Chapitre 3 : Gestionnaires de mise en forme (*Layout manager*)

RT2

Plan

- Premier composant : JButton
- Layout manager
- BorderLayout
- FlowLayout
- GridLayout
- GridBagLayout
- CardLayout
- BoxLayout

L'objet JButton

- Création

```
//méthode 1 : instanciation + le libellé

    JButton bouton = new JButton("Mon premier bouton");

//méthode 2 : instanciation + définition du libellé

     JButton bouton2 = new JButton();

     bouton2.setText("Mon deuxième bouton");
```

- Un bouton s'utilise avec la classe **JButton** du package javax.swing
- Pour ajouter un bouton dans une fenêtre, on utilise la méthode **add()** de son content pane.

Layout managers

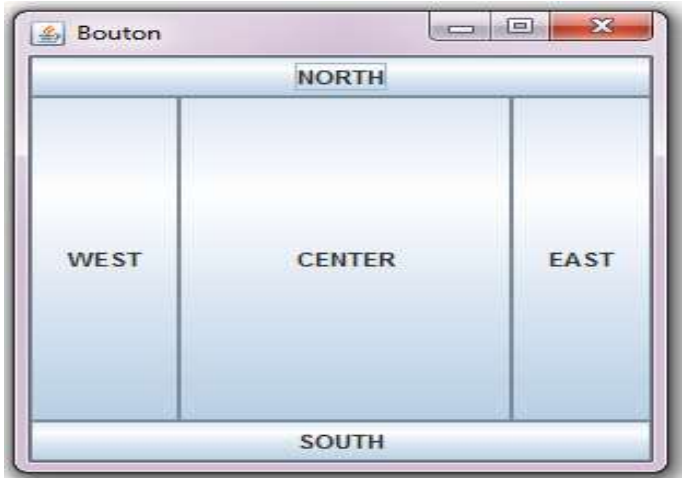
- Les layout managers se trouvent dans le package java.awt.
- Pour chaque **conteneur** (fenêtre, panneau, boîte de dialogue, etc.), Java permet de choisir un gestionnaire de mise en forme responsable de la disposition des composants.
- Les composants peuvent être placés dans des conteneurs tels des panneaux. Les conteneurs pouvant eux-mêmes être imbriqués dans d'autres, la classe Container étend Component.
- Pour définir un layout sur un conteneur, on appelle la méthode setLayout().

Layout manager personalisé

- Il est possible de créer son propre gestionnaire sous forme d'une classe implémentant l'interface **LayoutManager** qui doit comporter les cinq méthodes suivantes :
  - *addLayoutComponent*
  - *removeLayoutComponent*
  - *preferredLayoutSize*
  - *minimumLayoutSize*
  - *layoutContainer*

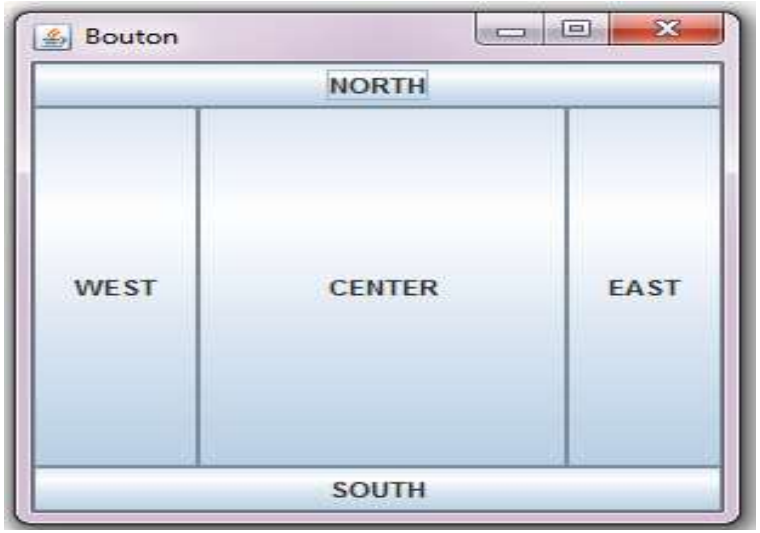
BorderLayout

```
public class Fenetre extends JFrame{
public Fenetre(){
this.setTitle("Bouton");
this.setSize(300, 300);
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
this.setLocationRelativeTo(null);
//On définit le layout sur le content pane
  this.setLayout(new BorderLayout());
//On ajoute le bouton au content pane de la Jframe
this.getContentPane().add(new JButton("CENTER"),BorderLayout.CENTER);//Centre
this.getContentPane().add(new JButton("NORTH"), BorderLayout.NORTH); // Nord
this.getContentPane().add(new JButton("SOUTH"), BorderLayout.SOUTH); // Sud
this.getContentPane().add(new JButton("WEST"), BorderLayout.WEST); // Ouest
this.getContentPane().add(new JButton("EAST"), BorderLayout.EAST); // Est
this.setVisible(true);
  }   }
```



BorderLayout

- C’est le gestionnaire par défaut du panneau de contenu contentPane de chaque JFrame
- Si aucune chaîne de position n’est mentionnée, la constante "Center" est utilisée.
- Tant qu’un bord n’est pas occupé par un composant, l’espace correspondant est utilisable par le composant central.

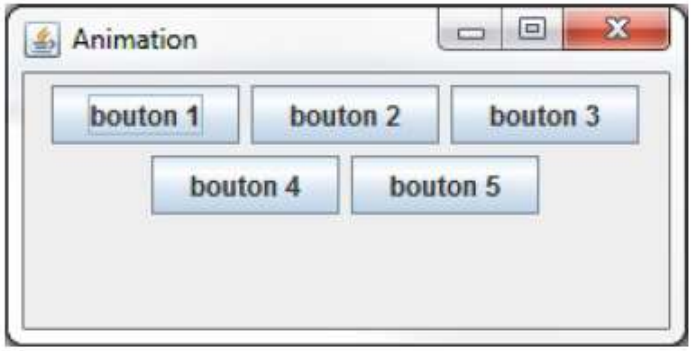


FlowLayout

- C'est le layout manager par défaut d'un objet JPanel.
- Ce layout centre les composants dans le conteneur.
- Il dispose les composants les uns à la suite des autres, sur une même ligne. Lorsqu'une ligne ne possède plus suffisamment de place, l'affichage se poursuit sur la ligne suivante..
- java.awt.FlowLayout

```
FlowLayout(int align)
FlowLayout(int align, int hgap, int vgap)
```

- Paramètres
  - align : constantes d'alignement LEFT, CENTER ou RIGHT.
  - hgap : intervalle horizontal en pixels à utiliser
  - vgap L'intervalle vertical en pixels à utiliser



FlowLayout

- Exemple

```
conteneur.setLayout(new FlowLayout(FlowLayout.CENTER)) ;
```

  - Les composants sont centrés sur les différentes lignes.
  - Ce choix est effectué une fois pour toutes à la construction : toutes les lignes de composants qui vont suivre vont garder le même alignement centré.

- Pour spécifier un intervalle entre les composants (Par défaut, il est de 5 pixels dans les deux directions), il faut indiquer le paramètre d'alignement :

```
conteneur.setLayout(new FlowLayout(FlowLayout.RIGHT, 10, 15)) ;
```

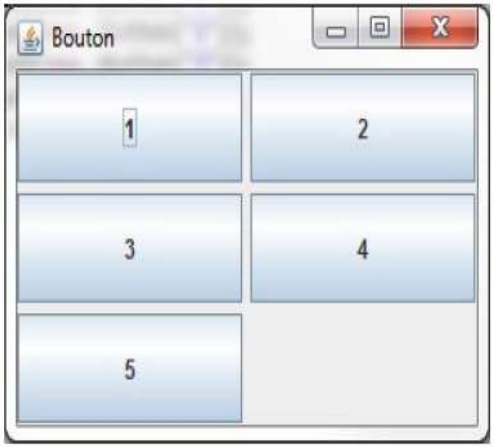
GridLayout

- C'est une grille définie par un nombre de lignes et de colonnes.
- Définition du nombre de lignes et de colonnes :

```
GridLayout g = new GridLayout();
g.setColumns(2);
g.setRows(3);
this.setLayout(g);
```

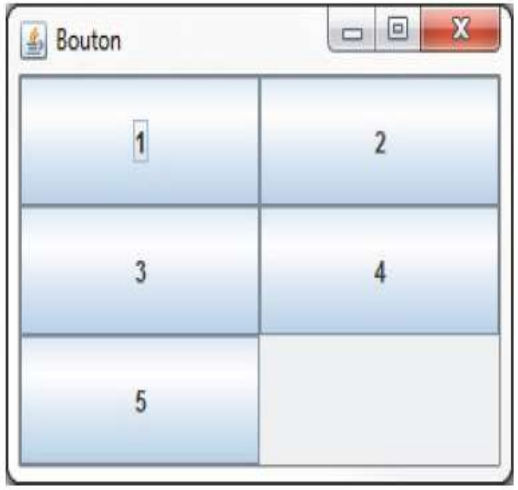
- Paramétrer l'espace entre les colonnes et les lignes :

```
GridLayout g = new GridLayout(3, 2);
//Cinq pixels d'espace entre les colonnes(H: Horizontal)
g.setHgap(5);
//Cinq pixels d'espace entre les lignes(V : Vertical)
g.setVgap(5);
//Ou en abrégé :
GridLayout g = new GridLayout(3, 2, 5, 5);
```



GridLayout

```
 public class Fenetre extends JFrame{
  public Fenetre(){
 this.setTitle("Bouton");this.setSize(300, 300);
 this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 this.setLocationRelativeTo(null);
//On définit le layout : Trois lignes sur deux colonnes
     this.setLayout(new GridLayout(3, 2));
//On ajoute le bouton au content pane
   this.getContentPane().add(new JButton("1"));
   this.getContentPane().add(new JButton("2"));
   this.getContentPane().add(new JButton("3"));
   this.getContentPane().add(new JButton("4"));
   this.getContentPane().add(new JButton("5"));
   this.setVisible(true);   }       }
```



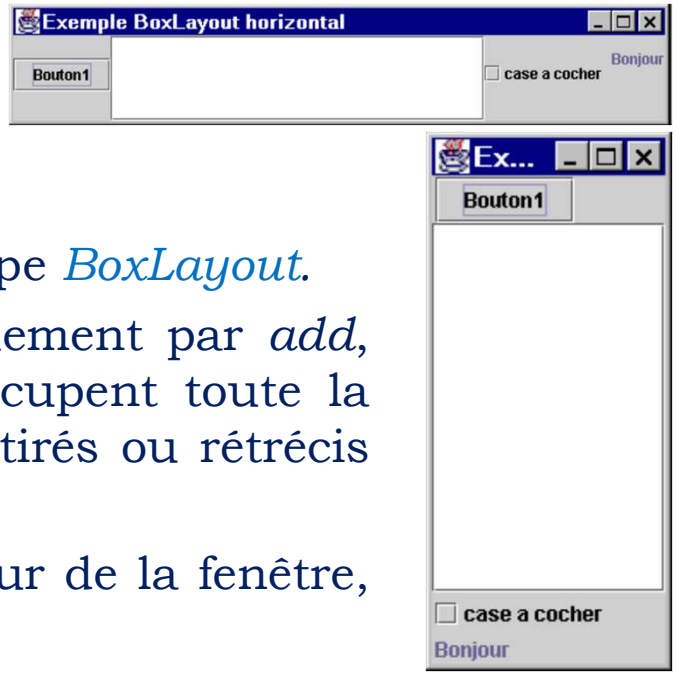
BoxLayout

- Ce gestionnaire permet de disposer des composants suivant une seule ligne ou une seule colonne.
- Pour créer un box horizontal :

```
Box ligne = Box.createHorizontalBox();
```

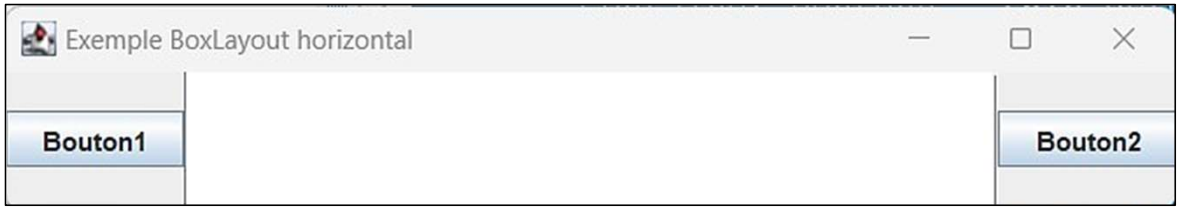
- Pour créer un box vertical :

```
Box ligne = Box.createVerticalBox();
```



- Un tel conteneur est doté par défaut d'un gestionnaire de type *BoxLayout*.
- Dans un box horizontal, les composants, ajoutés classiquement par *add*, sont alignés de gauche à droite ; ils sont contigus et occupent toute la largeur et toute la hauteur du conteneur. Ainsi, ils sont étirés ou rétrécis dans la mesure du possible.
- Si tous les composants ne peuvent pas tenir dans la largeur de la fenêtre, certains ne seront pas visibles.

BoxLayout : Exemple




```
class MaFenetre extends JFrame
{ private Box bHor ;
  private JButton b1, b2 ;
  private JTextField txt ;
  public MaFenetre ()
  {  this.setTitle ("Exemple BoxLayout horizontal") ;
     this.setSize (550, 100) ;
     Container contenu = this.getContentPane() ;
     bHor = Box.createHorizontalBox() ;   contenu.add(bHor) ;
     b1 = new JButton ("Bouton1") ;  bHor.add (b1) ;
     txt = new JTextField (20) ; bHor.add (txt) ;
     b2 = new JButton ("Bouton2") ; bHor.add (b2) ;
   }
}
```

Modifier l'espacement avec strut et glue

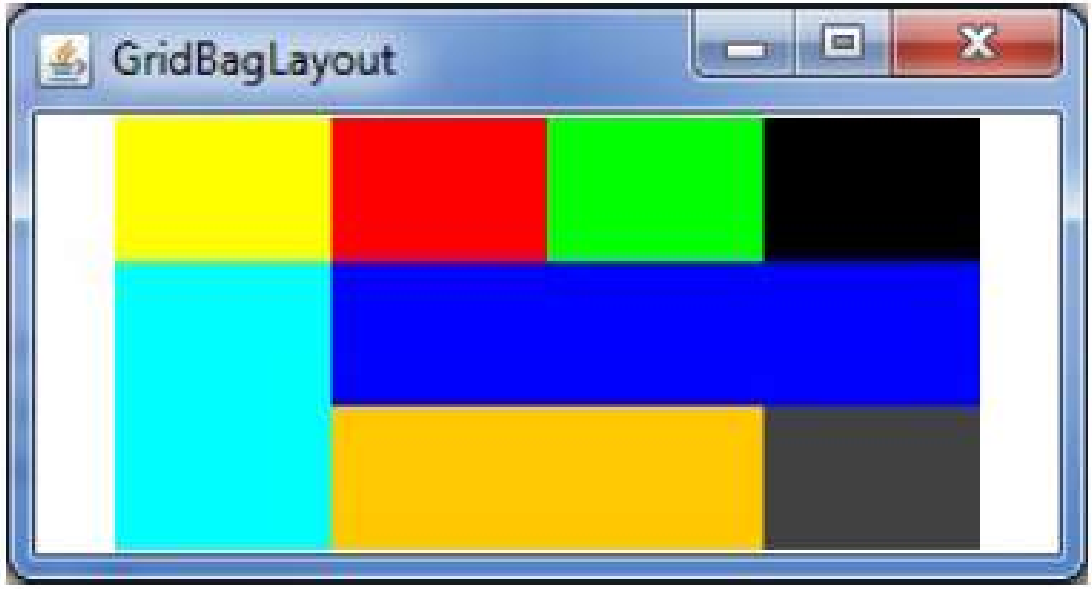
- Pour fixer des espaces précis entre certains composants, on peut créer des composants virtuels de taille fixe.
- Pour un box vertical, on peut créer un composant virtuel de hauteur donnée, à l'aide de la méthode statique *createVerticalStrut* de la classe *Box* :
- Pour un box horizontal, on dispose d'une méthode similaire : *createHorizontalStrut*
- Exemple : `Box.createVerticalStrut(10)`
- Dans cet exemple, quoi qu'il arrive, il subsistera toujours un espace de 10 pixels entre les composants situés de part et d'autre de ce composant virtuel
- La méthode statique *createGlue()* crée un emplacement virtuel de taille entièrement ajustable. Celle-ci est déterminée par le gestionnaire de façon à espacer au maximum les composants situés de part et d'autre.

strut et glue : Exemple

```
class MaFenetre extends JFrame
{ private Box bVert ;
  private JButton b1, b2, b3 ;
  public MaFenetre ()
{ this.setTitle ("Exemple strut et glue") ;
  this.setSize (150, 200) ;
  Container contenu = this.getContentPane() ;
  bVert = Box.createVerticalBox() ;
  contenu.add(bVert) ;
  b1 = new JButton ("Bouton1") ;
  bVert.add (b1);
  bVert.add (Box.createVerticalStrut(10)) ; // espace 10 pixels
  b2 = new JButton ("Bouton2") ;
  bVert.add (b2) ;
  bVert.add (Box.createGlue()) ; // espacement maximal
  b3 = new JButton ("Bouton3") ;
  bVert.add (b3) ;  }  }
```

GridBagLayout

- Objet GridBagLayout : Grille sous forme d'un tableau Excel où on peut positionner les composants en se servant des coordonnées des cellules.



CardLayout

- Objet CardLayout : Gérer les conteneurs comme un tas de cartes (les uns sur les autres), et basculer d'un contenu à l'autre.