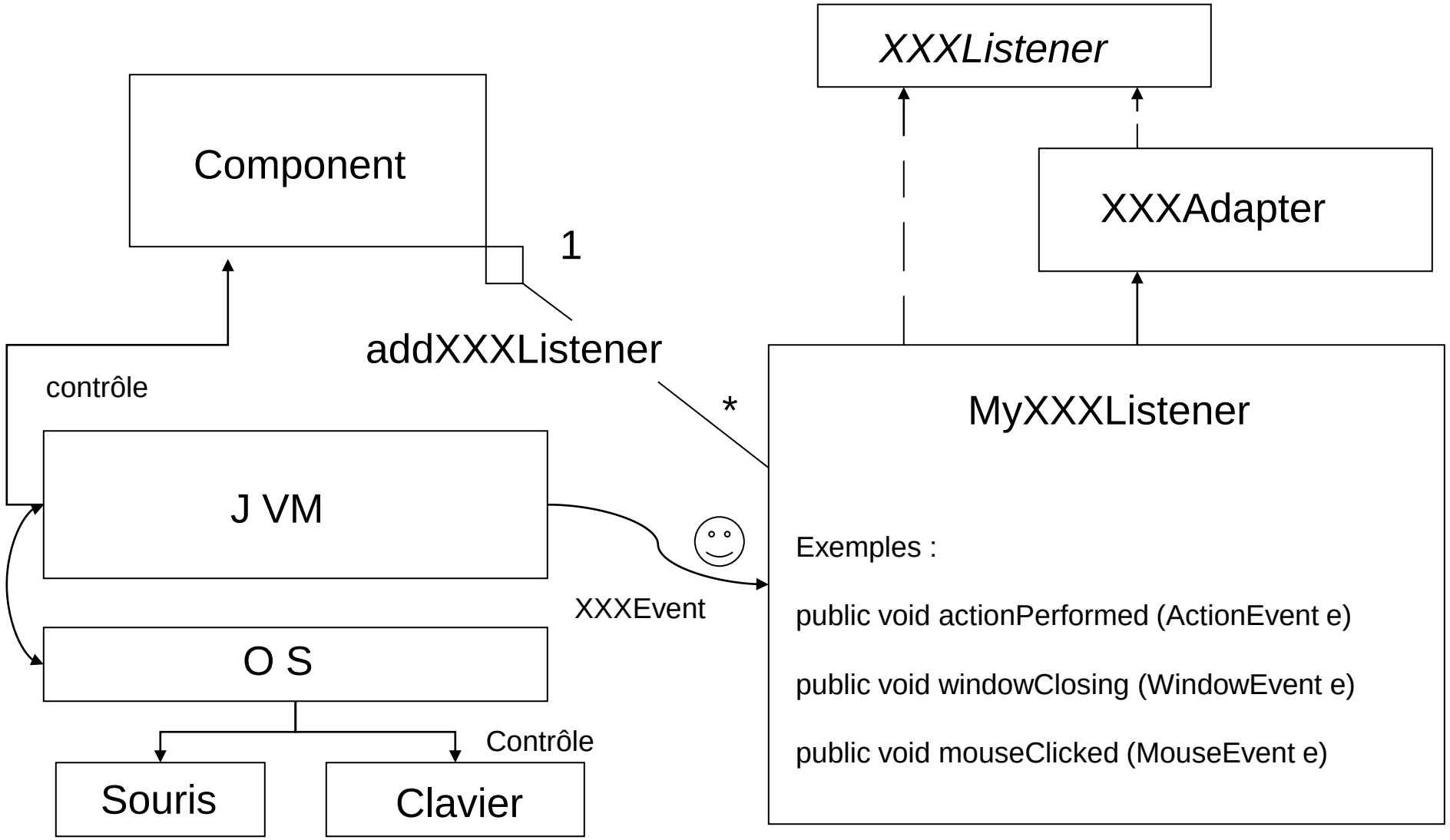
Gestion d'événements : Mécanismes et structure (2/4)

1. Un événement se produit
2. La source d'événement dans laquelle il se produit génère un objet de type événement
3. La source transmet l'événement à son (ses) écouteur(s)
4. L'écouteur appelle la méthode correspondant au type d'événement et lui passe en argument l'objet événement
5. La méthode en question spécifie les traitements à réaliser lorsqu'un événement du type correspondant se produit
6. Dans ses traitements, la méthode peut examiner les caractéristiques de l'événement (position du curseur de la souris, code de la touche pressée au clavier…) et adapter son comportement en fonction de ces caractéristiques

Gestion d'événements : Mécanismes et structure (3/4)

- Evénements
  - ActionEvent, KeyEvent , MouseEvent , WindowEvent , FocusEvent , AdjustmentEvent, ComponentEvent, ContainerEvent,, ItemEvent, TextEvent, …
- Les Interfaces Ecouteurs
  - ActionListener, KeyListener , MouseListener , WindowListener , FocusListener , AdjustmentListener, ComponentListener, ContainerListener,, ItemListener, MouseMotionListener,
- Les Adapteurs correspondants
  - ActionAdapter, WindowAdapter, KeyAdapter, MouseAdapter, ...
- Les Sources d'événements
  - Button, List, MenuItem, TextField, ScrollBar, CheckBox, Component, Container, Window

Gestion d'événements : Mécanismes et structure (1/4)



Gestion d'événements : Mise en oeuvre

- Par implémentation de l'interface
  - Usage
    - ➔ `public class MaClasse implements MouseListener`
  - Avantages et inconvénients
    - ☺ Meilleur sur le plan orienté objet
    - ☺ La classe peut hériter d'une autre classe
    - ☺ Consistance
- Par héritage de l'adapteur
  - Usage
    - A chaque interface correspond un adapteur qui l'implémente lui-même
    - ➔ `public class MaClasse extends MouseAdapter`
  - Avantages et inconvénients
    - ☺ Code simple (l'adapteur redéfinit déjà les méthodes, etc.)
    - ☹ La classe ne peut plus hériter d'une autre classe

Gestion d'événements : Mise en œuvre – Click de souris

```
class MonFrame extends Frame implements MouseListener
{
  public MonFrame()
  {
    addMouseListener(this);
  }
  public void mousePressed(MouseEvent e) {}
  public void mouseClicked(MouseEvent e)
  {
    if(e.getX()>50 && e.getY()<100){...}
  }
  public void mouseReleased(MouseEvent e) {}
  public void mouseEntered(MouseEvent e) {}
  public void mouseExited{MouseEvent e) {}
}
```

Gestion d'événements : Mise en œuvre – Déplacement de la souris

```
class MonFrame extends Frame implements MouseMotionListener
{
  public MonFrame()
  {
    addMouseMotionListener(this);
  }
  public void mouseDragged(MouseEvent e) {}
  public void mouseMoved(MouseEvent e)
  {
    if(e.getX()>50 && e.getY()<100){...}
  }
}
```

Gestion d'événements : Mise en œuvre – Clavier

```
class MonFrame extends Frame implements KeyListener
{
  public MonFrame()
  {
    addKeyListener(this);
  }
  public void keyPressed(KeyEvent e) {}
  public void keyTyped(KeyEvent e)
  {
    if(e.getKeyCode()==KeyEvent.VK_Q){System.exit(0)}
  }
  public void keyReleased(KeyEvent e) {}
}
```

Gestion d'événements : Mise en œuvre – Evènement de la fenêtre

```
class MonFrame extends Frame implements WindowListener
{
  public MonFrame()
  {
    addWindowListener(this);
  }

  public void windowClosed(WindowEvent e) {}
  public void windowIconified(WindowEvent e) {}
  public void windowOpened(WindowEvent e) {}
  public void windowClosing(WindowEvent e) {System.exit(0);}
  public void windowDeiconified(WindowEvent e) {}
  public void windowActivated(WindowEvent e) {}
  public void windowDeactivated(WindowEvent e) {}
}
```

Gestion d'événements : Mise en œuvre – Actions

```
public class MonPanel extends Panel implements ActionListener
{                          // MonPanel devient un écouteur d'événements
  public void actionPerformed(Action evt)
  {     // Ce qui est fait en réponse à l'événement….
        // je peux extraire de l'information sur l'événement
        // transmis par evt
        // je suis obligé de re-définir cette méthode abstraite
        // dans ActionListener
  }
}


MonPanel panel = new MonPanel();
JButton button = new Jbutton("OK");// Crée un bouton qui est source
                                   // d'évènements de type ActionEvent
button.addActionListener(panel)    // Associer un écouteur
                                   // d'événements panel au bouton
```