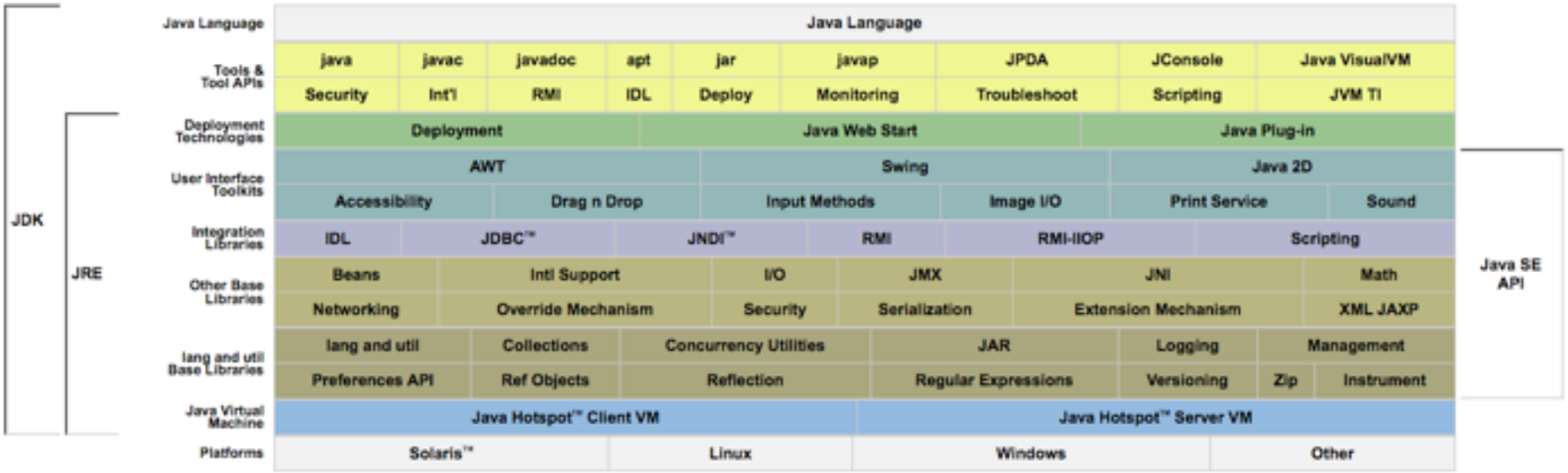
COMPOSANTS

Géry Casiez http://www.lifl.fr/~casiez
IHM Master 1 informatique - Université de Lille 1

Composants de la plateforme Java



La librairie AWT

- La librairie AWT (Abstract Window Toolkit) a été introduite dès les premières versions de Java.
- Utilisation de ressources propres au système encapsulées dans des abstractions
- L'affichage des composants est géré par le système
  - => l'apparence des composants est différente selon les plateformes
- Fonctionnalités assez rudimentaires

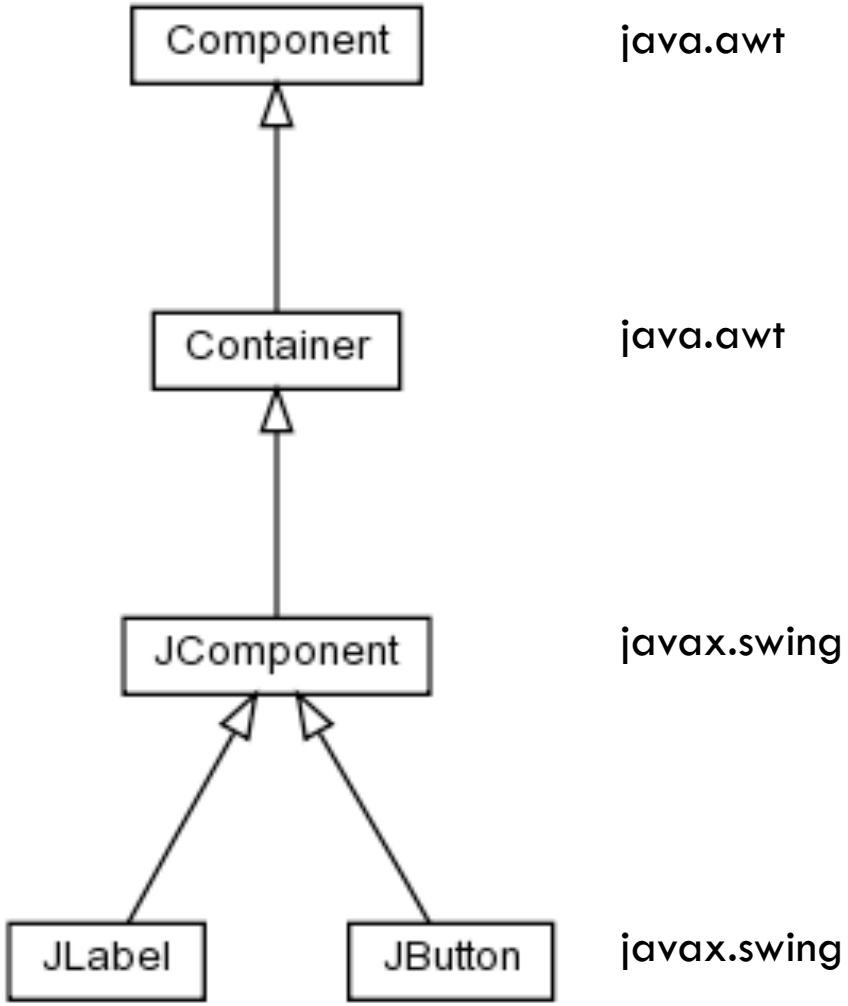
La librairie Swing

- Swing utilise des éléments d'AWT mais offre un grand nombre de nouveaux composants
- Les éléments graphiques ont la même apparence quelque soit le système d'exploitation
- Utilisation du Look-and-feel pour configurer la représentation visuelle
- Les noms des classes de la librairie Swing commencent par la lettre **J**

Widgets / Composants

- Le widget est un objet graphique interactif
- On parle de composant avec Swing

Composants



java.awt

java.awt

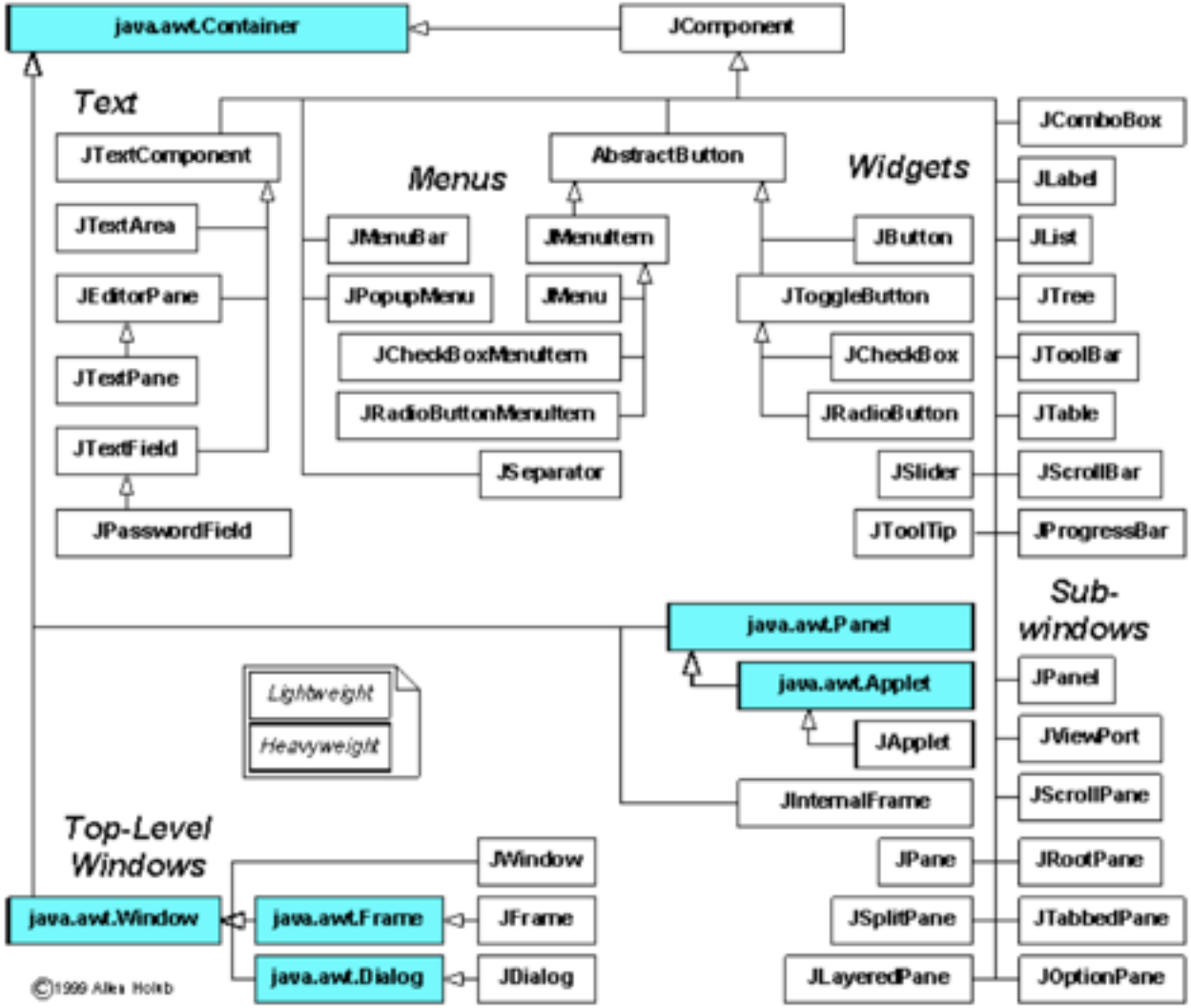
javax.swing

javax.swing

Composants

- Par héritage, les composants Swing sont également des composants AWT
- Les composant Swing se distinguent des composants AWT par la lettre J: JButton (Swing), Button (AWT)

Architecture de la librairie Swing



Propriétés des composants

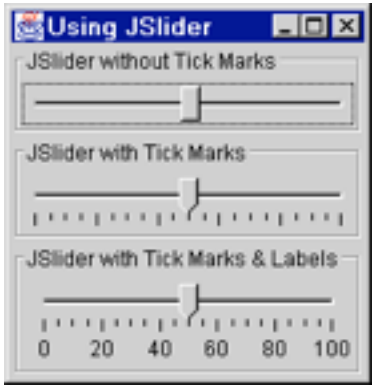
- Chaque composant Swing peut être personnalisé en apparence et en comportement en spécifiant des valeurs de propriétés
  - Utilisation des accesseurs de la classe du composant
- Toutes les propriétés héritées des super classes sont naturellement également disponibles pour le composant
  - Les propriétés de JComponent, Container et Component sont héritées par presque tous les composants Swing

Propriétés des composants

- Propriétés communes
  - Activation / désactivation d'un composant (méthode setEnable de JComponent)
  - Un composant peut être visible ou non (méthode setVisible de Jcomponent)
  - Taille
    - Dimension getSize() / Dimension getPreferredSize()
    - void setSize(Dimension rv) / void setPreferredSize(Dimension prefferedSize) – défini dans Component
  - Curseur: setCursor(un_curseur)

Propriétés des composants

- Propriétés communes
  - Tooltips, bulles d'aide: méthode setToolTipText définie dans la classe JComponent
  - Couleurs: setBackground, setForeground
  - Bordures: setBorder



Évolution des widgets

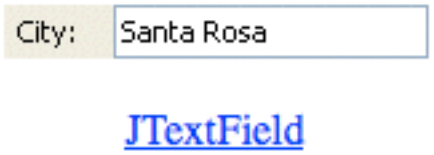
- Widgets de base du macintosh (1984)
  - Bouton (button)
  - Potentiomètre (slider)
  - Case à cocher (check box)
  - Bouton radio (radio button)
  - Champ texte (text field)
  - Boites de dialogue pour les fichiers (file open/save dialog)
- Ajouts ultérieurs: menus hiérarchiques, listes, combo box, tabs

Tâches élémentaires d'interaction

- Quelques exemples
  - Saisie
  - Sélection
  - Déclenchement
  - Défilement
  - Spécification d'arguments et de propriétés
  - …

Saisie

- Saisie de texte
  - Boite de saisie et clavier
- Saisie de quantités
  - Ex: potentiomètre
- Saisie de positions
  - Pointage
- Saisie de tracés



JTextField

JSlider

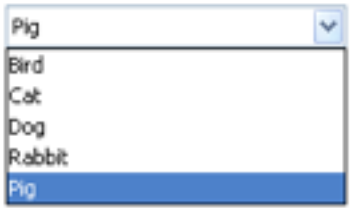
JSpinner

Sélection

- Choix d'un ou plusieurs éléments dans un ensemble
  - Cardinal fixe ou variable
  - Cardinal petit ou grand
- Exemples
  - Cardinal fixe: menu, boutons radio, cases à cocher
  - Cardinal variable: pointage, liste, saisie de nom ou combinaison de ces deux dernières techniques
- Sélection multiple: groupe ou intervalle, ajout et retrait

JRadioButton

JCheckBox



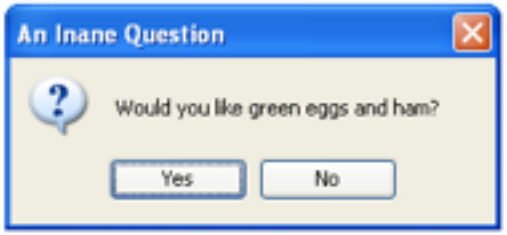
JComboBox

JList

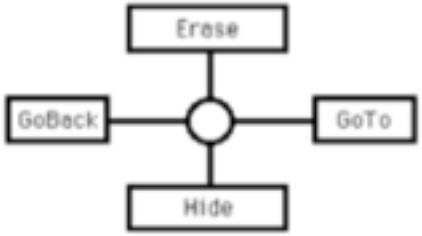
Sélection

- Menus
  - Déroulants
  - Pop-up
  - Palettes
- Organisation du menu
  - Hiérarchique
  - Circulaire

JMenu

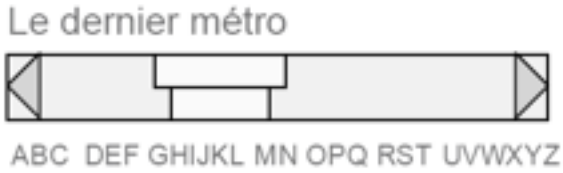


JDialog



Sélection

- Le *range slider*
  - Permet une sélection d'une plage de valeurs
- L'*alpha slider*
  - Permet de sélectionner rapidement un élément dans une longue liste
  - Exemple: de 13 à 24 secondes pour trouver un film dans une liste de 10 000



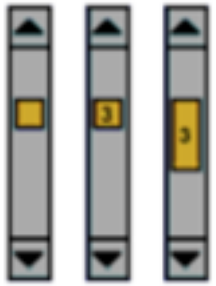
Déclenchement

JButton

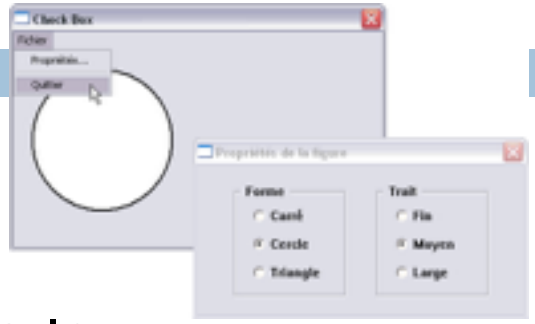
- Boutons et menus
- Cliquer-tirer (drag-and-drop)
  - L'action dépend de la source et de la destination
- Entrée gestuelle
  - Spécification simultanée de la commande et de l'objet

Défilement

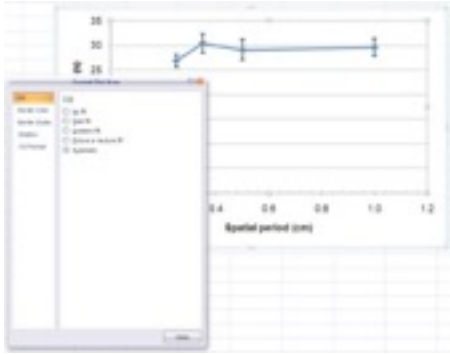
- Barres de défilement
  - Sens du défilement?
  - Découplage spatial
- Défilement direct
- Défilement automatique



Spécification d'arguments et de propriétés

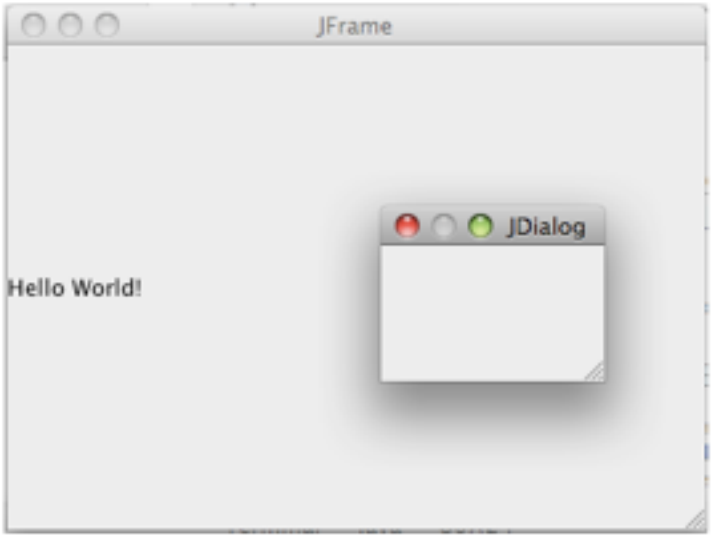


- Boites de dialogue
  - Découplage temporel et spatial entre la spécification de la commande, de ses paramètres et son exécution
  - Boites modales ou non modales
  - Parties optionnelles, boites à onglet
- Boites de propriétés
  - Effet immédiat des modifications sur les objets de la sélection



Fenêtre modale

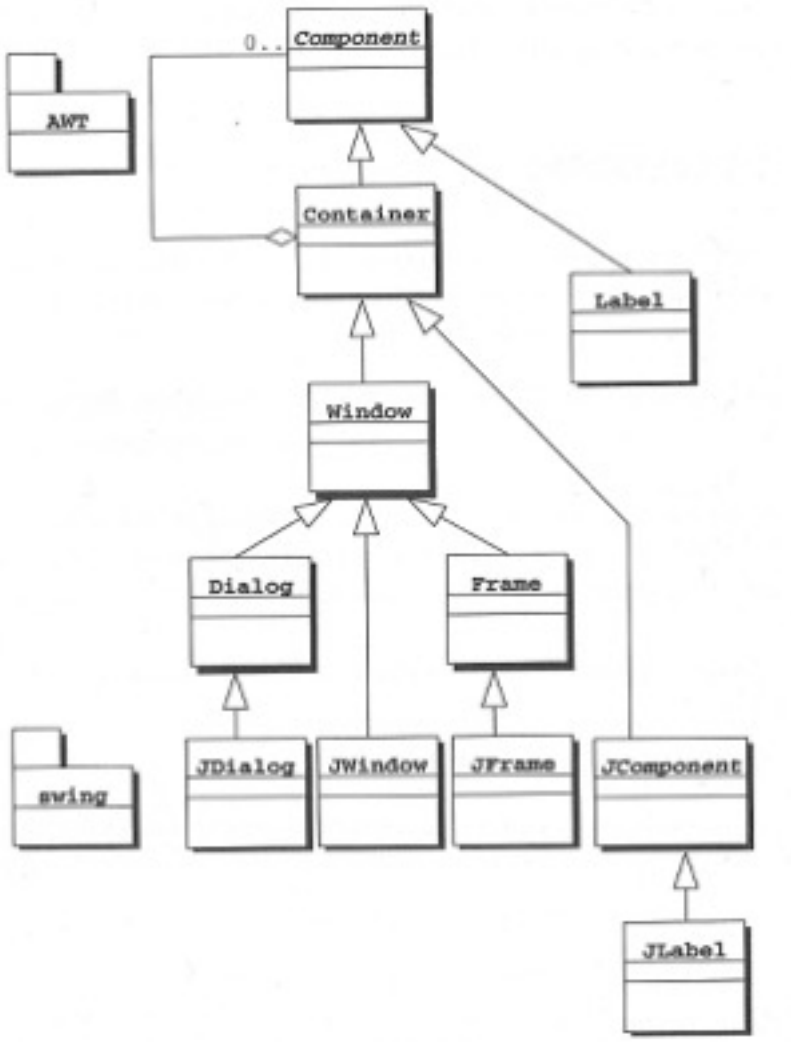
- Une fenêtre B est modale par rapport à une fenêtre A si l'affichage de B empêche l'accès – et non la vue – à la fenêtre A.
- Le constructeur de JDialog permet de préciser si une fenêtre est modale



Composants et conteneurs

- Les composants doivent être placés dans des conteneurs
- Les conteneurs servent à créer une hiérarchie de composants
- Nœuds = conteneurs, feuilles = composants

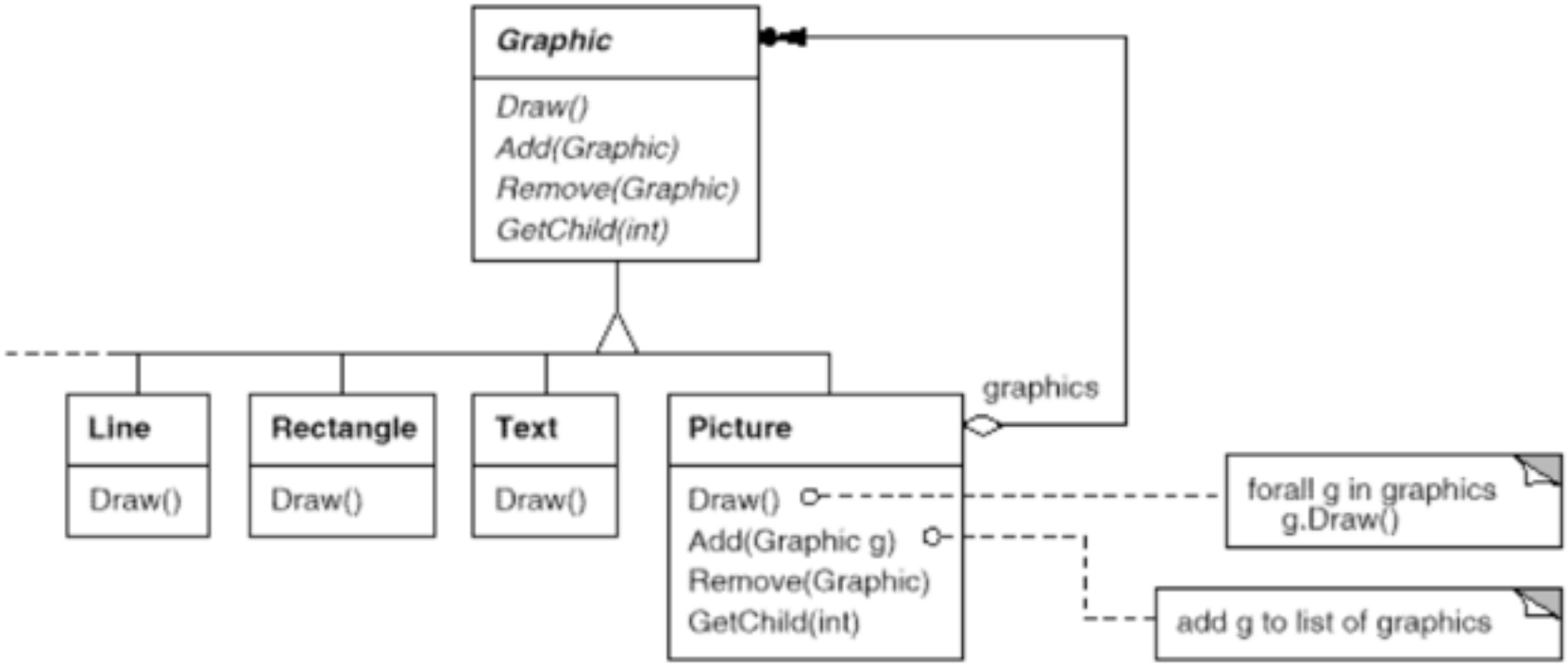
Conteneurs (Container)



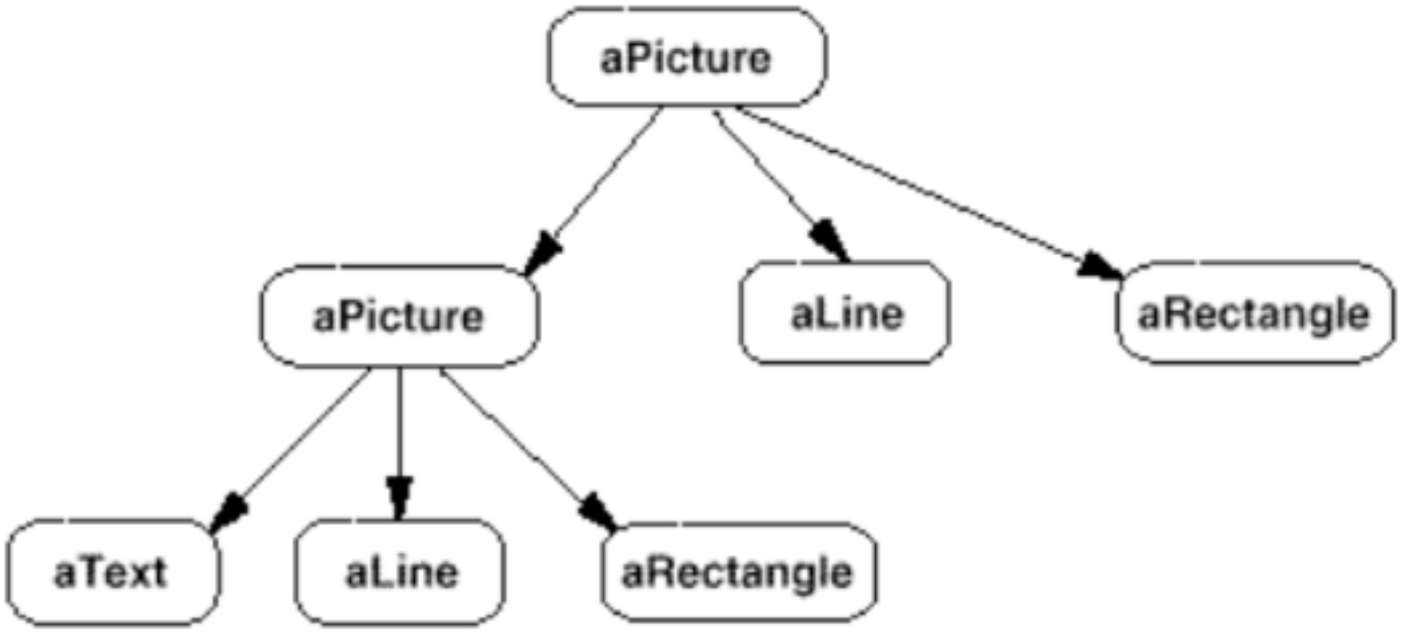
Container

- Container est un composant particulier dont le rôle est de contenir d'autres composants (il hérite de Component)
- Contient le nombre et la liste des composants
- Méthode add pour ajouter un composant
- Parmi l'ensemble des composants que contient un Container peut se trouver une instance de Container
- On crée ainsi une hiérarchie de composants (cela correspond au patron de conception *composite* qui permet de traiter tous les composants de la même façon)

Le patron de conception composite



Le patron de conception composite

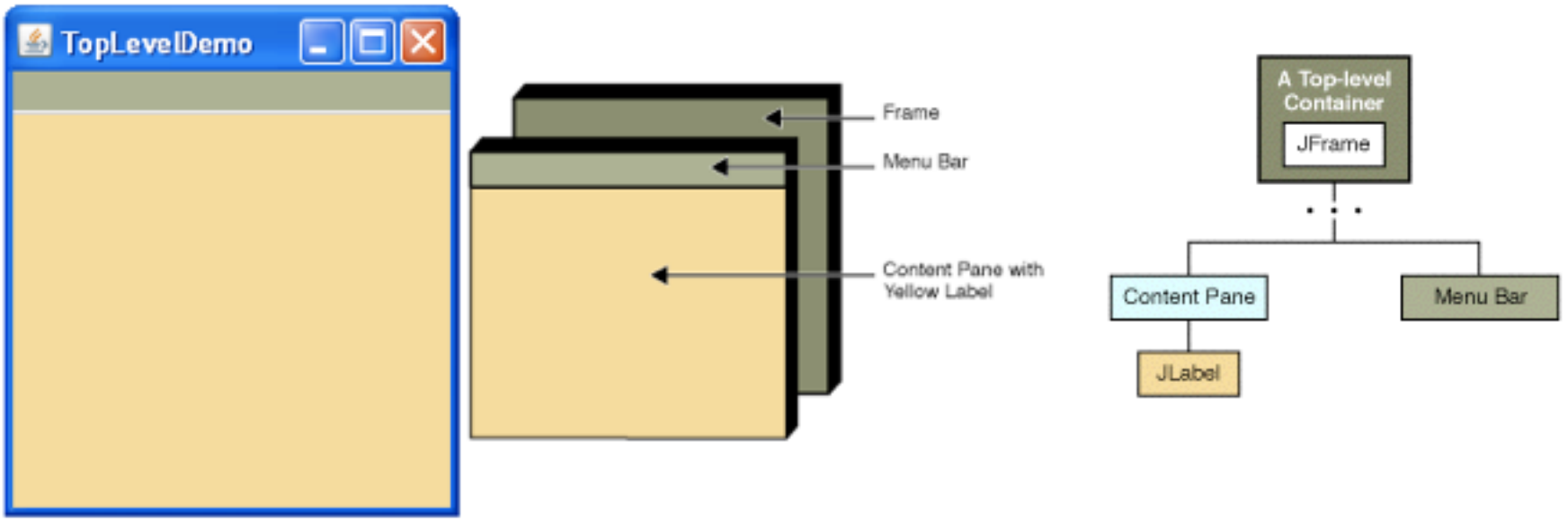


Hiérarchie de composants

- Conteneurs de haut niveau: JWindow, JFrame, JDialog et JApplet.
- Chaque application utilise au moins un conteneur de haut-niveau

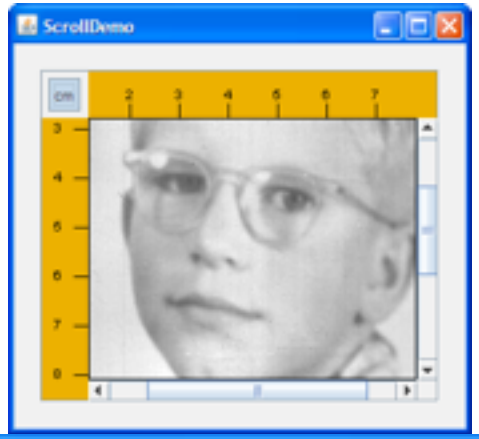
Hiérarchie de composants

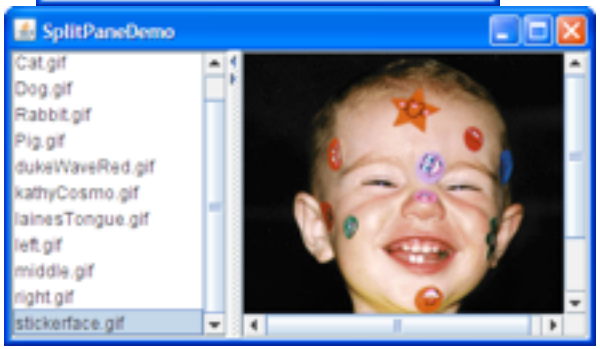
- Les conteneurs de haut niveau
- Utilisation de la méthode getContentPane accéder au container

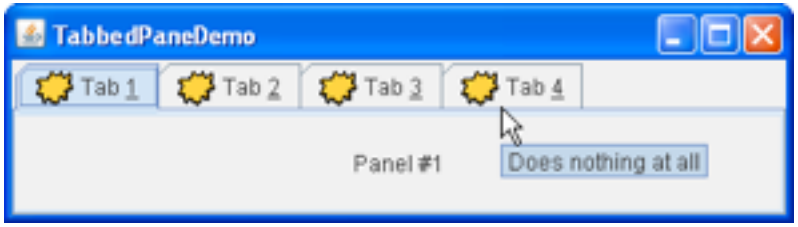


Conteneurs de niveaux intermédiaires

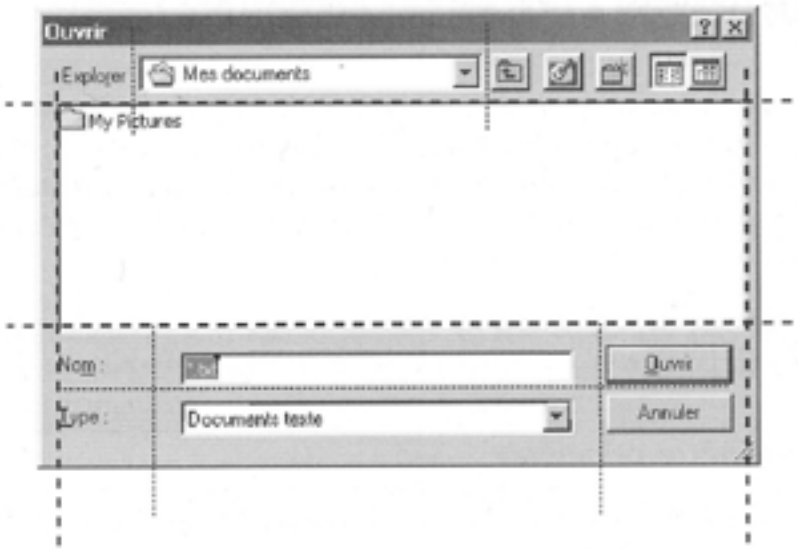
- JPanel
- JScrollPane
- JSplitPane
- JTabbedPane

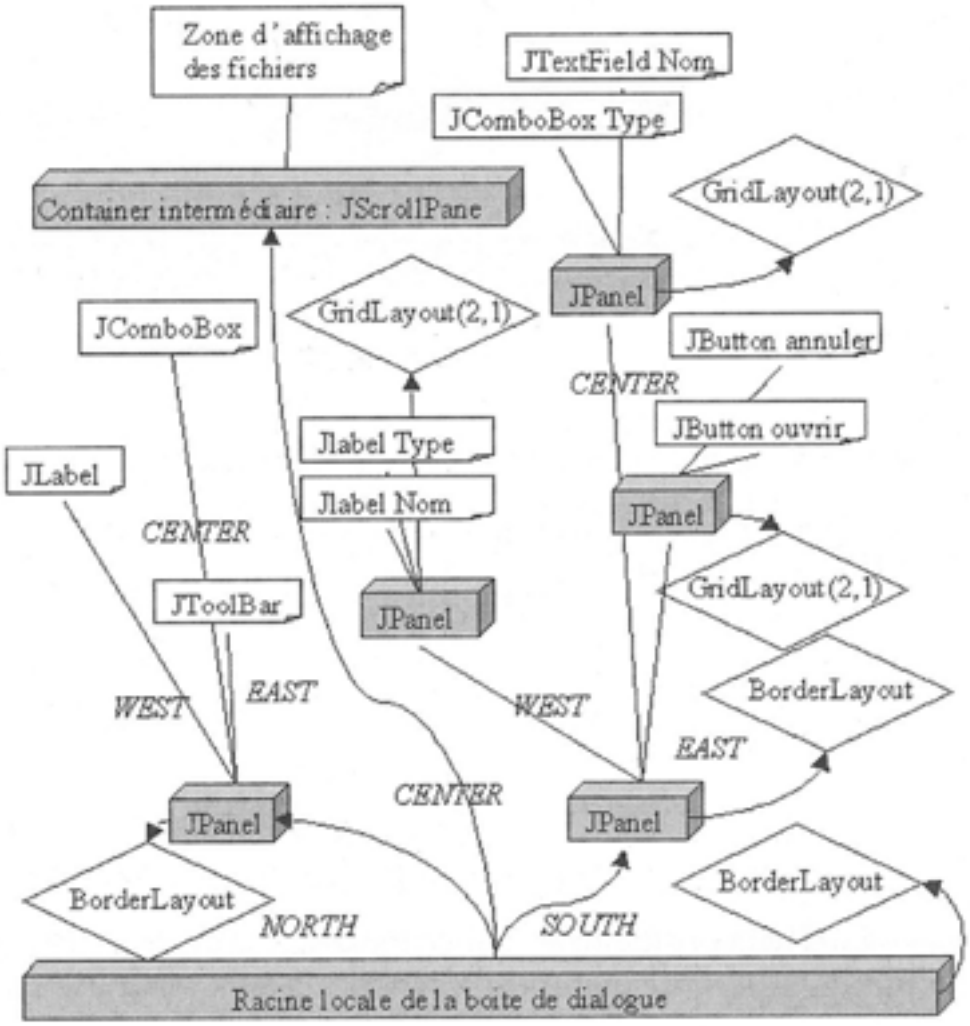






Exemple de hiérarchie de composants





Gestion du placement des composants

- Comment positionner les composants les uns par rapport aux autres?
- Comment un container agence-t-il visuellement les composants qu'il contient?
- Que se passe-t-il quand on agrandit la fenêtre? Les composants doivent-ils s'agrandir? Ajoute-t-on de l'espace? Où?
- Utilisation d'un gestionnaire de placement (layout)
  - Chaque layout définit une stratégie de positionnement

Gestionnaires de placement

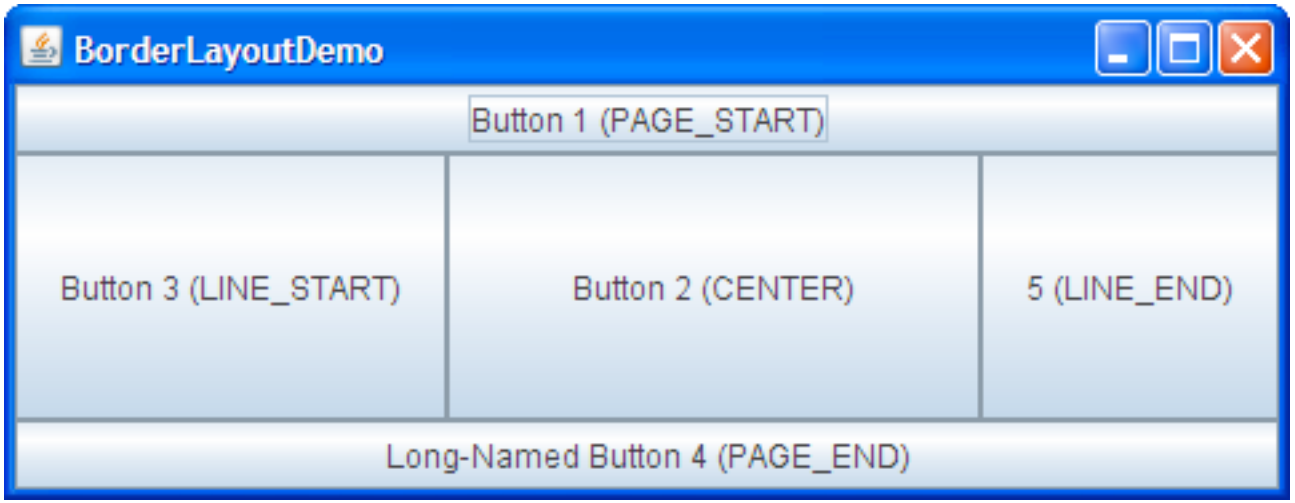
- Les différentes stratégies de positionnement doivent prendre en compte la position et la taille des composants qui lui sont confiés
- Chaque composant a deux tailles:
  - La taille effective (size)
  - La taille idéale (preferredSize)
- Un layout va essayer de rendre la taille réelle la plus proche possible de la taille préférée compte tenu des contraintes dues à sa stratégie et dues à la taille réelle du conteneur

Gestionnaires de placement

- Utilisation de la méthode setLayout (définie dans la classe Container) pour définir la stratégie de placement d'un Container
- fenetre.getContentPane().setLayout(new FlowLayout(FlowLayout.CENTER));

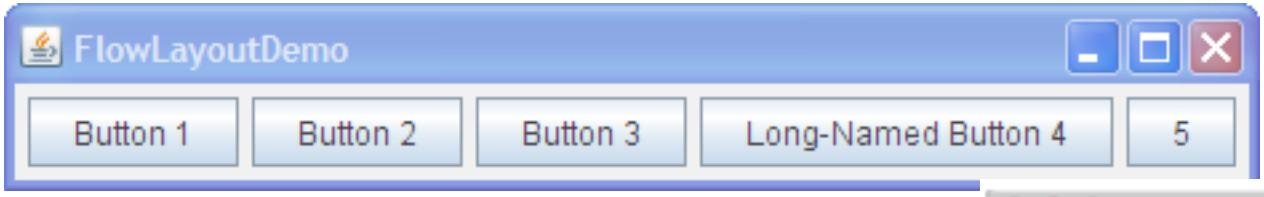
BorderLayout

- Layout par défaut des conteneurs de haut-niveau
- Le composant au centre essaie d'occuper le maximum d'espace disponible



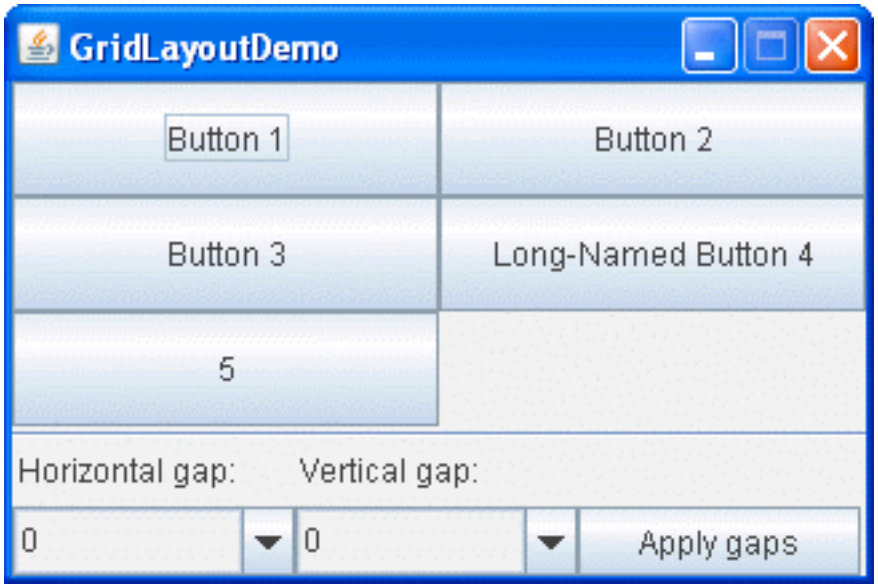
FlowLayout

- Conteneur par défaut des JPanel
- Place les éléments graphiques les uns à côté des autres



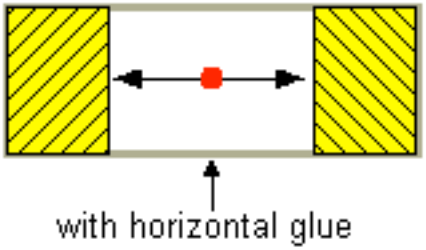
GridLayout

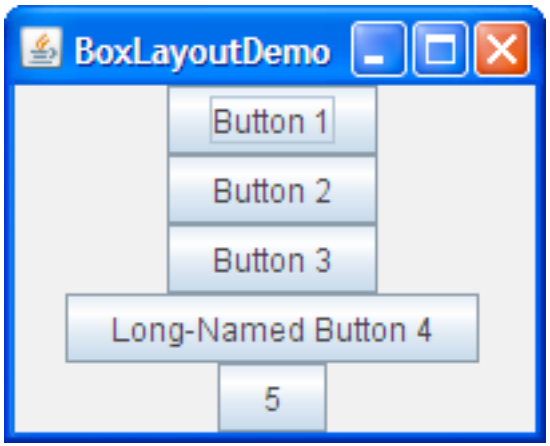
- L'espace du container est découpé en une grille
- Les composants sont égaux en taille

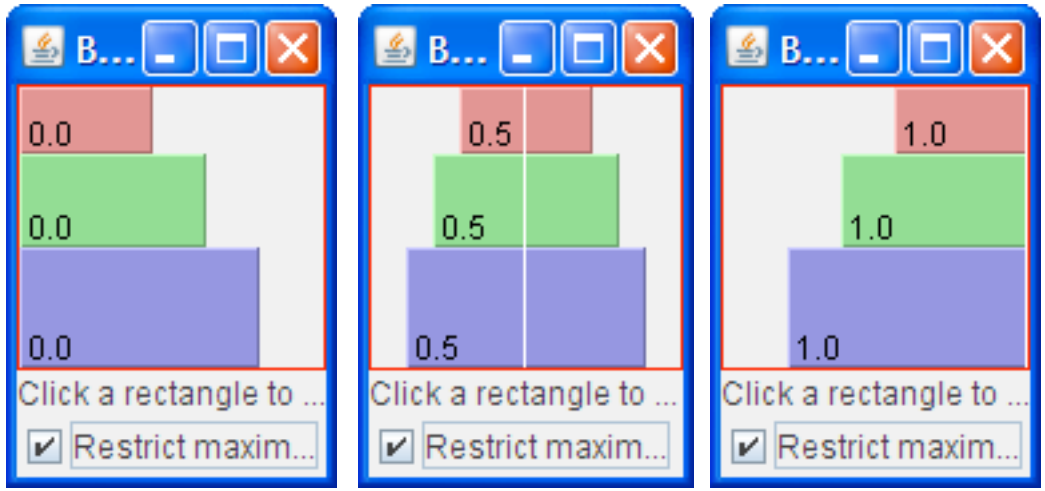


BoxLayout

- Alignement des composants suivant une seule ligne ou colonne
- Possibilité d'aligner les composants
- Possibilité d'ajouter des Filler

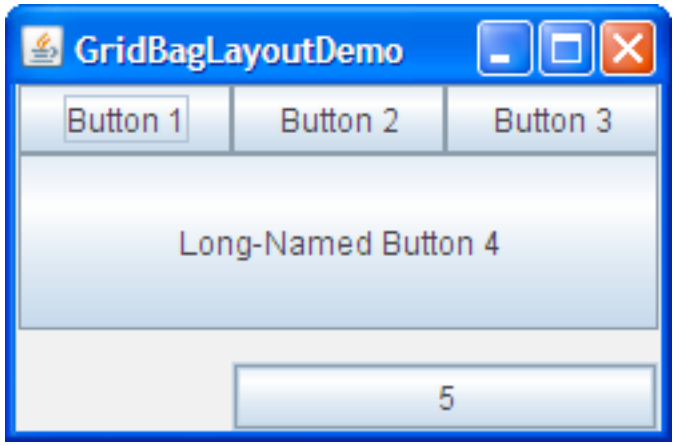


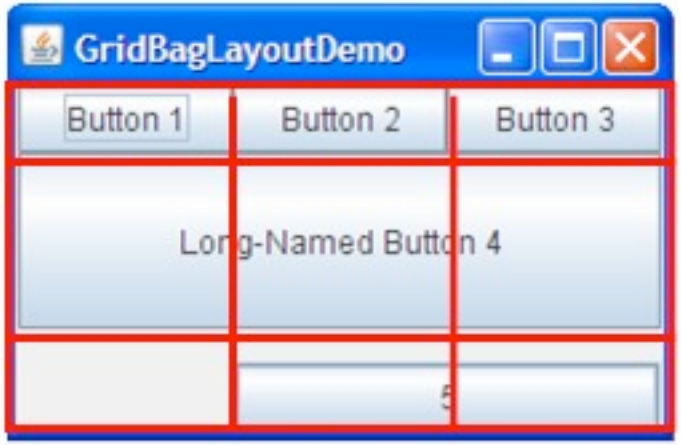




GridbagLayout

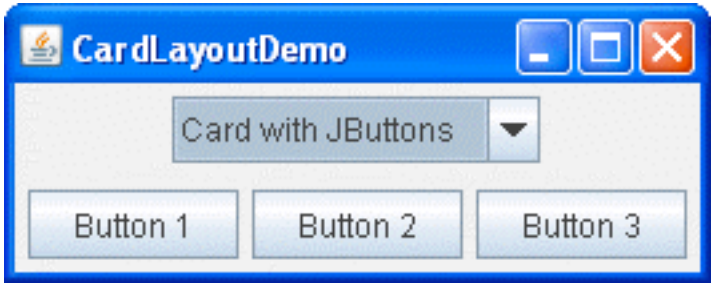
- Utilisation d'une grille
- Un composant peut occuper plusieurs cellules
- Toutes les lignes ne sont pas forcément de même hauteur (idem colonnes)

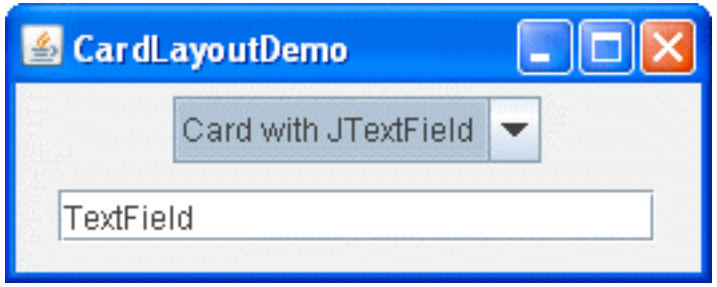




CardLayout

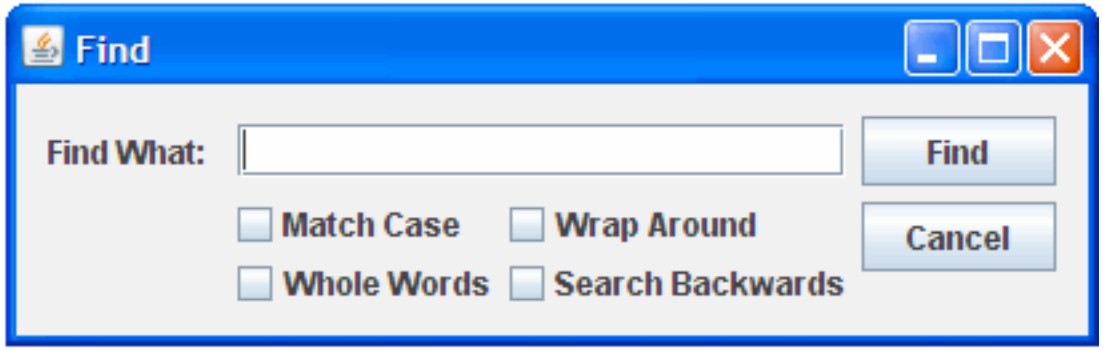
- Affichage de composants différents à des moments différents





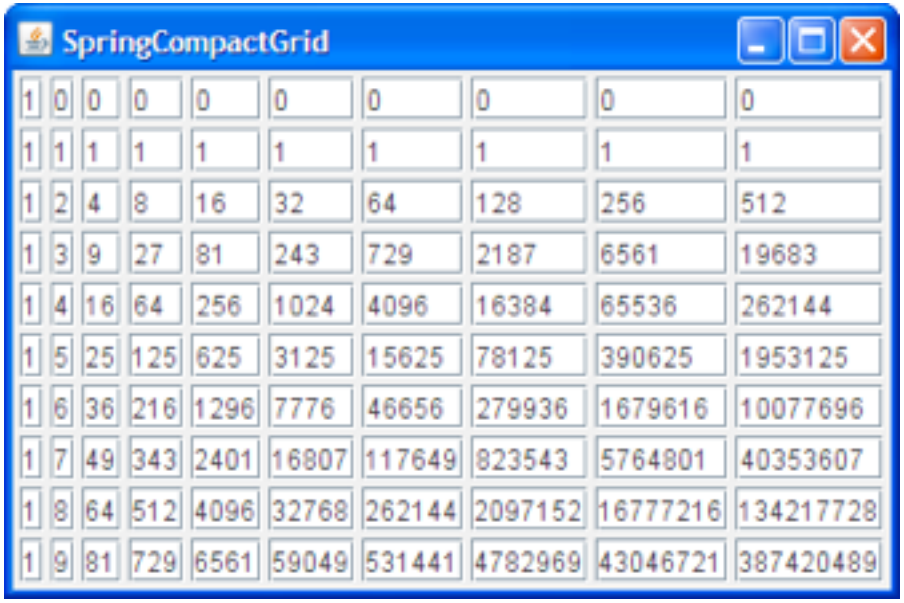
GroupLayout

- Utilisé par les générateurs d'interfaces
- Spécification des contraintes horizontales et verticales séparément



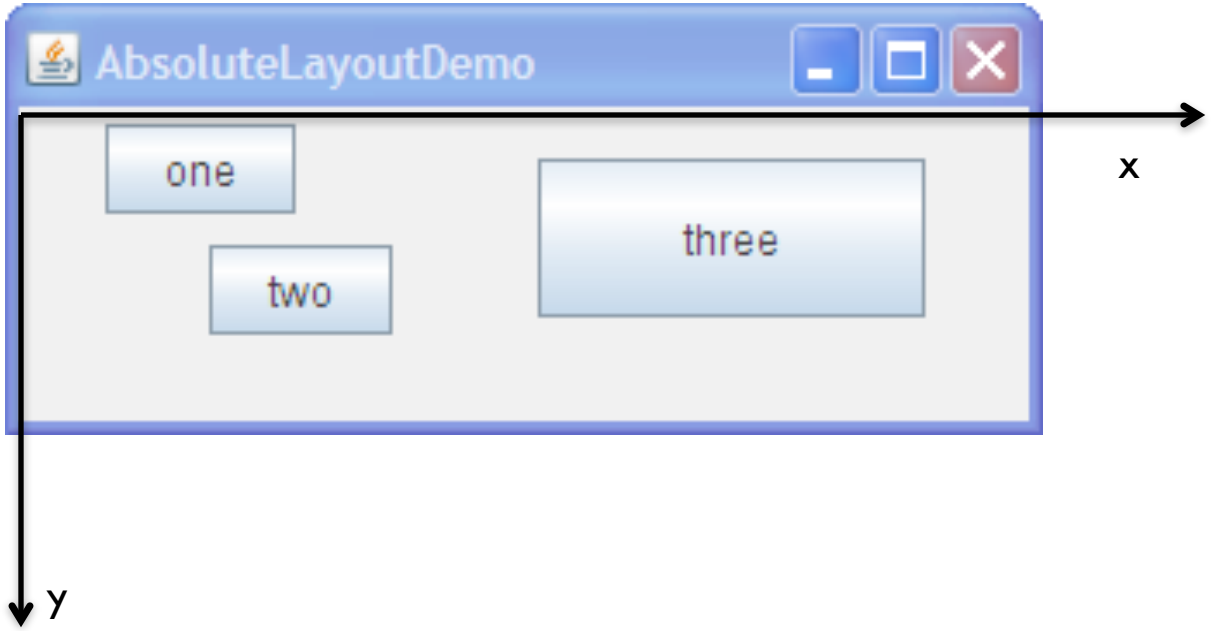
SpringLayout

- Utilisé par les générateurs d'interfaces
- Déconseillé de l'utiliser manuellement
- Définition de contraintes suivant les bords des composants



Positionnement absolu

- Définition de la taille et de la position de chaque composant
- L'origine est le coin supérieur gauche du conteneur



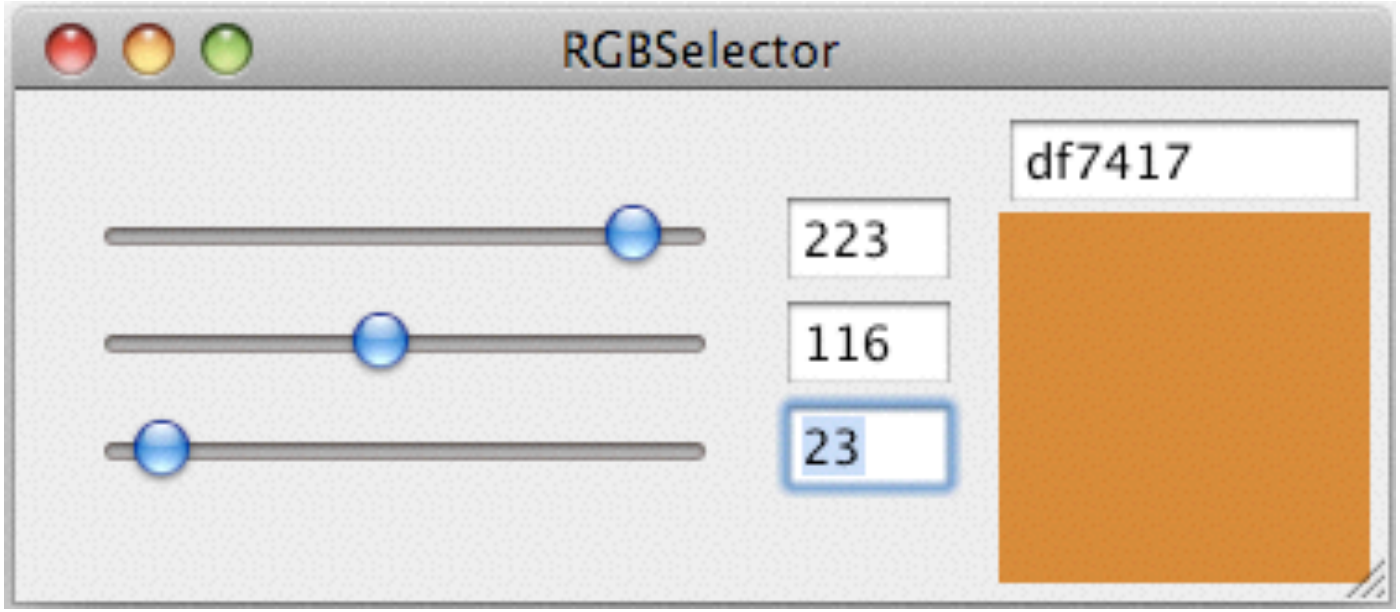
Créer son propre Layout

- Création d'une classe qui implémente l'interface LayoutManager
- Le conteneur s'adresse à son LayoutManager à des moments clés pour positionner ses composants
- Méthodes à implémenter:
  - Dimension preferredLayoutSize(Container)
  - Dimension minimumLayoutSize(Container)
  - void layoutContainer(Container)
    - Positionne et dimensionne chacun des composants du layout suivant la stratégie choisie (setSize, setLocation, setBounds)

Créer son propre Layout

- La méthode void layoutContainer(Container)
  - 1) récupérer la liste des composants du container: Component[] composants = parent.getComponents();
  - 2) pour chaque composant, définir sa taille (setSize), et sa position (setLocation) ou utiliser setBounds pour définir la position et la taille
  - 3) Ne pas tenir compte des composants invisibles

Exercice

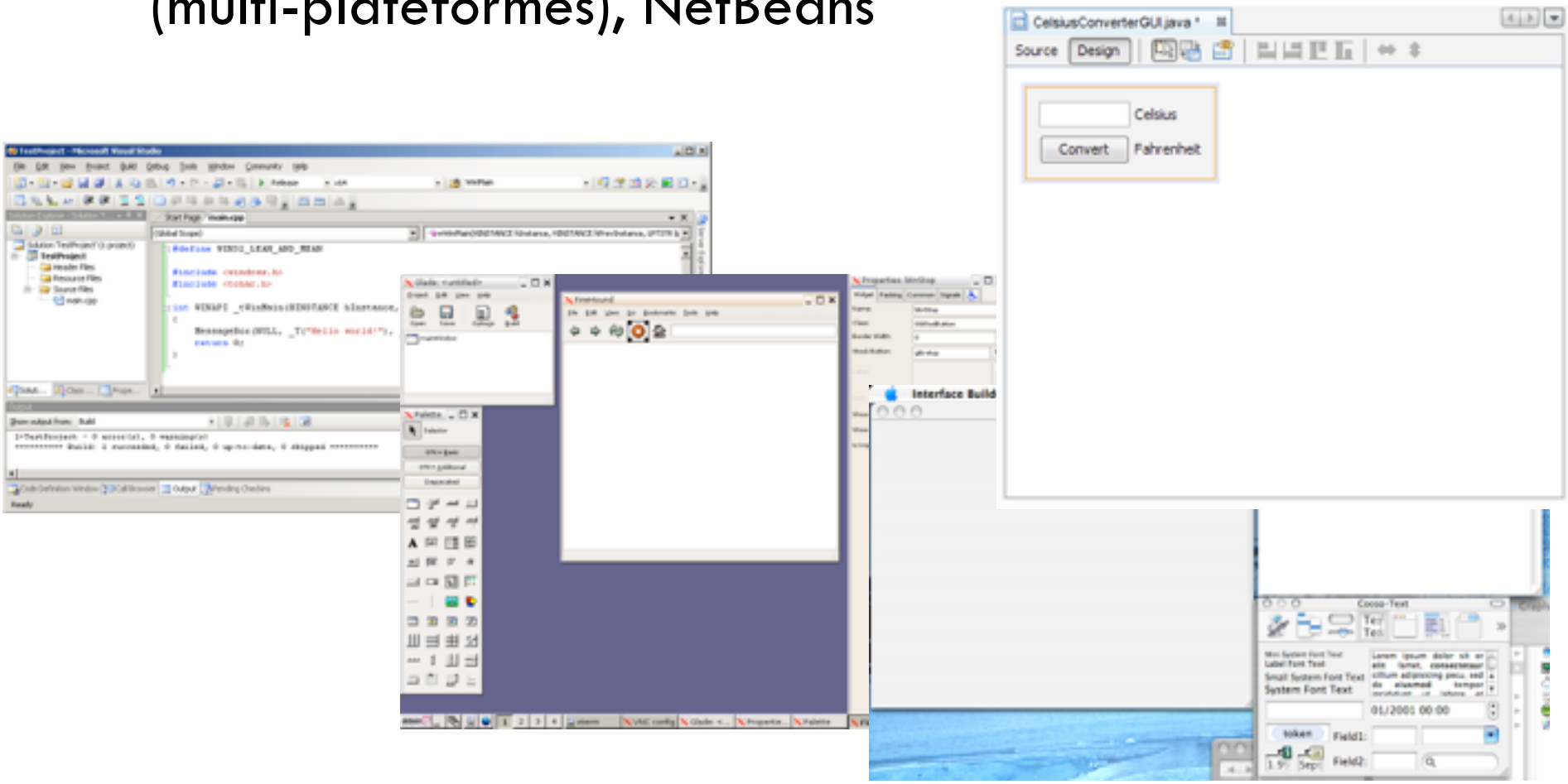


Générateurs d'interfaces

- But: aider la mise en œuvre d'interfaces
- De nombreux noms
  - User Interface Management Systems (UIMS)
  - User interface builder
  - User interface development environment
- Principe
  - Placer les widgets, modifier leurs attributs (couleur, etc …)
  - Les connecter à l'application
  - Tester leur comportement

Générateurs d'interfaces

- Ex: Visual * (Windows), interface builder (Mac), glade (multi-plateformes), NetBeans



Générateurs d'interfaces

- Inconvénients: What You See Is All You Get
  - Solutions partielles
  - Le code reste à écrire
  - Difficile de modifier le code généré

Pourquoi utiliser une boite à outils?

- Pour optimiser les paramètres suivants:
  - Temps de construction
  - Efficacité
  - Robustesse
  - Confort d'utilisation