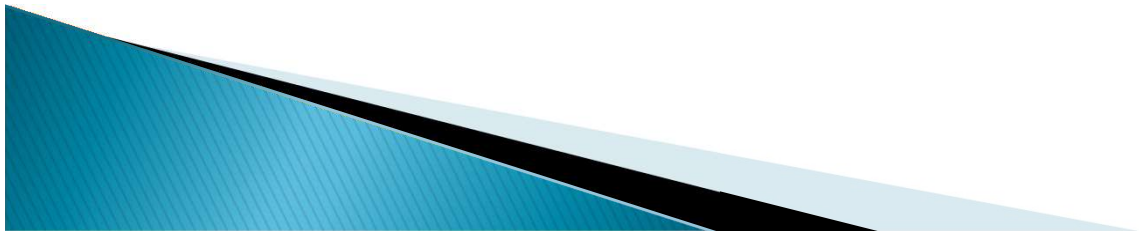
JAVA
Boîte à Outils: Java Swing

Boîtes à outils graphiques interactives

- Java est un langage Multi-Plateformes
- Java propose deux boîtes à outils graphiques standards :
  - AWT (Abstract Window Toolkit), simple, petite et limitée
  - SWING, cohérente, grosse et extensible.

- SWING facilite la transition à partir d'AWT



Conteneurs et composants

- Une interface graphique en Java est un assemblage de conteneurs (*Container*) et composants (*Component*).

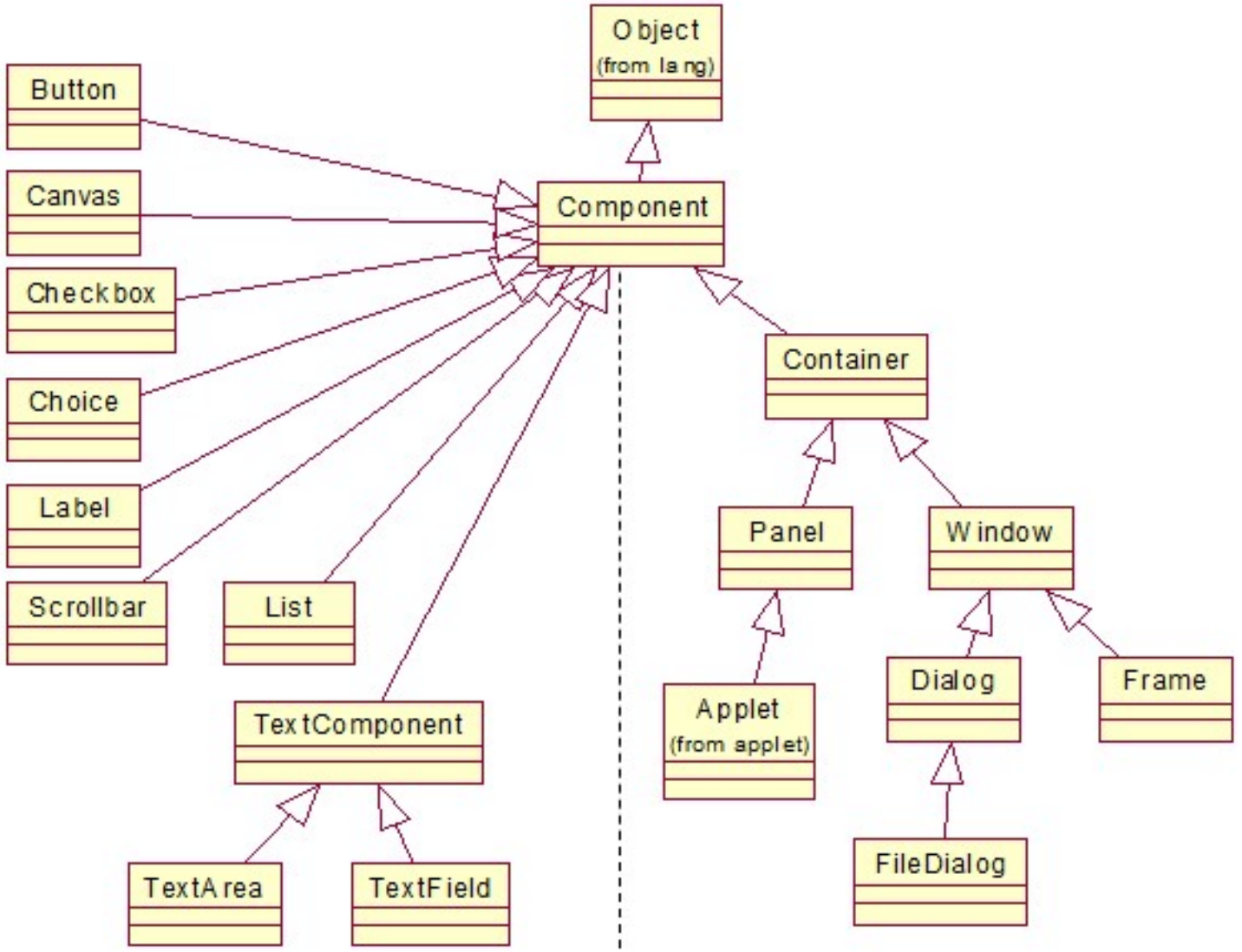
- Composant: partie "visible" de l'interface utilisateur Java.
  - Sous-classe de la classe abstraite **java.awt.Component.**
  - Exemple : **boutons, zones de textes , labels**, etc.

- Conteneur: espace destiné à accueillir plusieurs composants.
  - Sous-classe de la classe **java.awt.Container**, elle-même sous-classe de **Component**
  - Exemple: **Frames, applets, panels**, etc.

- «Layout Managers»: Gèrent la disposition des composants au sein d'un conteneur (BorderLayout, FlowLayout, GridLayout,..)

Conteneurs et composants

Java: Hiérarchie d'héritage des principaux composants de l'interface



Conteneurs et composants

- Conteneurs les plus courants: **Frame** et **Panel**.

- **Frame**: fenêtre de haut niveau avec un titre, une bordure et des angles de redimensionnement.
  - La plupart des applications utilisent au moins un Frame comme point de départ de leur interface graphique.

- **Panel**: n'a pas une apparence propre et ne peut pas être utilisé comme fenêtre autonome.
  - Les **Panels** sont créés et ajoutés aux autres conteneurs de la même façon que les composants tels que les boutons

Conteneurs et composants

- On ajoute un composant dans un conteneur, avec la méthode **add**() :

```
Panel p = new Panel();
Button b = new Button();
p.add(b);
```

- On retire un composant de son conteneur par **remove**

```
p.remove(b);
```

- Un composant a (notamment) :
  - une taille préférée obtenue avec **getPreferredSize**()
  - une taille minimum: **getMinimunSize**()
  - une taille maximum: **getMaximunSize**()

Conteneurs et composants


```
import java.awt.*;

public class EssaiFenetre1
{
  public static void main(String[] args)
  {
    Frame f =new Frame("Ma 1ère fenêtre");
    Button b= new Button("coucou");
    f.add(b);
    f.pack();
    f.show();
  }
}
```

Création d'une fenêtre (un objet de la classe Frame) avec un titre

Création du bouton ayant pour label « coucou »

Ajout du bouton dans la fenêtre

pack() → choisir la taille minimum de la fenêtre
show() → se rendre visible

Gestionnaire de présentation

- **Gestionnaire de Présentation** ou **Layout Manager**: associé à chaque conteneur
- Gère le positionnement et le (re)dimensionnement des composants d'un conteneur.
- Gestionnaires de présentation de AWT: **FlowLayout**, **BorderLayout, GridLayout, CardLayout, GridBagLayout**

Gestionnaire de présentation

- **setLayout()** permet de changer de Layout Manager

- Gestionnaires de présentation par défaut:
  - **BorderLayout** pour **Window** et ses descendants (**Frame**, **Dialog**, …)
  - **FlowLayout** pour **Panel** et ses descendants (**Applet**, ..)

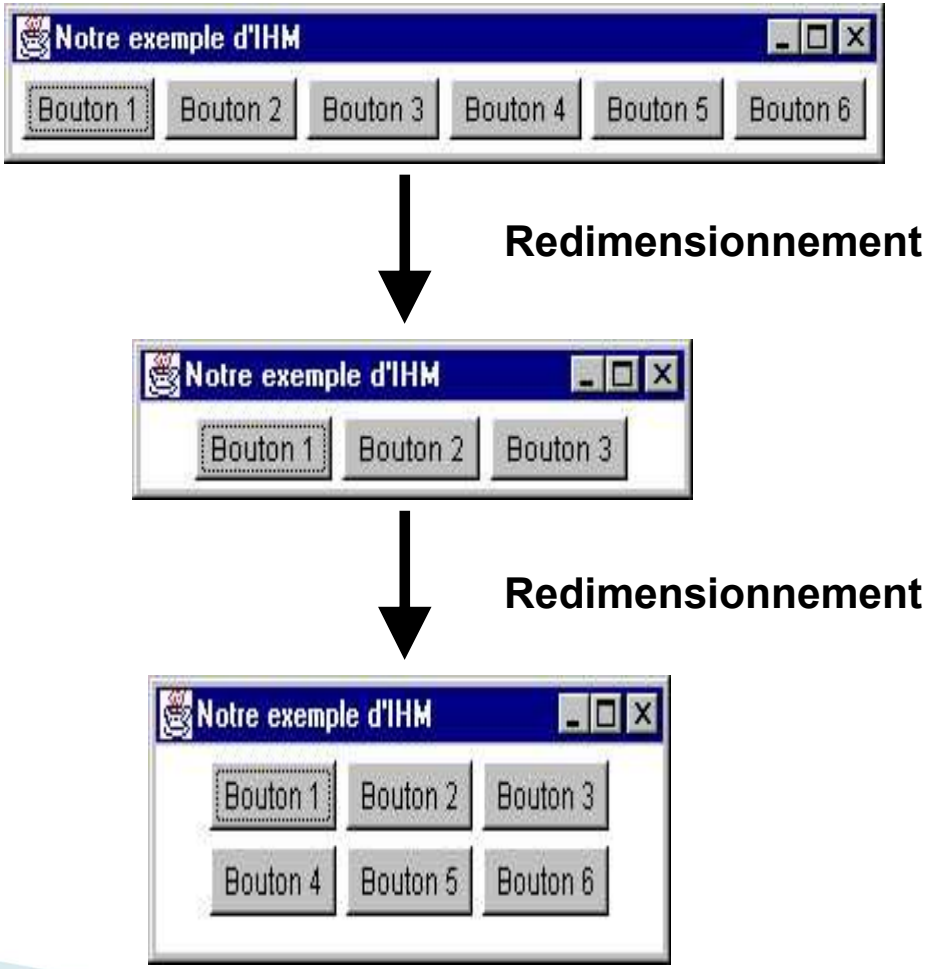
FlowLayout

- Utilisé par défaut dans les **Panel** si aucun **LayoutManager** n'est spécifié.
- Permet de disposer les composants de façon horizontale (ligne par ligne)
- Un FlowLayout peut spécifier :
  - une justification à gauche, à droite ou centrée,
  - un espacement horizontal ou vertical entre deux composants.
  - Par défaut, les composants sont centrés à l'intérieur de la zone qui leur est allouée.

FlowLayout

- Stratégie de disposition :
  - Respecter la **taille préférée** de tous les composants.
  - Disposer autant de composants qu'on peut faire tenir horizontalement à l'intérieur de l'objet **Container**
  - Commencer une **nouvelle ligne** de composants si on ne peut pas les faire tenir sur une seule ligne.
  - Si tous les composants ne peuvent pas tenir dans l'objet **Container**, ce n'est pas géré (c'est-à-dire que les composants **peuvent ne pas apparaître**).

FlowLayout


Notre exemple d'IHM
Bouton 1
Bouton 2
Bouton 3
Bouton 4
Bouton 5
Bouton 6
Redimensionnement
Notre exemple d'IHM
Bouton 1
Bouton 2
Bouton 3
Redimensionnement
Notre exemple d'IHM
Bouton 1
Bouton 2
Bouton 3
Bouton 4
Bouton 5
Bouton 6

FlowLayout

- cache réellement et effectivement les composants qui ne rentrent pas dans le cadre.
- Il n'a d'intérêt que quand il y a peu de composants.
- L'équivalent **vertical** du **FlowLayout** n'existe pas

FlowLayout: exemple

```
import javax.swing.*;
import java.awt.*;

public class PremiersJButtons    {
public PremiersJButtons()
 { JFrame frame = new JFrame();
     frame.setTitle("My first window !");
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.getContentPane().setLayout(new FlowLayout());
   frame.getContentPane().add(new  JButton(" 1er JButton"));
   frame.getContentPane().add(new  JButton(" 2è JButton"));
   frame.getContentPane().add(new  JButton("3è JButton"));
   frame.setVisible(true);     frame.pack();    }

public static void main(String[] args)   { new PremiersJButtons(); } }
```

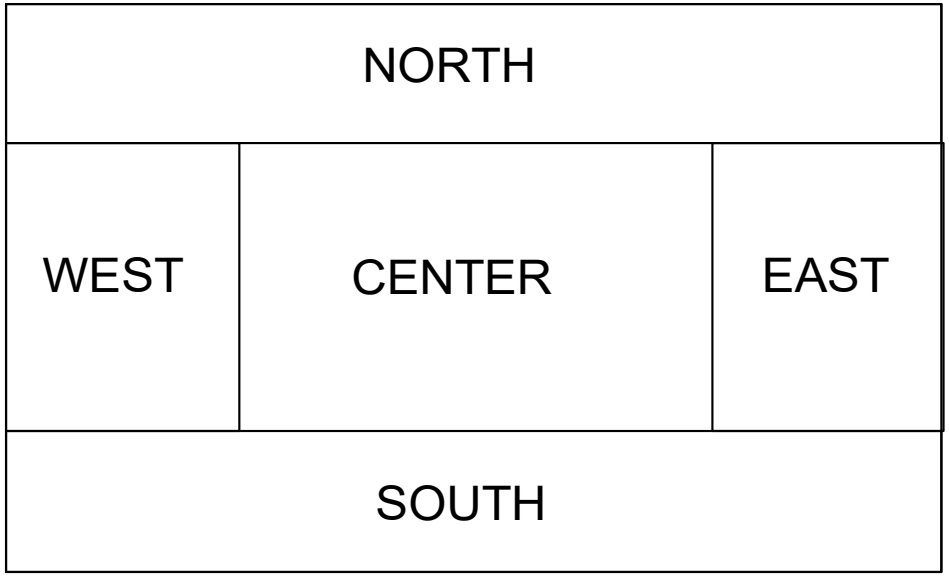
BorderLayout

- Divise son espace de travail en cinq zones géographiques : **North, South, East, West** et **Center**
- Les composants sont ajoutés par nom à ces zones (un seul composant par zone)
  - Exemple

    add("North", new Button("Le bouton nord !"));
  - Si une des zones de bordure ne contient rien, sa taille est 0

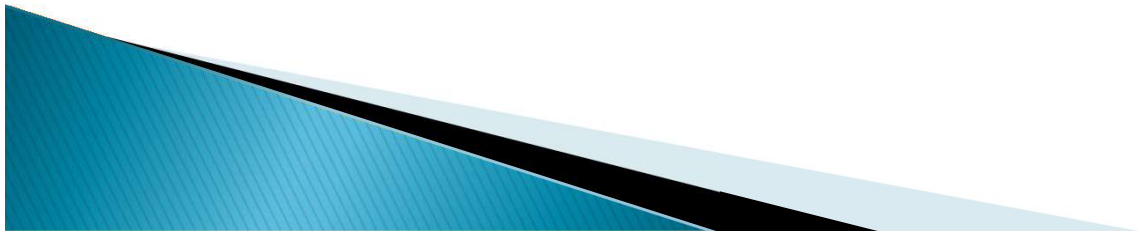
BorderLayout

- Division de l'espace avec le BorderLayout



BorderLayout: exemple

```
import javax.swing.*;
public class DisplayFrame {
   public static void main (String[] args) {
      JFrame f = new JFrame("FrameDemo");
      f.setLayout(new BorderLayout());
       JButton button = new JButton("Bouton (A)");
       f.add(button, BorderLayout.PAGE_START);
      f.setSize(300,200);  //alternative: f.pack();
  f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.setVisible(true);       }
}
```

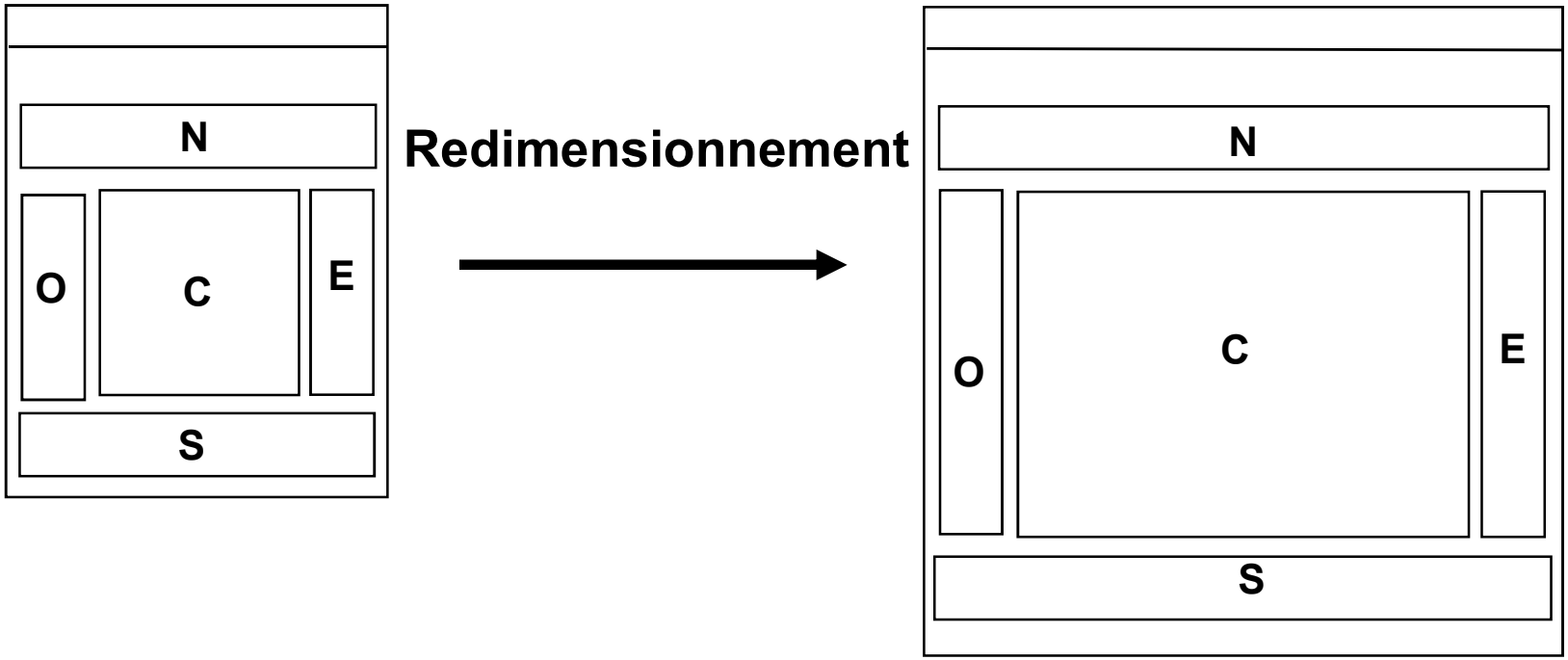


BorderLayout: stratégie de disposition

- Partie NORTH ou SOUTH → il respecte sa **hauteur préférée** si possible et fixe sa **largeur** à la totalité de la largeur disponible de l'objet Container (éventuellement **élargie** mais **pas allongée**)

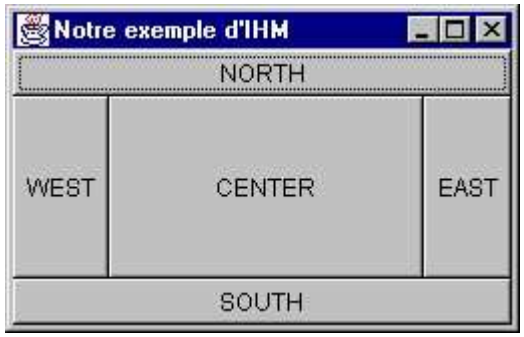
- Partie EAST ou WEST → il respecte sa **largeur préférée** si possible et fixe sa **hauteur** à la totalité de la hauteur disponible (**allongée** mais **pas élargie**)
- Partie CENTER → il lui donne la place qui reste, s'il reste encore (étirée dans les 2 sens)
- Le **composant** est aussi redimensionné en fonction de la taille de la zone

BorderLayout



BorderLayout (8)

Redimensionnement



Redimensionnement

GridLayout

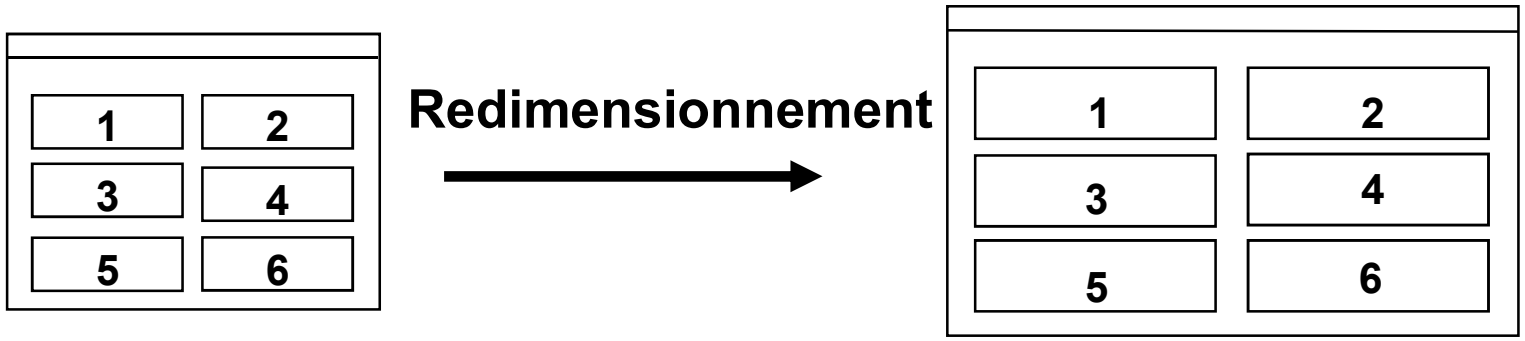
- Dispose les composants dans une **grille.**
  - Découpage de la zone d'affichage en lignes et co-lonnes définissant des cellules de dimensions égales
  - Chaque **composant a la même taille**
    - quand ils sont ajoutés dans les cellules le remplissage s'effectue de gauche à droite et de haut en bas.
  - Construction: **new GridLayout(3,2); 3 lignes et 2 colonnes**

GridLayout: exemple

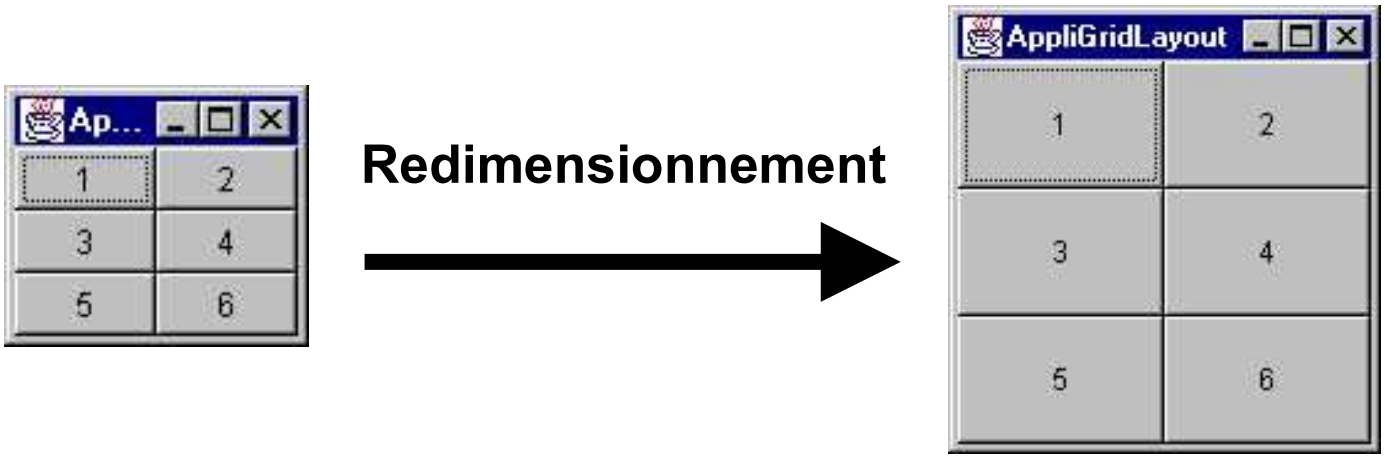
```
import java.awt.*;
public class AppliGridLayout extends Frame
{
public AppliGridLayout()
 {
  super("AppliGridLayout");
  this.setLayout(new GridLayout(3,2));
   for (int i = 1; i < 7; i++)
     add(new Button(Integer.toString(i)));
  this.pack();
  this.show();
 }

public static void main(String args[])
 {
  AppliGridLayout appli = new AppliGridLayout();
 }  }
```

GridLayout

- Lors d'un redimensionnement les composants changent tous de taille mais leurs positions relatives ne changent pas.



GridLayout



CardLayout

- **CardLayout** n'affiche qu'un composant à la fois:
  - les composants sont empilés, comme un tas de cartes
- **CardLayout** permet à plusieurs composants de partager le même espace d'affichage.
- Pour ajouter un composant avec CardLayout, on utilise **add(String cle, Component monComposant)**
- Le passage d'un composant à un autre se fait avec les méthodes **first, last, next, previous** ou **show**

GridBagLayout

- Fournit des fonctions de **présentation complexes** basées sur une **grille** dont les **lignes et les colonnes** sont de taille **variables**. Il permet:
  - à des composants simples de prendre leur taille préférée au sein d'une cellule, au lieu de remplir toute la cellule
  - l'extension d'un même composant sur plusieurs cellules
- **GridBagLayout** est compliqué à gérer
  - On évite généralement de l'utiliser en associant des **Container** avec différents **Layout Managers**

GridBagLayout (2)

- Associé à un objet GridBagConstraints
-
  - **GridBagConstraints** définit des contraintes de positionnement, d'alignements, de taille, etc. d'un composant dans un conteneur **GridBagLayout**
  - Un même objet **GridBagConstraints** peut-être associé à plusieurs composants.
  - **GridBagConstraints** est assez fastidieux à spécifier…

Exemple: plusieurs Layouts

```
super("AppliComplexeLayout");
setLayout(new BorderLayout());
Panel pnorth = new Panel();
pnorth.add(b1); pnorth.add(b2);
pnorth.add(b3); pnorth.add(b4);
this.add(pnorth,BorderLayout.NORTH);

Panel pcenter = new Panel();
pcenter.setLayout(new GridLayout(2,2));
pcenter.add(gr1); pcenter.add(gr2);
pcenter.add(gr3); pcenter.add(gr4);
this.add(pcenter,BorderLayout.CENTER);

Panel psouth = new Panel();
psouth.setLayout(new FlowLayout());
psouth.add(ch); psouth.add(tf);
this.add(psouth, BorderLayout.SOUTH);
```

D'autres gestionnaires?

- Un objet «container» peut ne pas avoir de gestionnaire en fixant le LayoutManager à null

  - **Frame f = new Frame(); f.setLayout(null);**
  - A la charge du programmeur de positionner chacun des composants « manuellement » en indiquant leur **position absolue** dans le repère de la fenêtre.
  - Ex: bouton1.setBounds (x, y, largeur, hauteur); → **disposition absolue**

Récapitulatif

- FlowLayout
  - Flux : composants placés les uns derrière les autres
- BorderLayout
  - Ecran découpé en 5 zones (« North », « West », « South », « East », « Center »)
- GridLayout
  - Grille : une case par composant, chaque case de la même taille
- CardLayout
  - « Onglets » : on affiche un élément à la fois
- GridBagLayout
  - Grille complexe : plusieurs cases par composant