Adapté de Géry Casiez et Eric Lecolinet

Programmation évènementielle et GUIs en java

Sylvain Malacria - www.malacria.fr

Programmation « classique »

Programme principal **initialise** et appelle des fonctions **dans un ordre pre-déterminé**:

- Les éventuels évènements utilisateurs sont « demandés » (programme en pause)

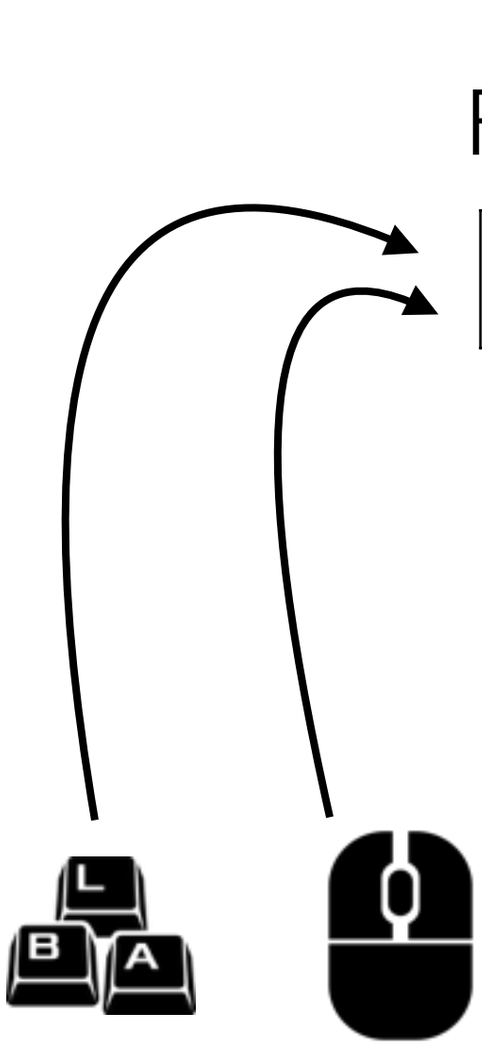
Programmation événementielle

Programme principal **initialise** des variables et les fonctions **réagissent aux événements**:

- Le déroulement est contrôlé par la survenue d'événements (dont les actions de l'utilisateur)
- Boucle principale qui traite les événements (enfouie dans la bibliothèque)

```
while (true){
        if(!queue.isEmpty()){
            event = queue.nextEvent();
            source = findSourceForEvent(event);
            source.processEvent(event);
        }
    }
```

File d’attente (queue FIFO)



```
public class MyObject  extends JFrame, …{
   ...
   public void keyDown(...){
        ...
   }
   public void mousePress(...){
        ...
   }
   public static void main (String [] args){
        new MyObject();
   }
}
```

Evénements Swing

Evenements « haut niveau »

– **ActionEvent:** **activer** un bouton, un champ de texte

– **TextEvent:** modification du texte dans un champ de texte

– etc.

Evenements « bas niveau »

– **MouseEvent:** appuyer, relâcher, bouger la souris

– **KeyEvent:** appuyer, relâcher une touche du clavier

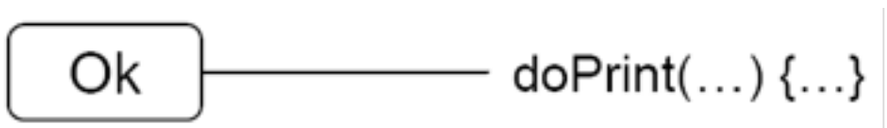
– **WindowEvent:** fermeture, déplacement des fenêtres

– etc.

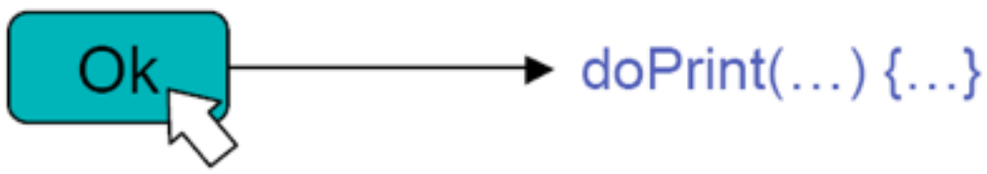
Lien composant-application

Fonctions de rappel (callbacks)

- Enregistrées dans le composant à sa création (abonnement)



- Appelées lorsque l'une des opérations du composant est activée (notification)

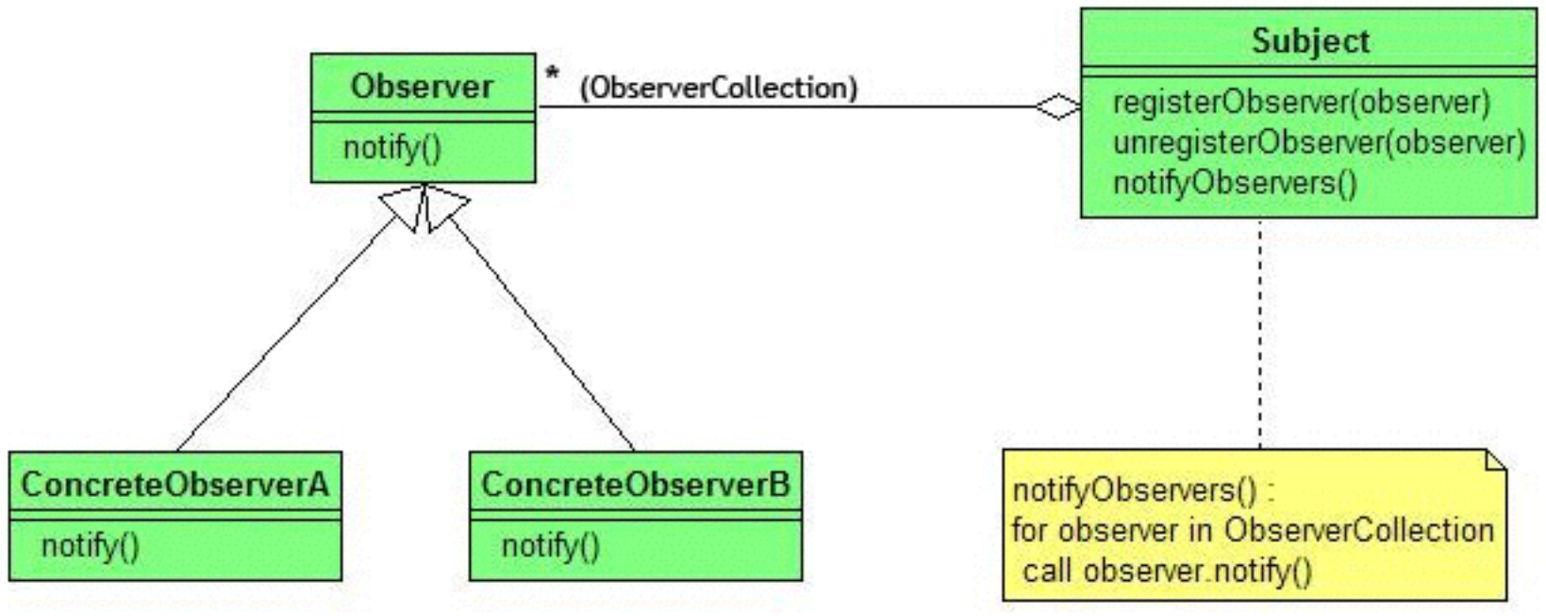


Utilisation du patron de conception **Observateur**

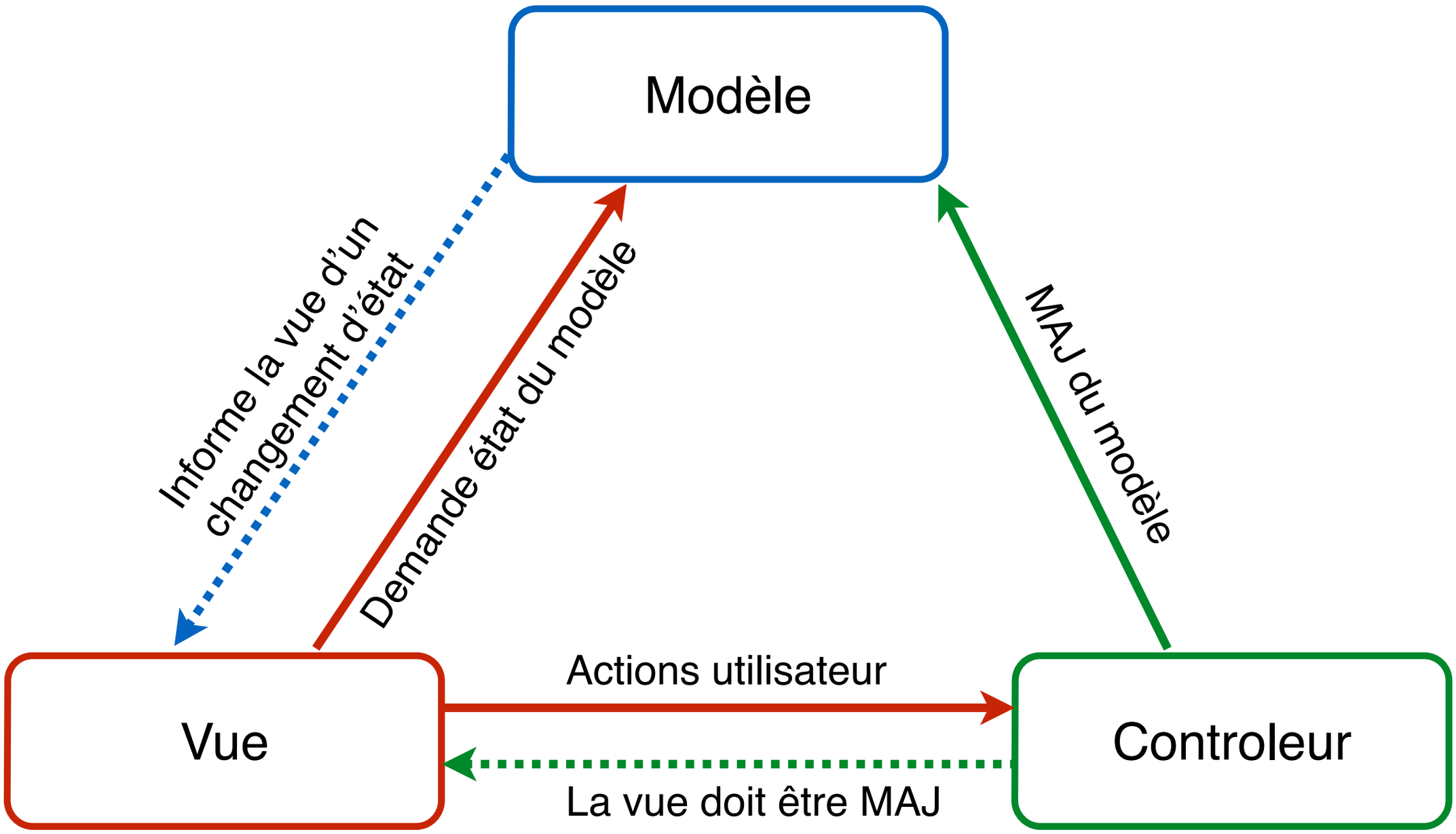
Pattern observateur/observé

Principe

- associer un (des) observateurs(s) à un (des) objet(s) observé(s)
- observateur(s) notifié(s) automatiquement quand une certaine condition se produit sur un observé



MVC


Modèle
Informe la vue d'un changement d'état
Demande état du modèle
MAJ du modèle
Actions utilisateur
Vue
Controleur
La vue doit être MAJ

MVC

- Représente le comportement de l'application
  - Contient les données manipulées par l'application
  - Assure la gestion de ces données et leur intégrité
  - Mise à jour des données (insertion, suppression, modification)
  - Méthodes pour accéder à ces données

- Possibilité d'avoir plusieurs vues partielles des donnée

MVC

- Correspond à l'interface avec laquelle l'utilisateur interagit
- Présentation des résultats renvoyés par le modèle
- Réception des actions de l'utilisateur (clics souris..) et envoi de ces informations au contrôleur
- La vue n'effectue aucun traitement, se contente d'afficher les traitements effectués par le modèle

MVC

- Gestion des événements de synchronisation pour mettre à jour le modèle
- Reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer
- Si changement des données, le contrôleur demande la modification des données au modèle
- Le contrôleur n'effectue aucun traitement, ne modifie aucune donnée
- Peut notifier la vue qu'il y a eu une erreur

MVC

- Modèle asymétrique
  - Une paire Contrôleur/Vue est associée à un seul modèle
  - Un modèle peut se voir associé plusieurs paires Contrôleur/ vue
- Listes des dépendants et notification
  - Les paires Contrôleur/Vue d'un modèle sont enregistrées dans une liste de « dépendants »
  - Lorsque l'état du modèle est modifié, tous les dépendants sont notifiés

Implémentation MVC

3 classes abstraites définissent les comportements génériques des composants MVC

- Class Model
  - Mécanismes permettant la gestion des dépendants
  - Mécanismes de diffusion des notifications
- Class View
  - Affiche une représentation du modèle
  - Transmet les interactions utilisateur au contrôleur
- Class Controller
  - Permet le contrôle et la manipulation d'un modèle et d'une vue

MVC: Exemple d'interaction

1. L'utilisateur clique sur un bouton de l'interface
2. Le contrôleur est notifié de l'action de l'utilisateur et vérifie la cohérence de cette action
3. Si l'action est cohérente, le contrôleur notifie le modèle de l'action de l'utilisateur (éventuellement modification de l'état du modèle)
4. Le modèle notifie les vues qu'un changement d'état a eu lieu
5. Les vues utilisent le modèle pour générer l'interface appropriée
6. Attente des événements suivants

Conclusion sur MVC

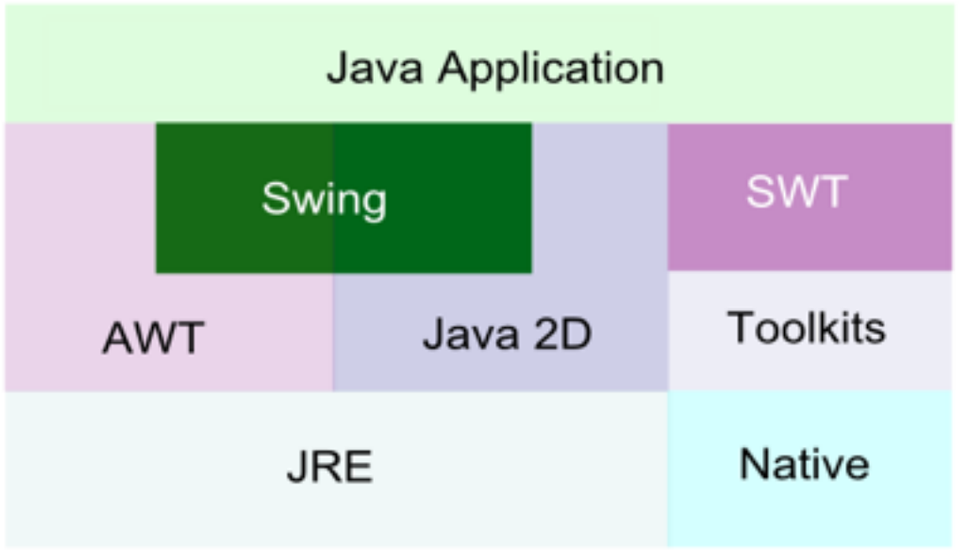
- Avantages
  - Vues multiples synchronisées
  - Vues et contrôleurs modulaires
  - Développement de composants réutilisables

- Inconvénients
  - Complexité de communication entre les composants (principalement entre C et V)

Toolkits graphiques Java

Il y en a trois !

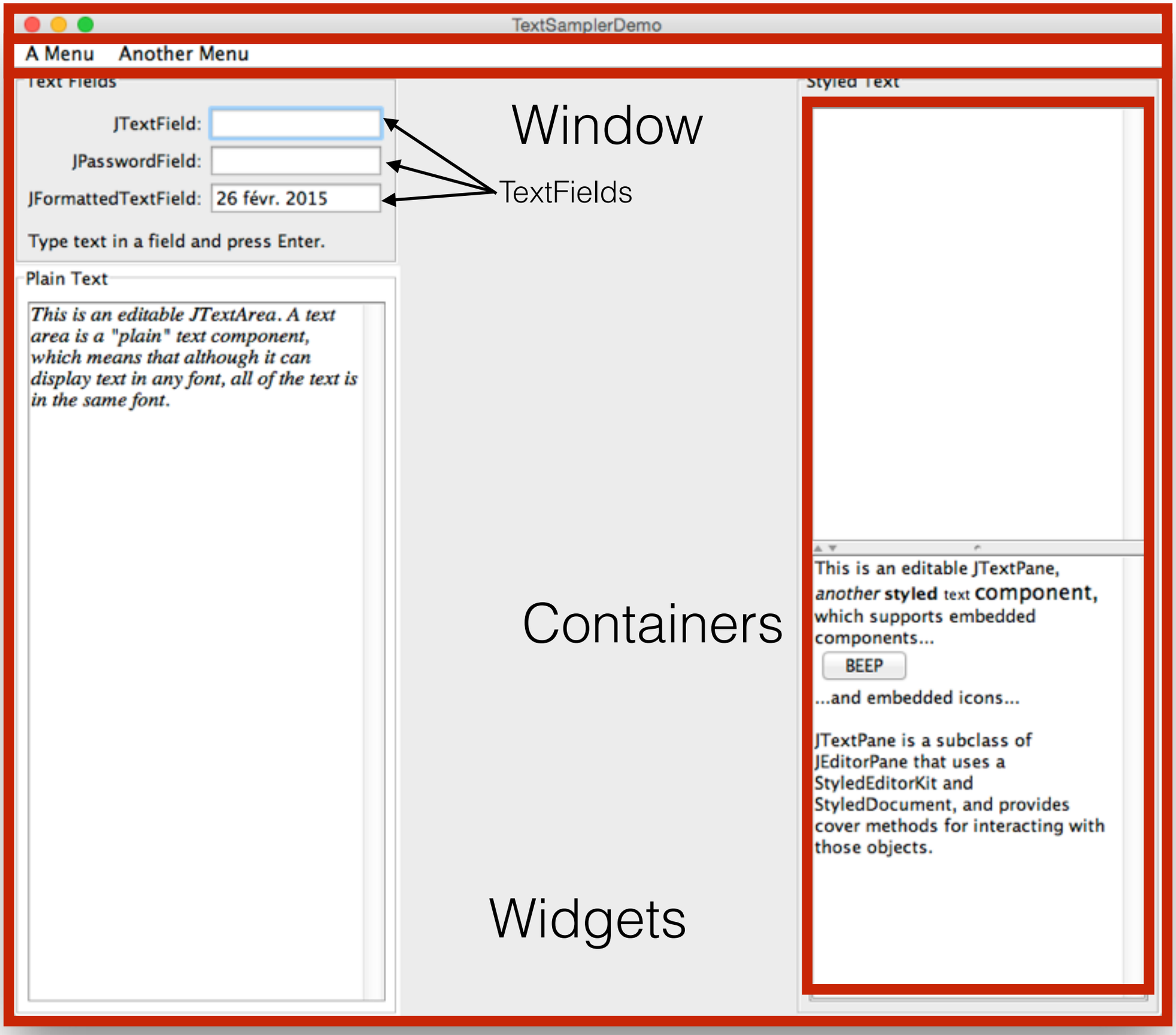
– **AWT Components**, obsolète
– **SWT** libre, initié par IBM / Eclipse
– **Swing** supporté par Sun/Oracle
– tous (+ ou -) multi-plateformes



Java Graphics - The Layer Cake

Swing repose sur AWT mais **Swing est différent** d'AWT

Menu

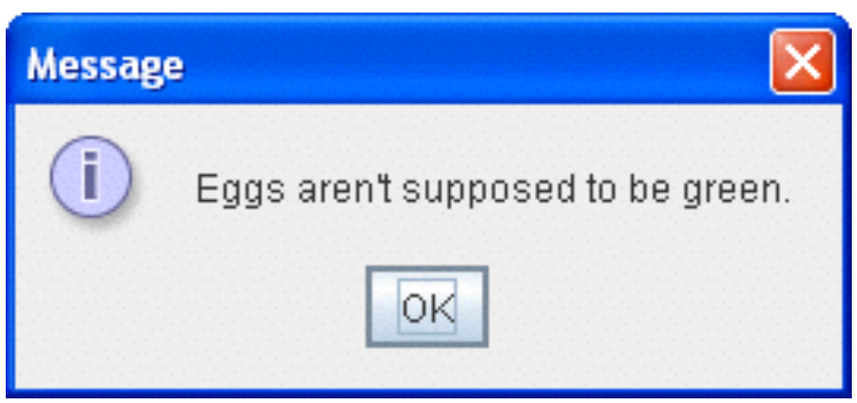

TextSamplerDemo
A Menu
Another Menu
Text Fields
JTextField:
JPasswordField:
JFormattedTextField:
26 févr. 2015
Type text in a field and press Enter.
Plain Text
This is an editable JTextArea. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.
Styled Text
This is an editable JTextPane, another styled text component, which supports embedded components...
BEEP
...and embedded icons...
JTextPane is a subclass of JEditorPane that uses a StyledEditorKit and StyledDocument, and provides cover methods for interacting with those objects.
Window
TextFields
Containers
Widgets

Composants graphiques

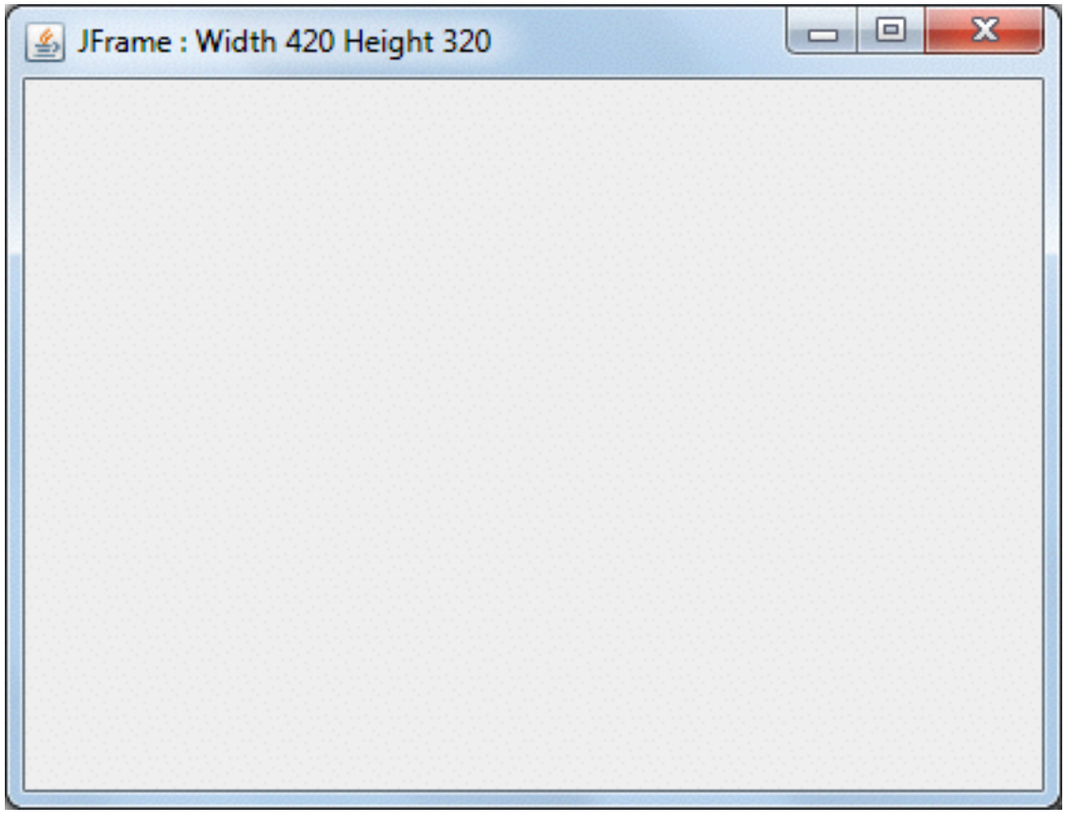
Haut niveau

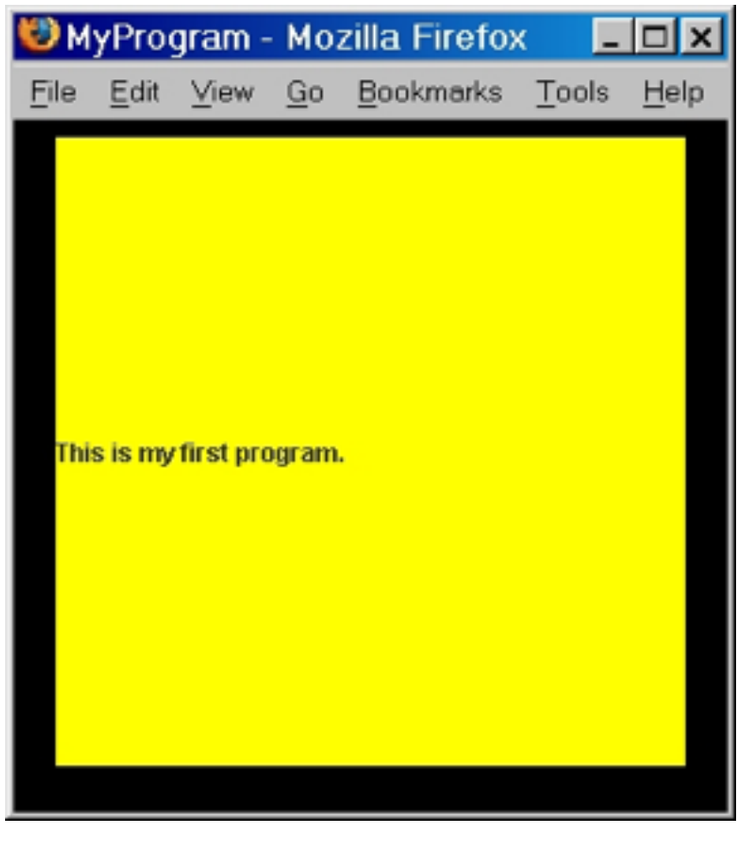
– **Fenêtres**

– **Dialog** (certains prédéfinis)



– **Applet** (web)





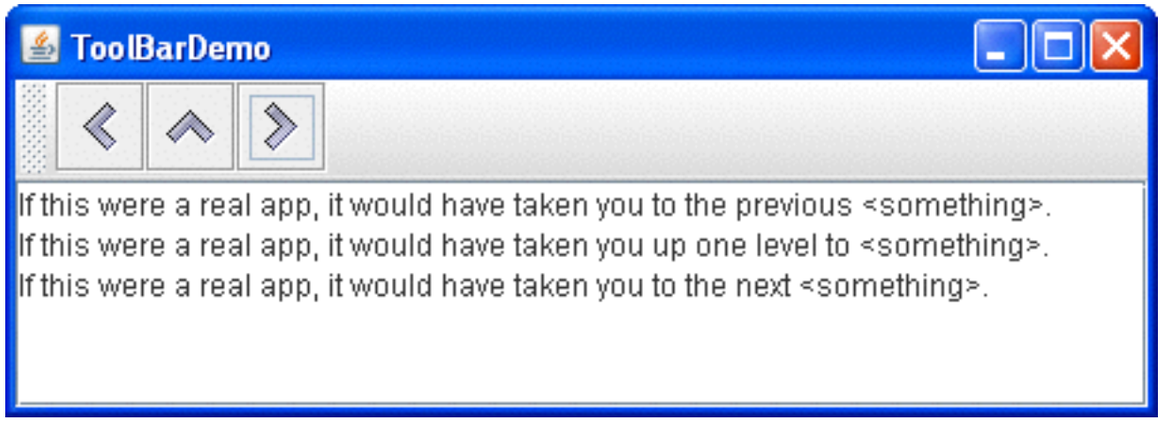
Composants graphiques

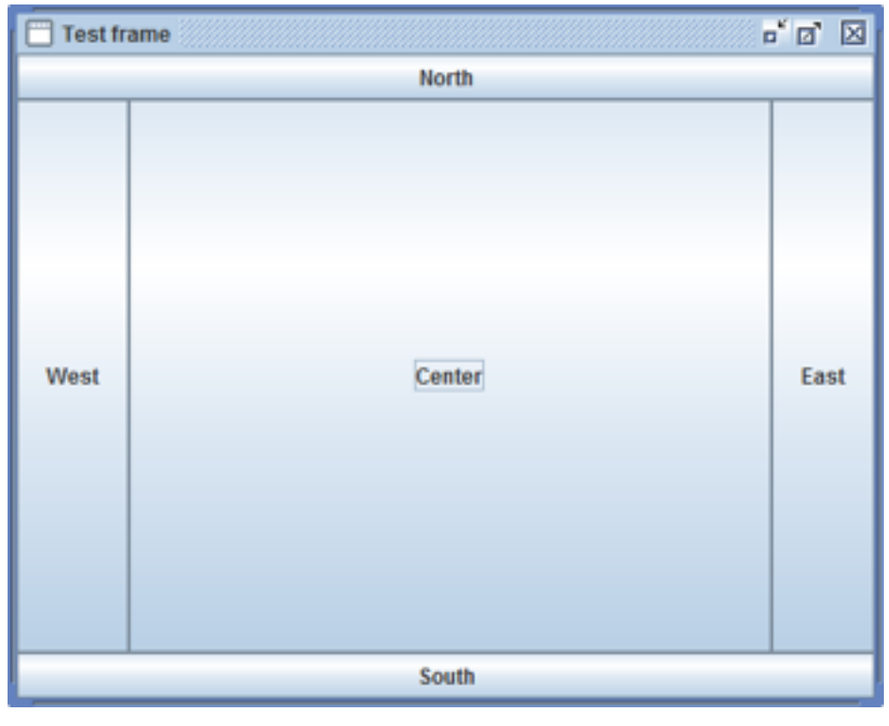
Containers

– **Combiner** (JPanel)

– **Manipuler** (Onglets, sliders, barres de défilement)

– **Organiser des composants de manière spécifique**
(Barres d’outils, menus, etc.)

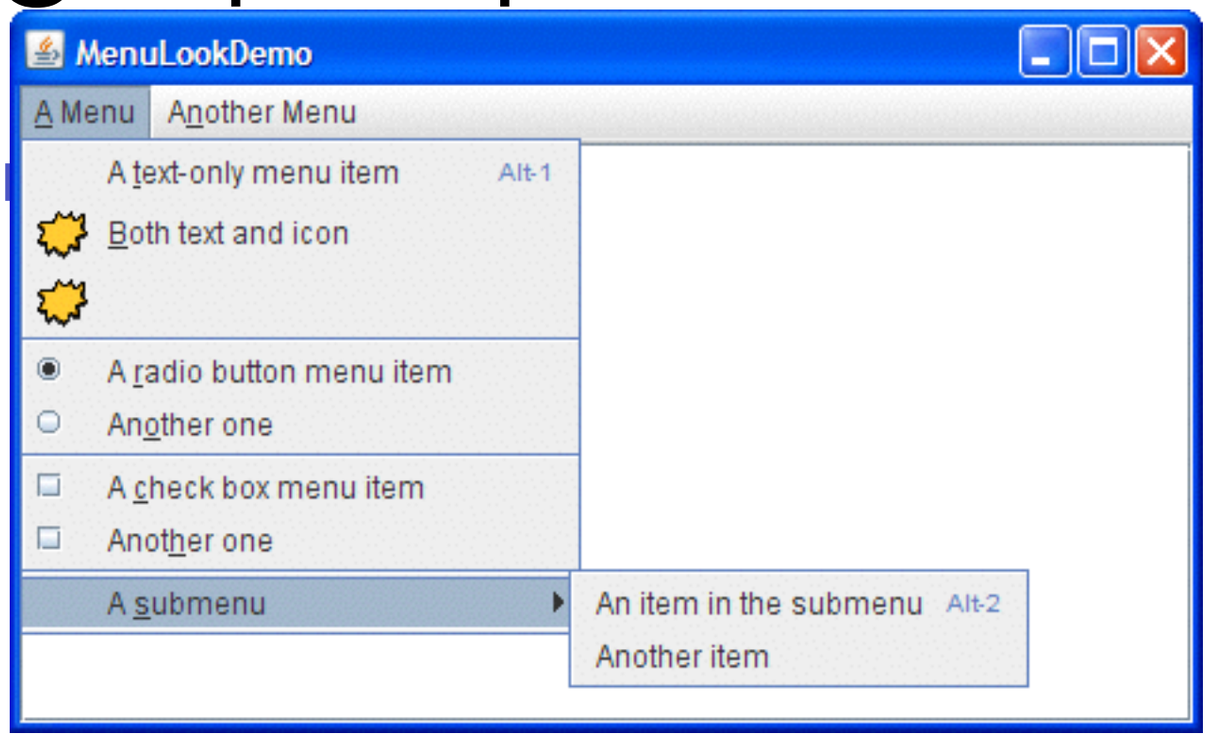




Composants graphiques

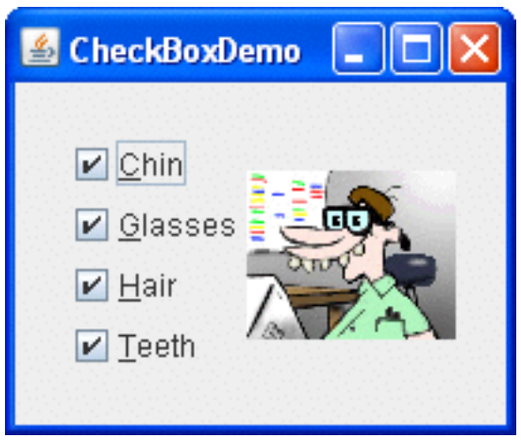
Containers (panel, viewport, scrollpa

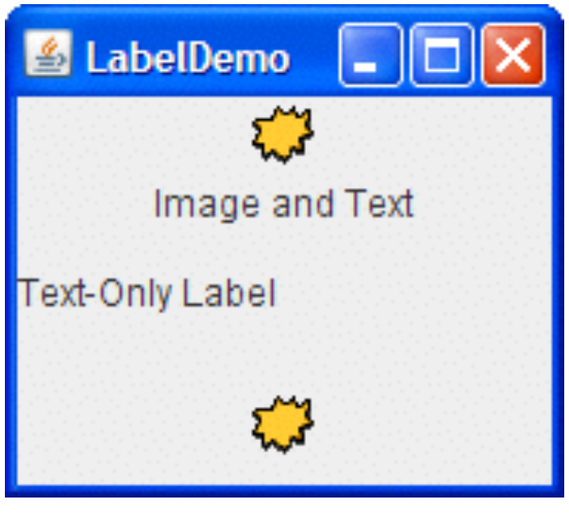
- **Boutons**
- **Text**
- **Sliders, etc**

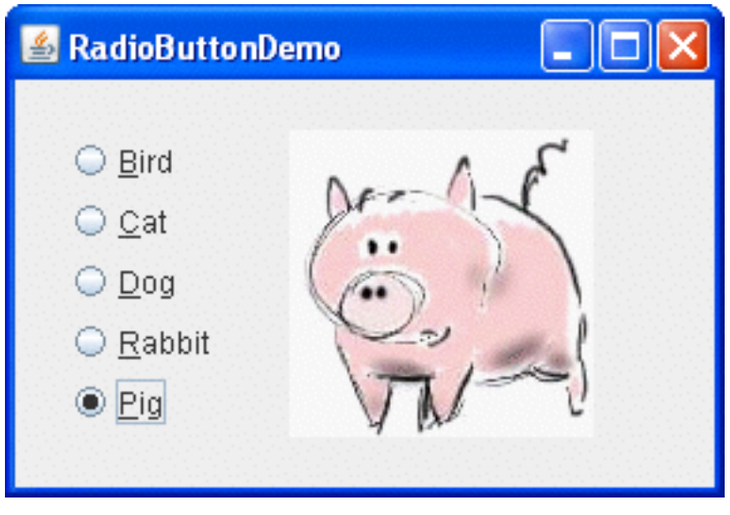


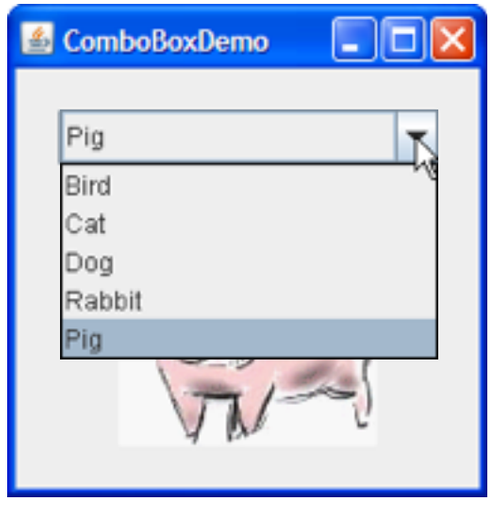
Composants graphiques

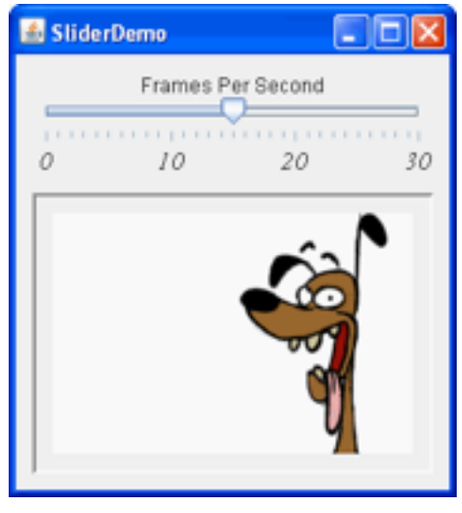
Interacteurs (Widgets)

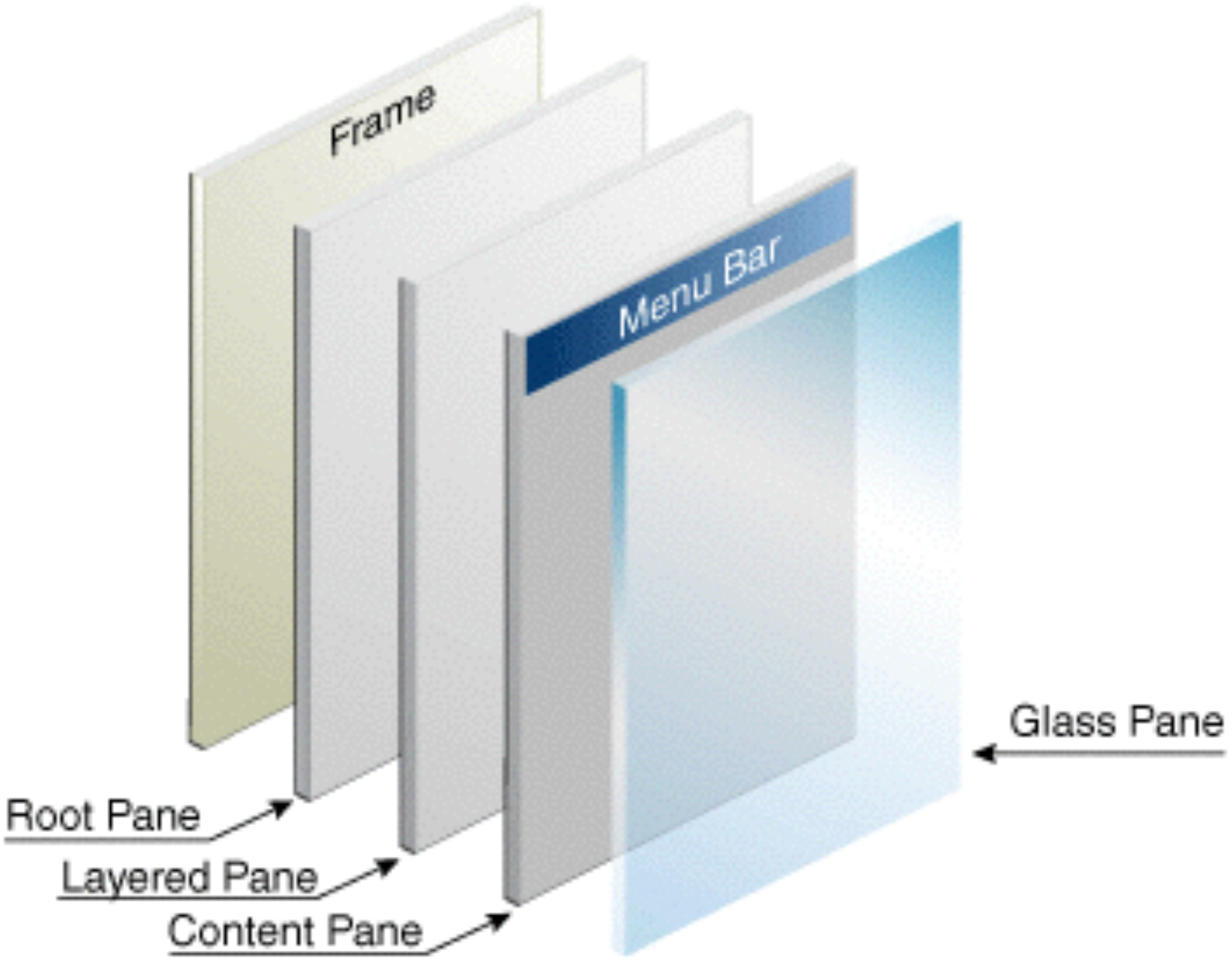








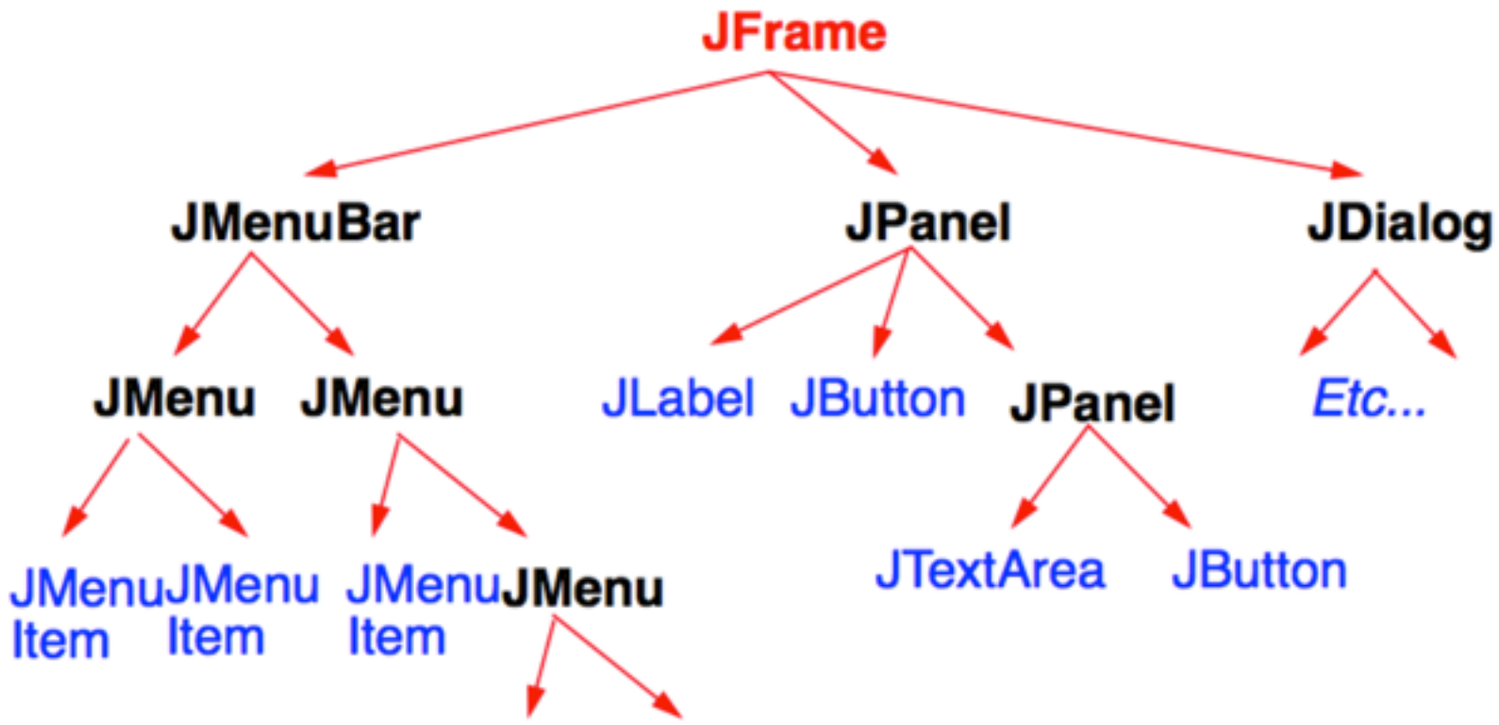



Frame
Menu Bar
Glass Pane
Root Pane
Layered Pane
Content Pane

Disposition - hiérarchie

Arbre d'instanciation

– arbre de filiation des instances de composants graphiques



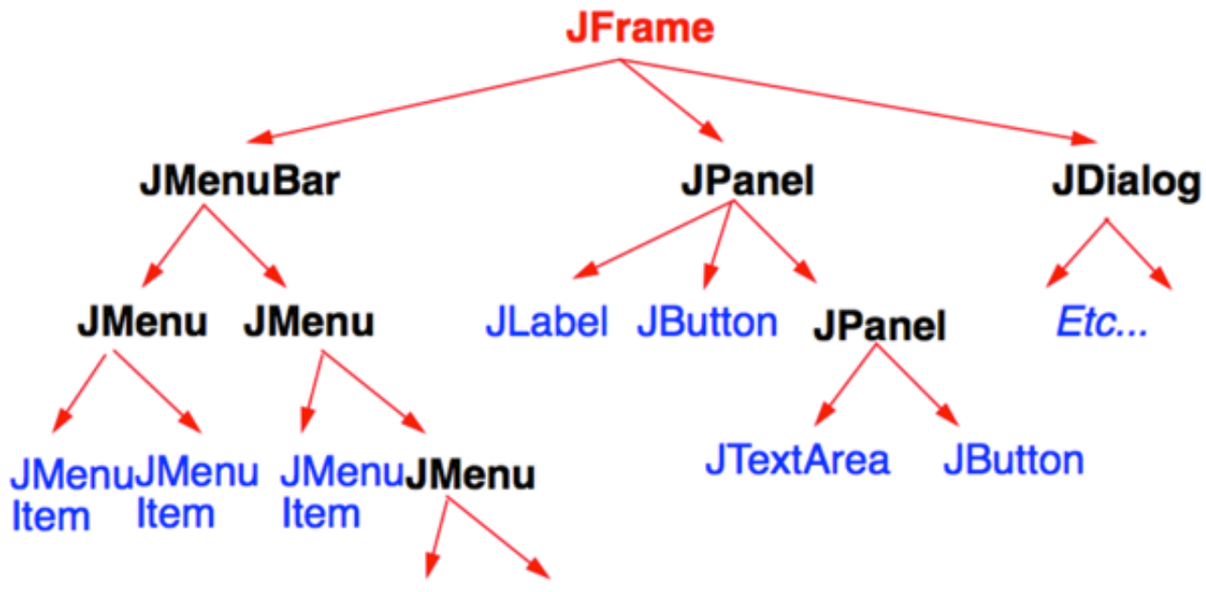
Arbre d'instanciation

Objet de + haut niveau

– JFrame ou JApplet

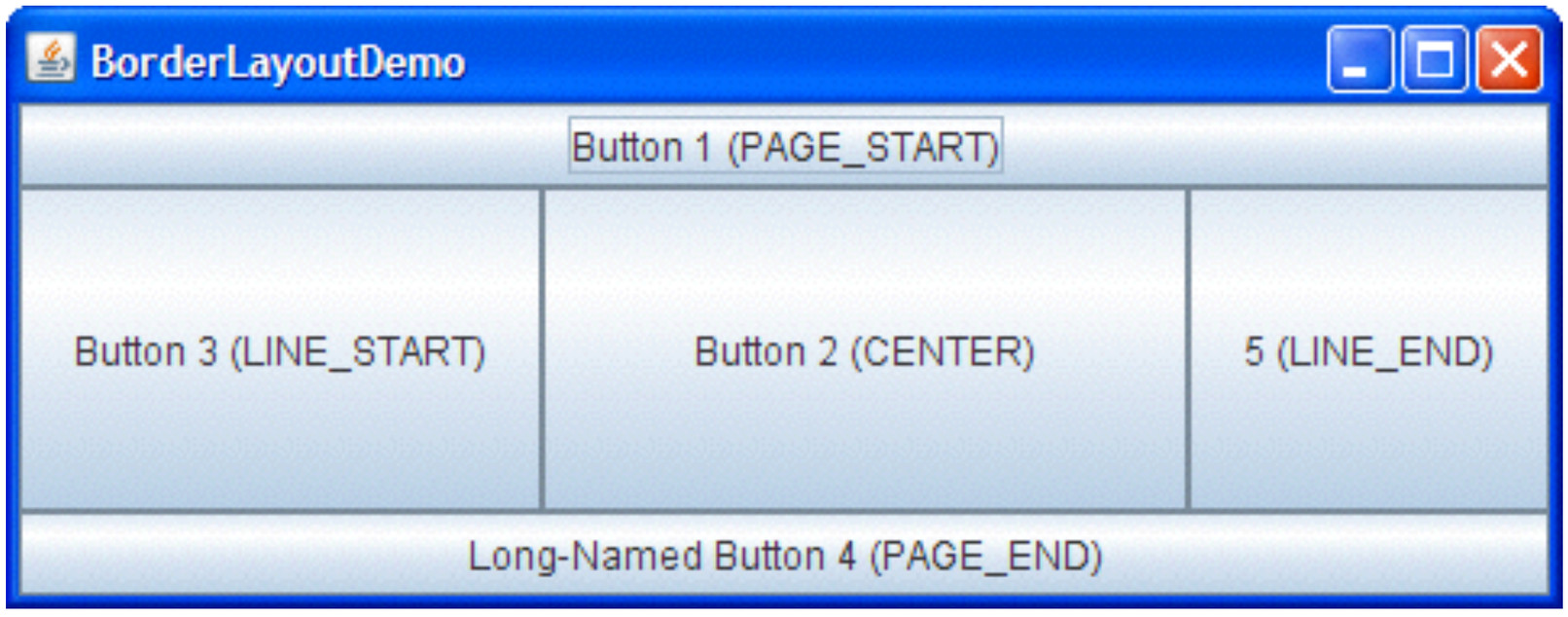
Conteneur peuvent être imbriqués

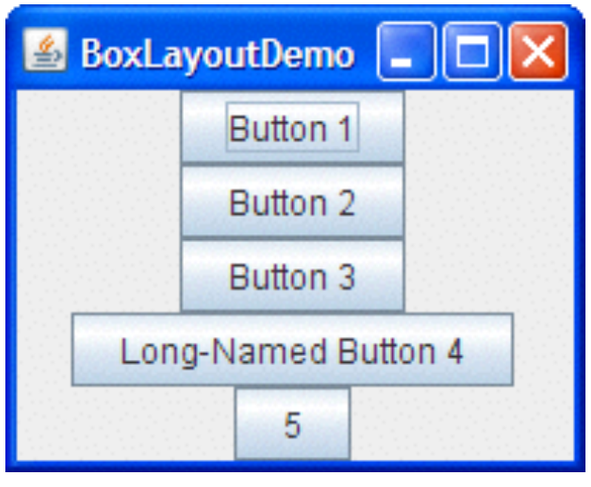
– En particulier JPanel

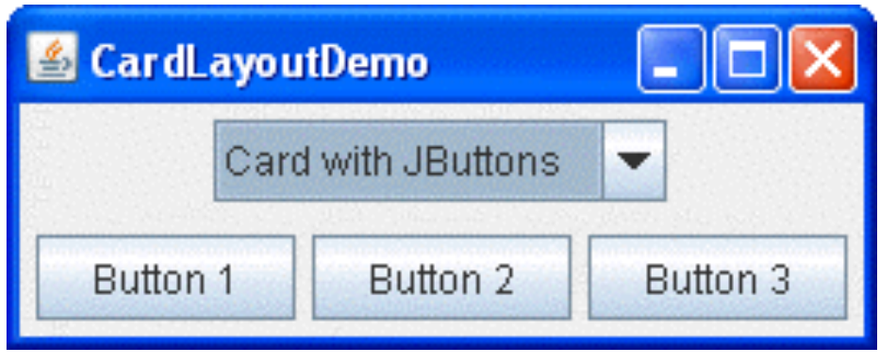


Disposition

Layout managers



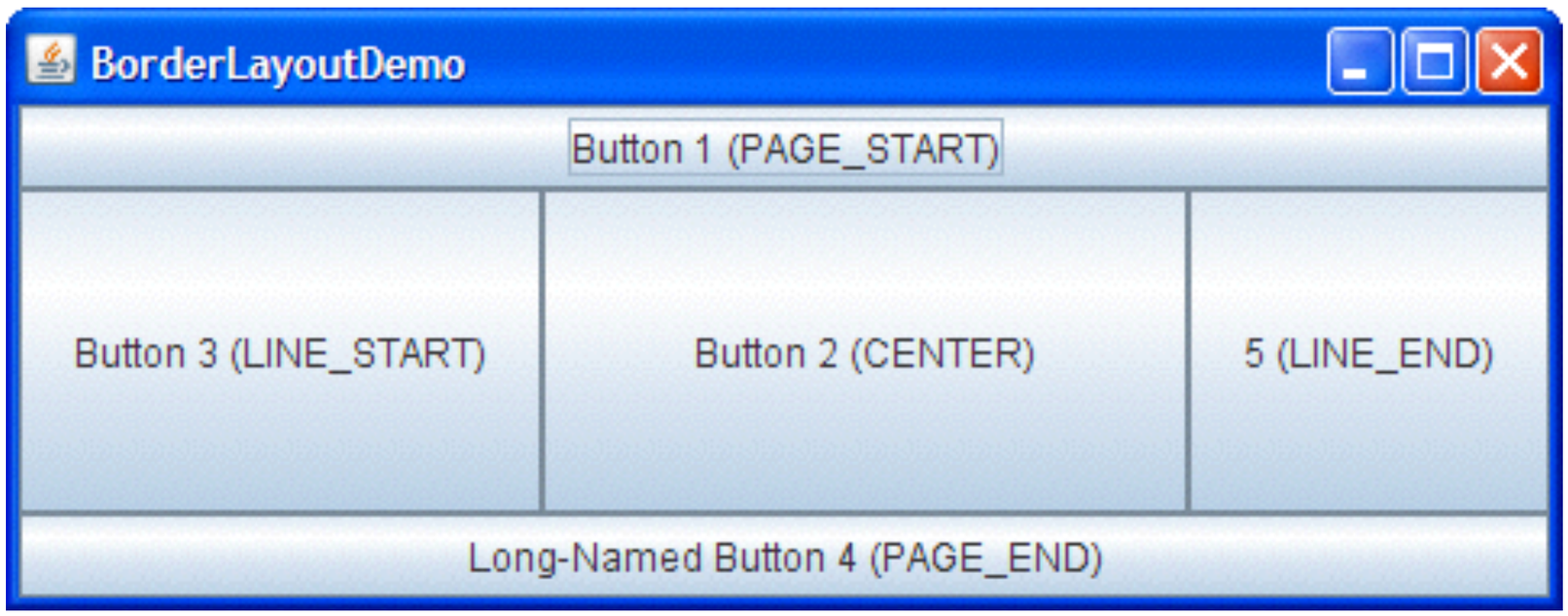




Disposition

Border Layout

– Layout manager par défaut pour les ContentPane
(conteneur de composant haut niveau comme fenêtre, applet, etc.)
– Place les composants dans **5 zones**
(North, South, East, West, Center)
– Un seul composant par zone (composants imbriqués nécessaires)



Disposition

Flow layout

– Layout manager par défaut pour les JPanels

– Place les composants sur une ligne, de gauche à droite (tant qu'il y a de la place)

Disposition

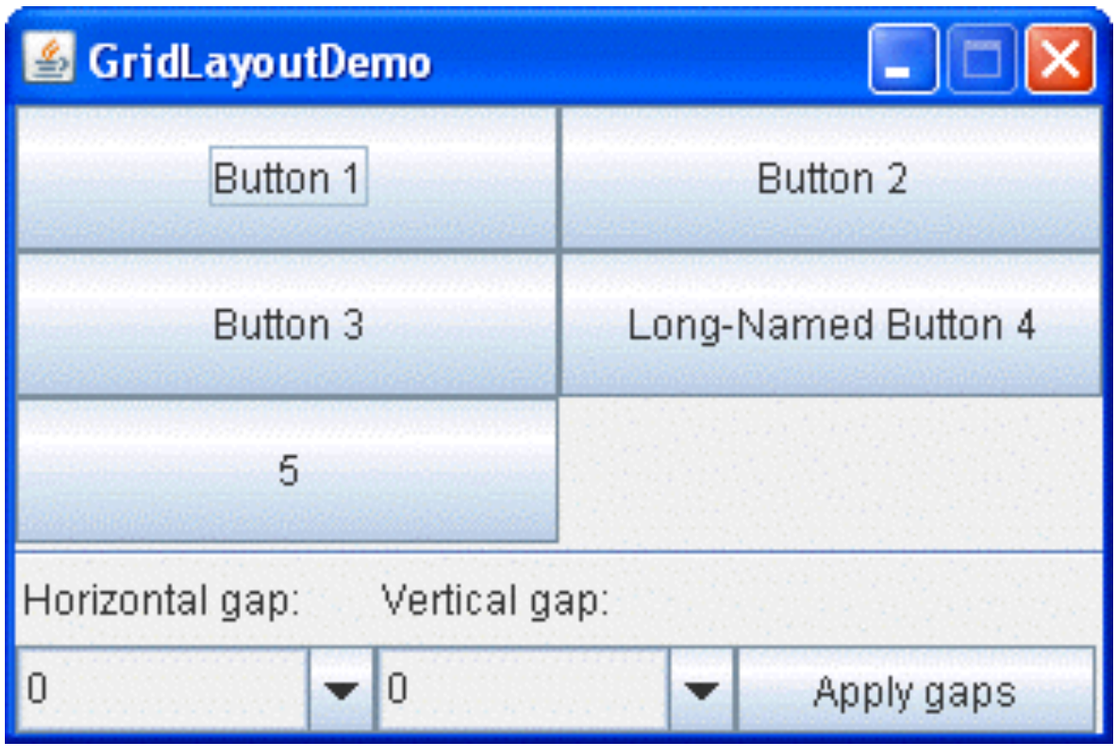
Grid Layout

– Dispose les éléments dans une grille ***n x m***

– Pas indispensable de spécifier *n* ET *m*

– Très utile pour prototypage rapide

– Dans cet exemple: 1 gridlayout ***0 x 2*** et un autre ***2 x 3***



Disposition

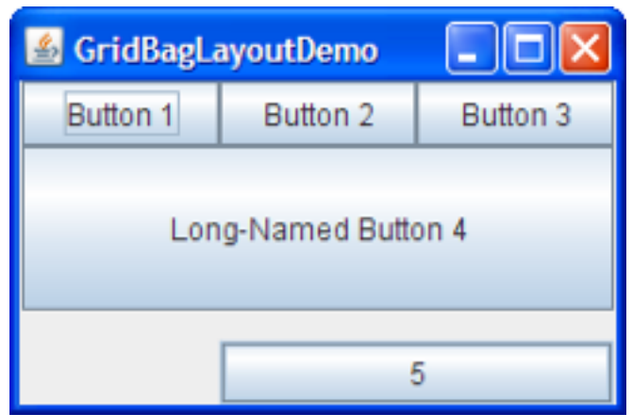
Other layouts

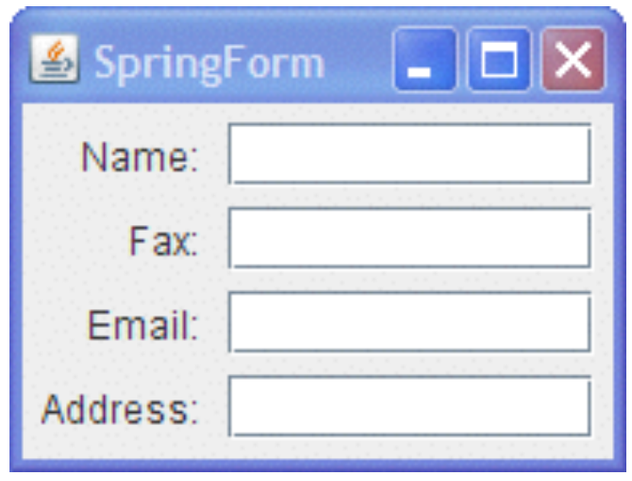
– Nombreux layout managers différents

– Nécessité éventuelle de combiner et/ou imbriquer différents layouts

– Recherchez sur le web :

« a visual guide to layout managers »





Comment ajouter un listener?

Exemple de l'ActionListener (JButton)

– **ActionEvents** envoyés quand le bouton est cliqué

– **ActionListener** interface à implémenter qui intercepte ces ActionEvents et proceed

```
7⊖    public TESTB(){
8         JButton button = new JButton();
```

Method Summary

Methods

Modifier and Type	Method and Description
void	`actionPerformed(ActionEvent e)` Invoked when an action occurs.

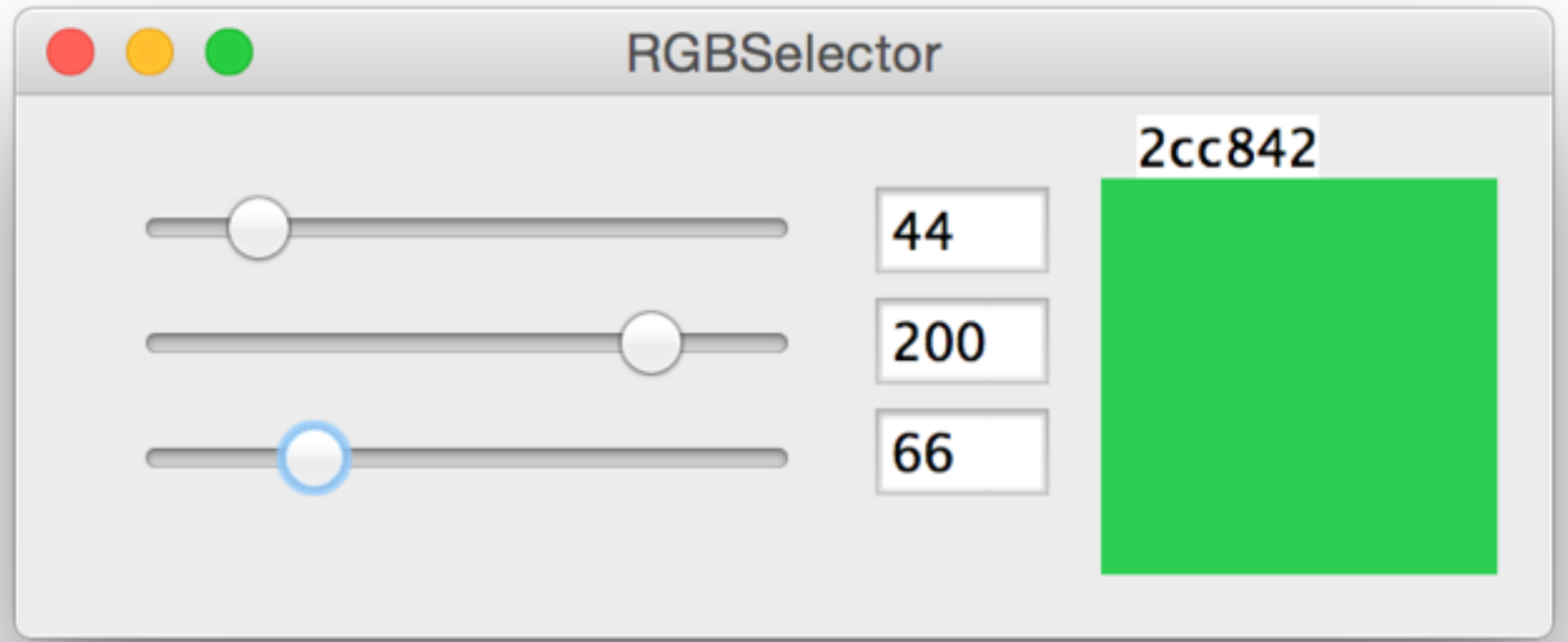
Method Detail

actionPerformed

`void actionPerformed(ActionEvent e)`

Invoked when an action occurs.

A simple RGB chooser



Avant de finir

Site web complet pour débuter avec Swing

– http://docs.oracle.com/javase/tutorial/uiswing/

Liste longue d'événements en fonction des composants

– Consultez la documentation !

Avant de finir

Evénements utiles pour animations

– classe javax.swing.Timer (attention, pas ~~java.util.Timer~~)

Timer timer = new Timer(speed, this);
timer.start();

speed = pause en ms entre chaque événement du timer

this = classe qui reçoit l'évènement (ActionListener)

Avant de finir

Notification centers

– Dispo sous Cocoa, Android …
– Un objet notifie avec une « clé »
– N’importe quel objet peut écouter (sous reserve de connaitre la clé)
– Plus souple, mais moins de contrôle