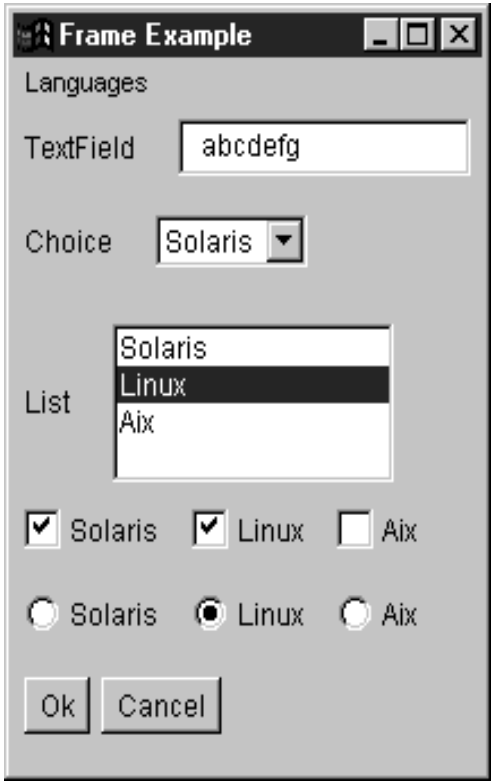
Composants Graphiques

Interfaces java

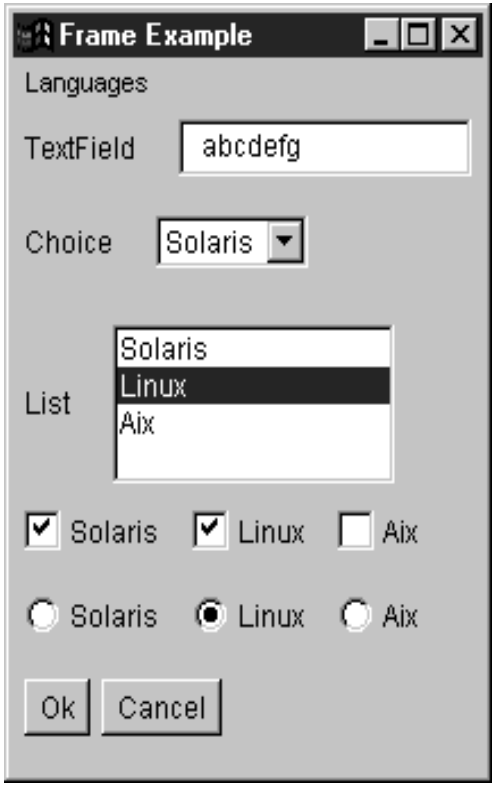
- Interfaces java: menus, butons, cases à cocher, zone de saisie texte, ...etc

- Utilisation de deux API java:
  - L'API AWT (Abstract Window Toolkit)
    - Première API, dès JDK 1.0
    - Avantage/inconvénient : appel à l'OS sous-jacent pour l'affichage (rapide)
  - L'API Swing
    - Portable à 100% , dessin par Java
    - Coût plus élevé (lent)
    - Tous les composants du swing dérivent d'une classe de l'AWT
    - Avec les applets, swing n'est pas utilisable avec tous les navigateurs
- Les deux API, nécessitent la gestion des événements et la stratégie de placement des objets dans leur conteneur (Layout)



Aperçu général

- Les programmes à interfaces graphiques font usage des classes *awt* (*abstract windowing toolkit*) et/ou *swing*.
- Ils sont dirigés par évènements.
- Classe de base des *awt* : la classe abstraite **`Component`**.
- Classe de base des composants *swing* : **`JComponent`**.
- On distingue, par service
  - les classes conteneur
  - les classes d'interaction
  - les menus et dialogues
- *Swing* offre une palette bien plus large.



“Bonjour, monde”

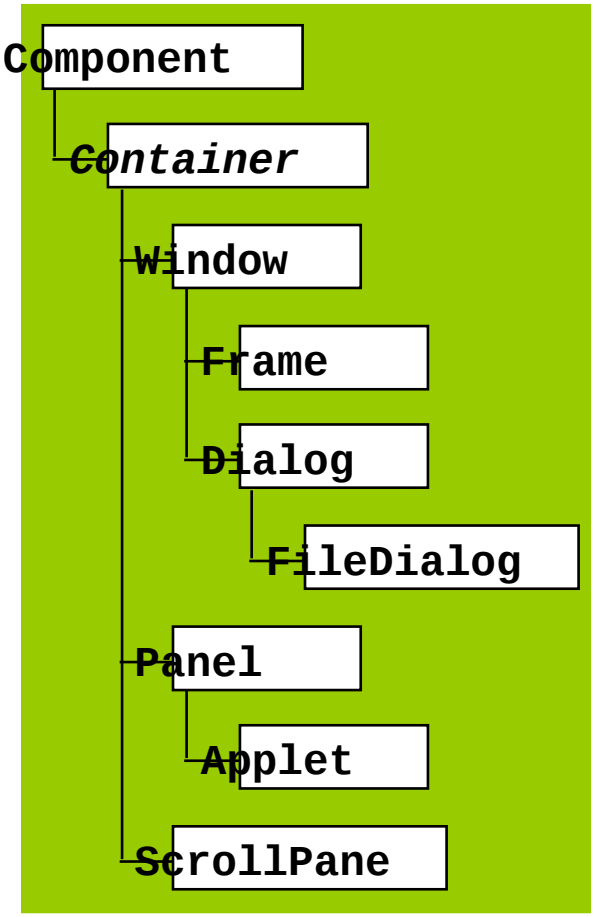

```
import javax.swing.*;

class bjm {
  public static void main(String[] args) {
    JOptionPane.showMessageDialog(null, "Bonjour, monde");
    System.exit(0);
  }
}
```

- Le programme est ***dirigé par événements***
  - un thread dédié: **`EventDispatchThread`** gère la distribution des événements
  - le programme ne termine pas implicitement, d’où le **`System.exit(0)`**

Les conteneurs

- **`Container`** classe abstraite, responsable du layout
- **`Window`** pour interaction avec le système
- **`Frame`** fenêtre principale d'application
- **`Panel`** contient des composants
- **`Applet`**
- **`ScrollPane`** enrobe un conteneur d'ascenseurs

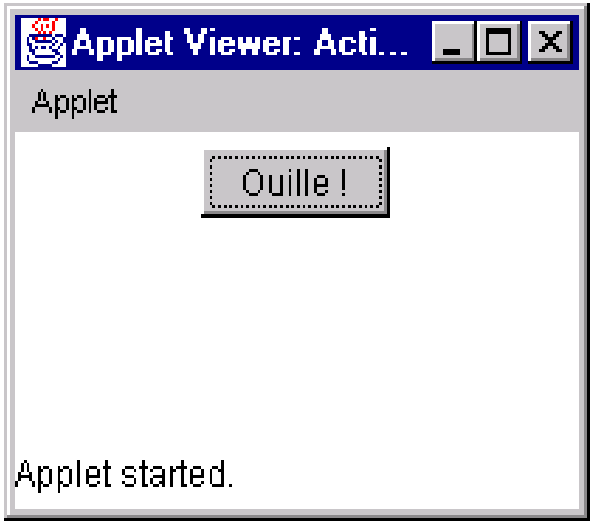


- **un *programme* étend `Frame`**
- **une *applette* étend `Applet`**

Exemple

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

public class ActionExemple extends Applet
   implements ActionListener
{
   Button b;
   public void init() {
      b = new Button("En avant !");
      b.addActionListener(this);
      add(b);
   }
   public void actionPerformed(ActionEvent e) {
      if (b.getLabel().equals("En avant !"))
         b.setLabel("Ouille !");
      else
         b.setLabel("En avant !");
   }
}
```

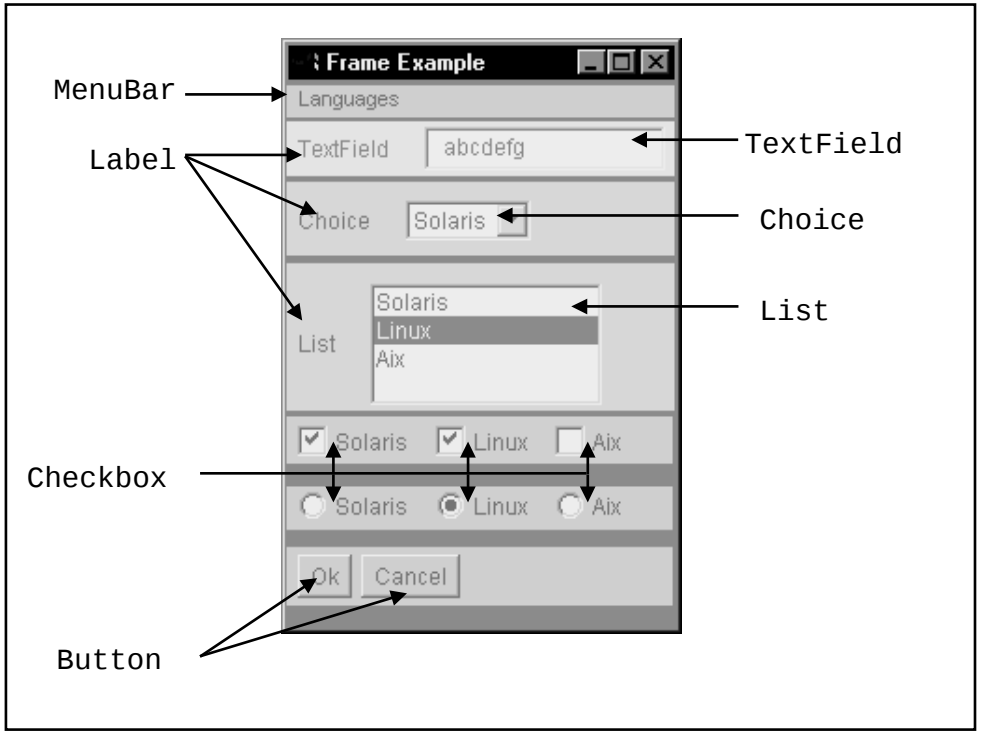


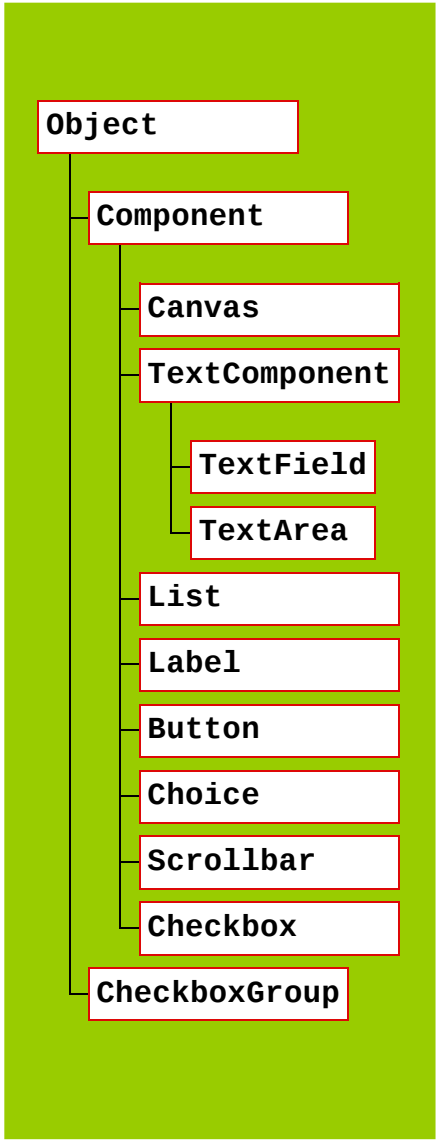
Classes d'interaction

`Canvas` pour le dessin

`Choices` est une sorte de combobox

`CheckboxGroup` composant logique



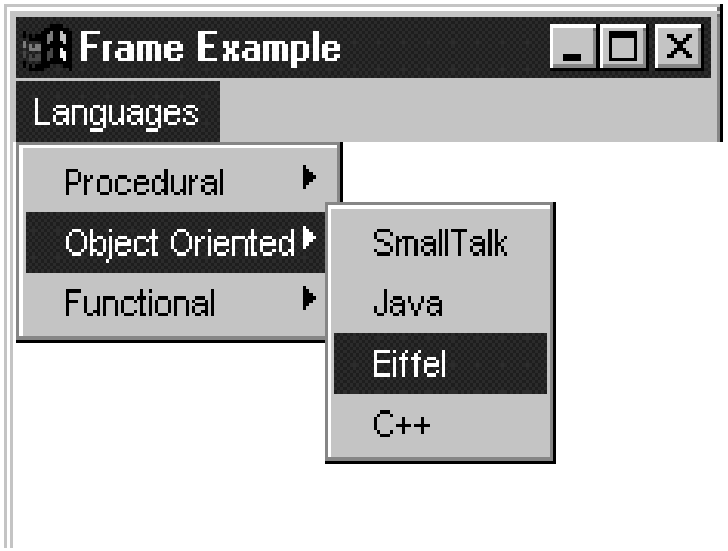


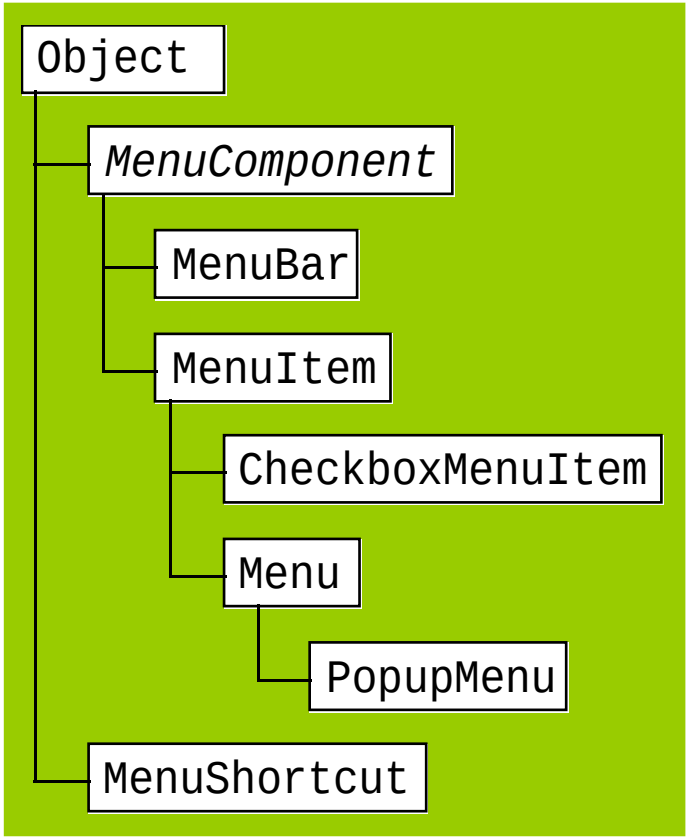
Menus

MenuComponent est abstraite

Menu est une sous-classe de **MenuItem** par le *design pattern* des conteneurs

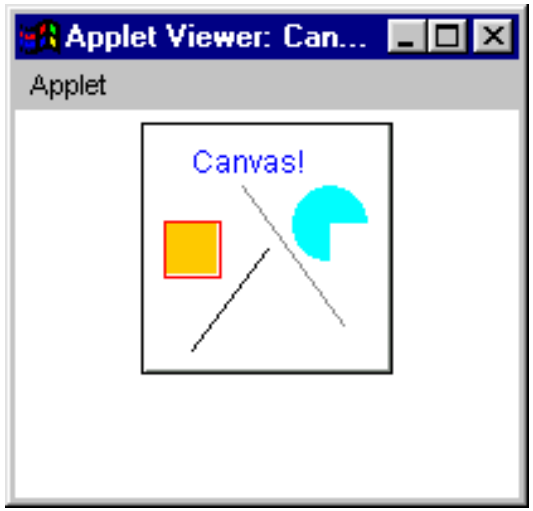
Les raccourcis sont adaptés à la plate-forme





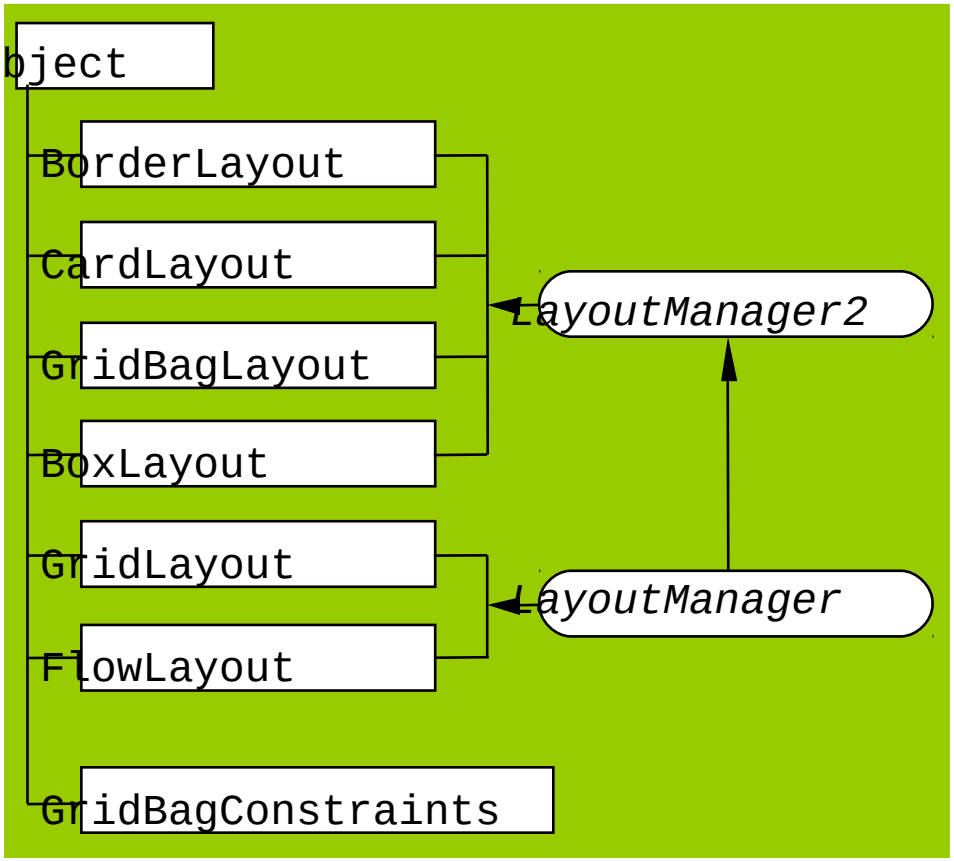
Graphique

- `Graphics` fournit à la fois le canal d'affichage et les outils de dessin
- `Image` pour les images
- `Point`, `Polygon`, `Rectangle`
- `Font`, `FontMetrics` pour les polices
- `Color`
- ***`Java2D`*** a beaucoup de possibilités



Layouts : gestionnaires de géométrie

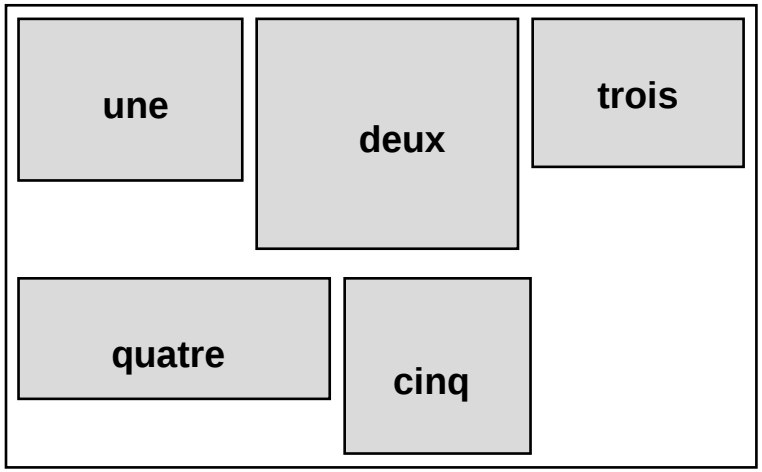
- Gère la disposition des composantes filles dans un conteneur
- Les gestionnaires par défaut sont
  - `BorderLayout` pour
    - `Window`
    - `Frame`
    - `Dialog`
  - `FlowLayout` pour
    - `Panel`
    - `Applet`



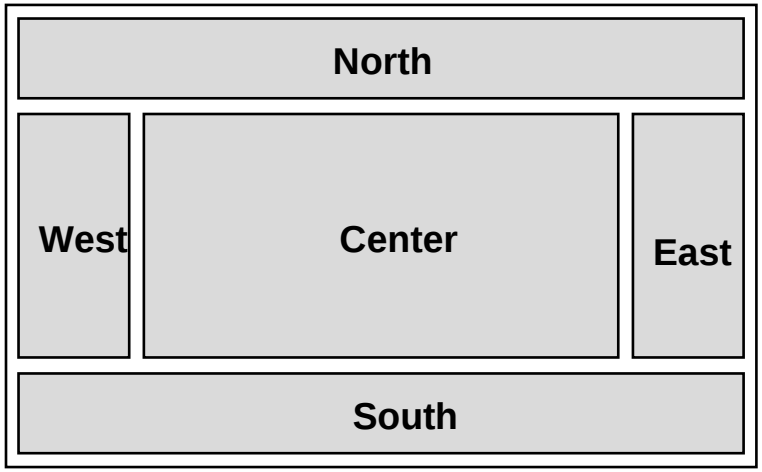
- **`BoxLayout`** est nouveau et utile
- **`LayoutManager`** et **`LayoutManager2`** sont des interfaces

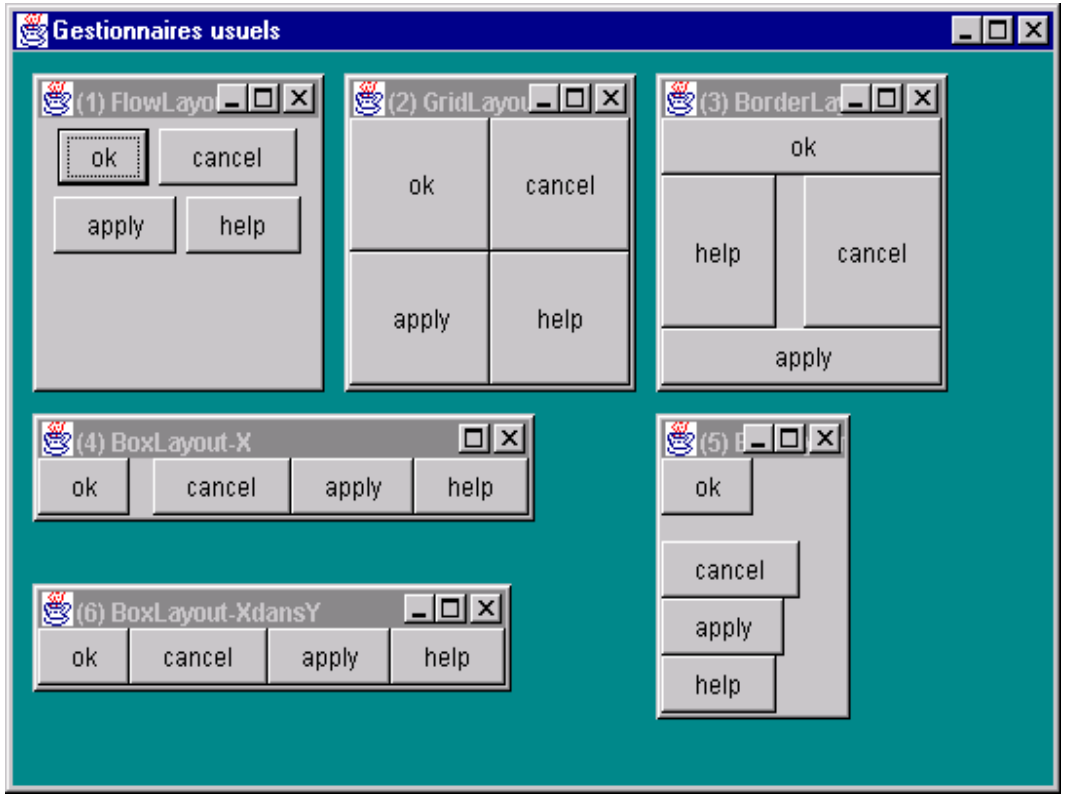
Exemples

- FlowLayout



- BorderLayout



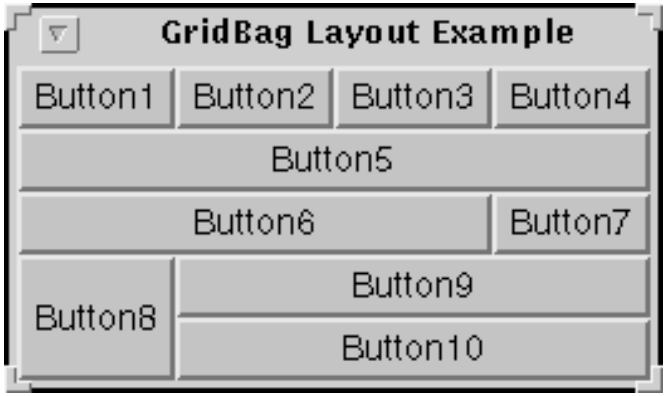


Exemples

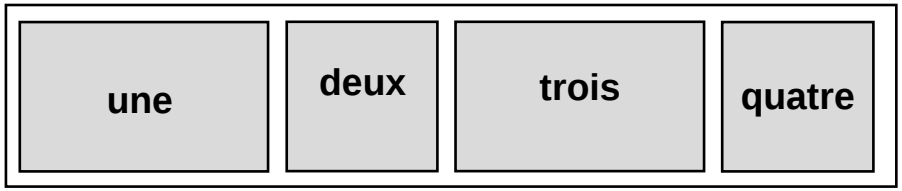
- **`GridLayout`** : chaque cellule de même taille

une	deux	trois
quatre	cinq	six

- **`GridbagLayout`** : grille, lignes et colonnes de taille variable

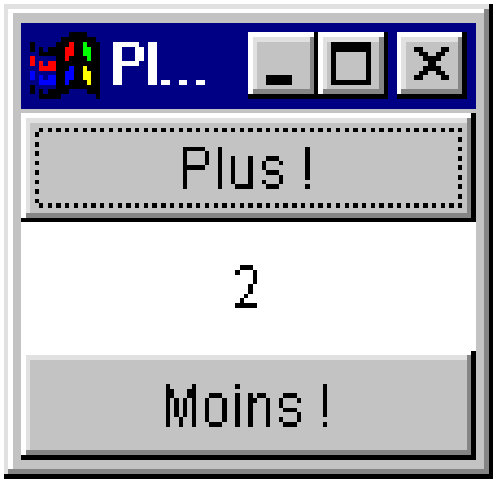


- **`BoxLayout`** (horizontal) : cellules de même hauteur, de largeur variable



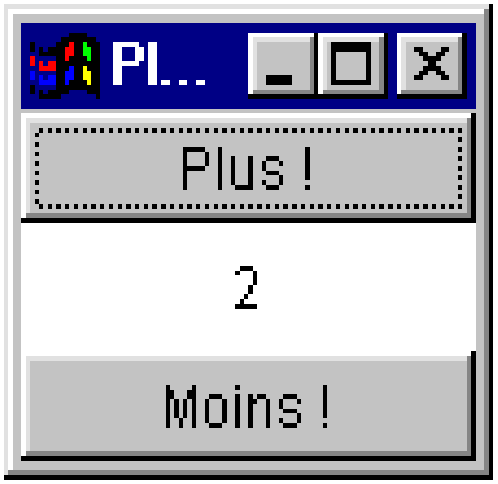
Gestion des évènements

- Un composant enregistre des ***auditeurs d'évènements*** (**Listeners**).
- Lorsqu'un événement se produit dans un composant, il est ***envoyé*** aux auditeurs enregistrés.
- Chaque auditeur définit les actions à entreprendre, dans des méthodes aux noms prédéfinis.



- Exemple:
- Un **`Button`** enregistre des **`ActionListener`**
- Lors d'un clic, un **`ActionEvent`** est envoyé aux **`ActionListener`** enregistrés.
- Ceci provoque l'exécution de la méthode **`actionPerformed`** de chaque **`ActionListener`**

Exemple



- Une classe de boutons

```
class MonBouton extends Button {
  int incr;

  MonBouton(String titre, int incr) {
    super(titre);
    this.incr = incr;
  }

  int getIncr() { return incr; }
}
```

- Une classe d'étiquettes auditrices

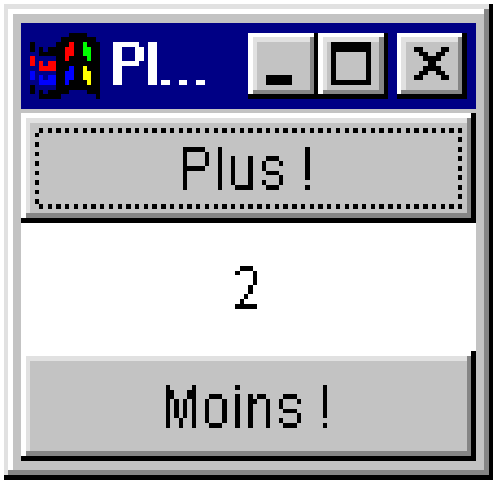
```
class ListenerLabel extends Label implements ActionListener {

  ListenerLabel() { super("0", Label.CENTER);}

  public void actionPerformed(ActionEvent e) {
    MonBouton b = (MonBouton)e.getSource();
    int c = Integer.parseInt(getText());
    c += b.getIncr();
    setText(Integer.toString(c));
  }
}
```

Exemple (fin)

- Le cadre




```
class PlusouMoins extends Frame {
  public PlusouMoins() {
    super("Plus ou moins");
    Button oui = new MonBouton("Plus !", +1);
    Button non = new MonBouton("Moins !", -1);
    Label diff = new ListenerLabel();
    add(oui, "North");
    add(diff,"Center");
    add(non,"South");
    oui.addActionListener((ActionListener) diff);
    non.addActionListener((ActionListener) diff);
  };

  public static void main (String[] argv) {
    Frame r = new PlusouMoins();
    r.pack();
    r.setVisible(true);
    r.addWindowListener(new WindowCloser());
  }
}
```

- Et pour fermer

```
class WindowCloser extends WindowAdapter {
  public void windowClosing(WindowEvent e) {
    System.exit(0);
  }
}
```

Qui est à la manœvre ?

- Le modèle MVC (*model -view -controller*)
  - le modèle gère les données
  - la vue affiche les données
  - le contrôleur gère la communication et les mises-à-jour
- Origine : Smalltalk


```
class PlusouMoinsMVC {

   Model model;
   View view;
   Controller control;

   public PlusouMoinsMVC() {
     model = new Model();
     control = new Controller();
     view = new View(control);

     control.setModel(model);
     control.setView(view);
     view.setVisible(true);
     view.addWindowListener(new WindowCloser());
   }

   public static void main (String[] argv) {
     PlusouMoinsMVC r = new PlusouMoinsMVC();
   }
}
```

Echange de messages

```
Vue a controlleur : bouton Up !
Controlleur a modele : changer compteur de 1
Controlleur a modele : que vaut compteur ? reponse = 1
Controlleur a vue : afficher compteur dans etiquette
Vue a controlleur : bouton Up !
Controlleur a modele : changer compteur de 1
Controlleur a modele : que vaut compteur ? reponse = 2
Controlleur a vue : afficher compteur dans etiquette
Vue a controlleur : bouton Down !
Controlleur a modele : changer compteur de -1
Controlleur a modele : que vaut compteur ? reponse = 1
Controlleur a vue : afficher compteur dans etiquette
...
```