IUT Lyon 1

l'excellence technologique

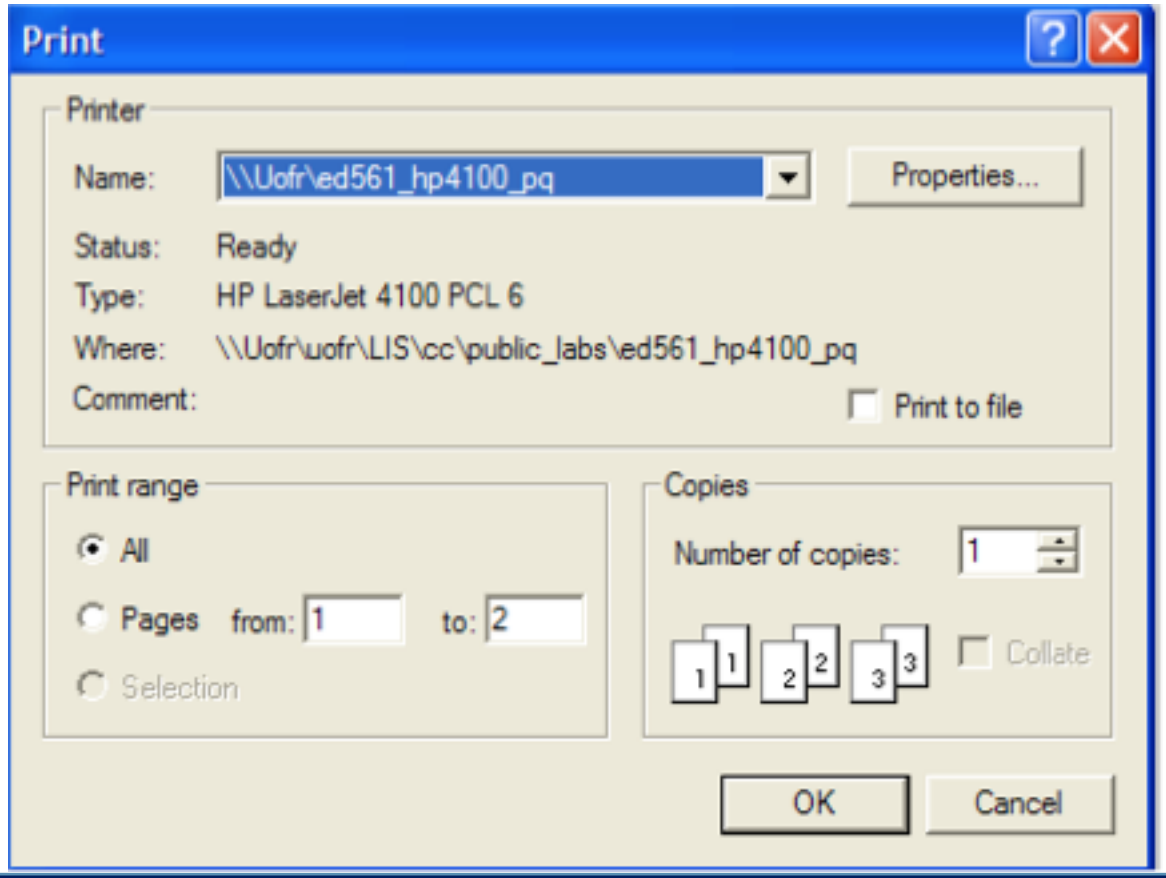
Java Avancé - Cours 1 (suite)

V. DESLANDRES, I. GUIDARA

veronique.deslandres@univ-lyon1.fr

Avril 2020

Sommaire de ce cours

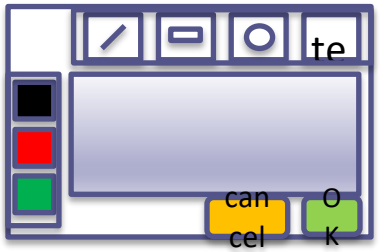


Placer les composants dans la fenêtre (Layout)

Layout : répartir les composants dans la fenêtre

L'intérêt principal des *layouts* permet de bien gérer les redimensionnements de la fenêtre. Généralités :

- Les composants sont **toujours placés dans** un conteneur
  - Soit **celui de la fenêtre**, sinon dans un **panneau (`JPanel`)**
- On peut définir *n* panneaux sur une même fenêtre, par ex. :
  - Panneaux ligne « précédent, suivant » pour une navigation
  - Panneau «choix des couleurs »,
  - Panneau des boutons d'interactions avec l'utilisateur, etc.



- On peut dynamiquement les rendre **visible** ou pas
- Tous les containers de haut niveau (`JFrame, JWindow,…`) ont un *contentPane* qui reçoit les éléments :
  - `frame.getContentPane();` *par défaut, retourne un `Container`*
- On peut **convertir** le conteneur de la fenêtre en panneau par :
  - `JPanel c =(JPanel)getContentPane();`

Fenêtre sans Layout

```
public class MaFenetre extends JFrame {
    private JButton btVoir, btDebut, btPrecedent, btSuivant, btFin;

    // Constructeur
    public MaFenetre() {
     initComponents();
     this.setTitle("Une Première fenêtre");
     this.setResizable(false);
     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

private void initComponents() {
    btVoir = new javax.swing.JButton("Voir");
    btDebut = new javax.swing.JButton("Début");
    btPrecedent = new javax.swing.JButton("Précédent");
    btSuivant = new javax.swing.JButton("Suivant");
    btFin = new javax.swing.JButton("Fin");

    Container cp = this.getContentPane();
    cp.add(btVoir);
    cp.add(btDebut);
    cp.add(btPrecedent);
    cp.add(btSuivant);
    cp.add(btFin);

}
```

1

2

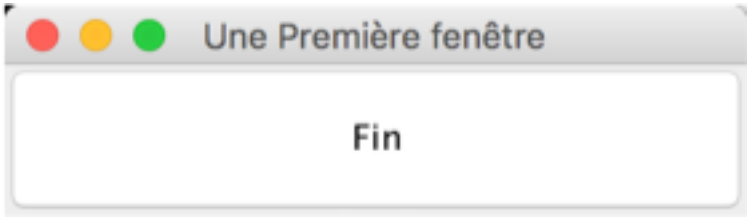
3

3 étapes

1- **Instanciation** des **composants**

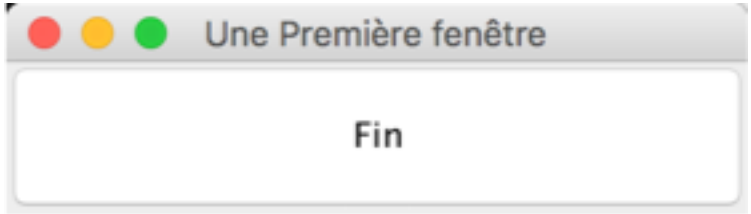
2- **Récupération** du **conteneur de la fenêtre**

3- **Insertion** des **composants** dans le **conteneur**



Répartition des composants

- Pour placer un composant : méthode `add` de `JPanel`
- PROBLÈME : la méthode `add` ajoute toujours le composant **au même endroit** dans le conteneur
  - Ainsi lorsqu'on veut ajouter plusieurs composants dans le panel, seul le dernier composant apparaît
  - Il faut donc **répartir** les composants avec la méthode `setLayout()`

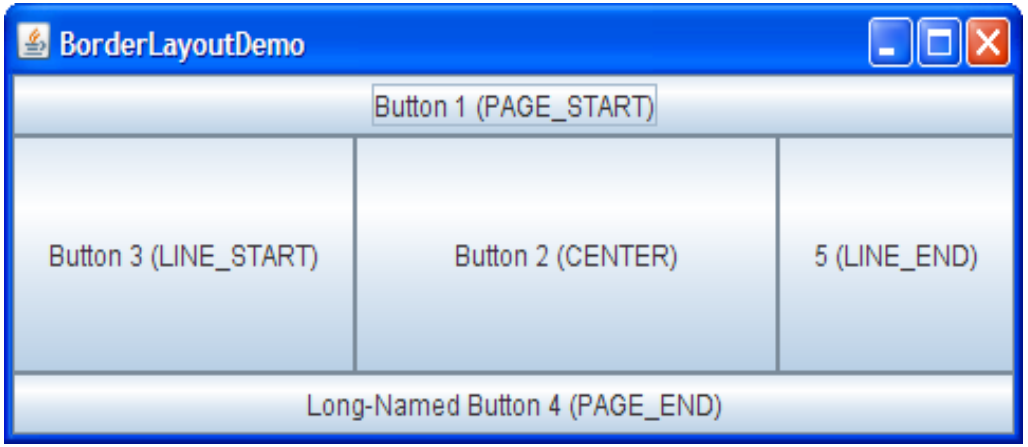


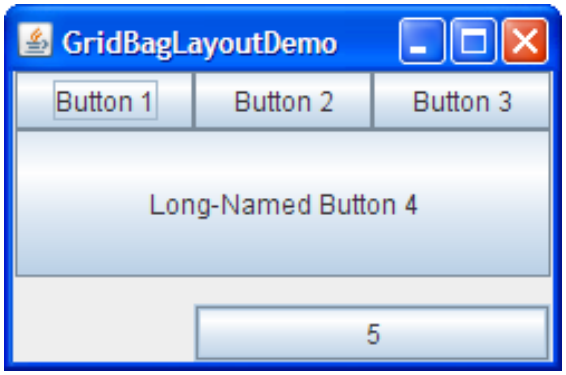
Répartition des composants (Layout)

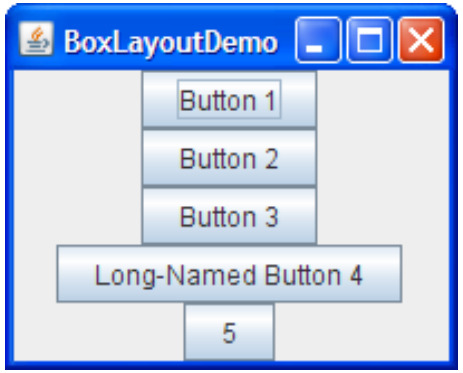
- Pour gérer la disposition des composants
  - Il existe des modes de répartitions *prédéfinis*
  - On choisit le **gestionnaire de répartition** avec `setLayout(monLayout)`

- Les gestionnaires de répartition appartiennent au package `java.awt`
  - `GridLayout`
  - `BorderLayout` (disposition par défaut dans une `JFrame`)
  - `FlowLayout` (disposition par défaut dans un `JPanel`)
  - Etc : `xxxLayout`

Quelques *layouts* de Swing Java





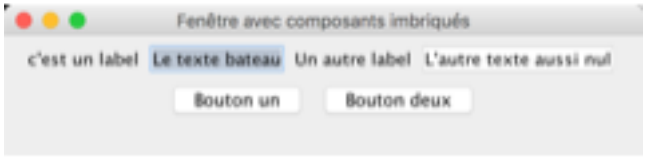


FlowLayout (1/4)

- Place les composants *les uns à la suite des autres*, de façon centrée
  - ligne par ligne, en passant à la ligne suivante si nécessaire
  - C'est le layout par défaut des `JPanel`

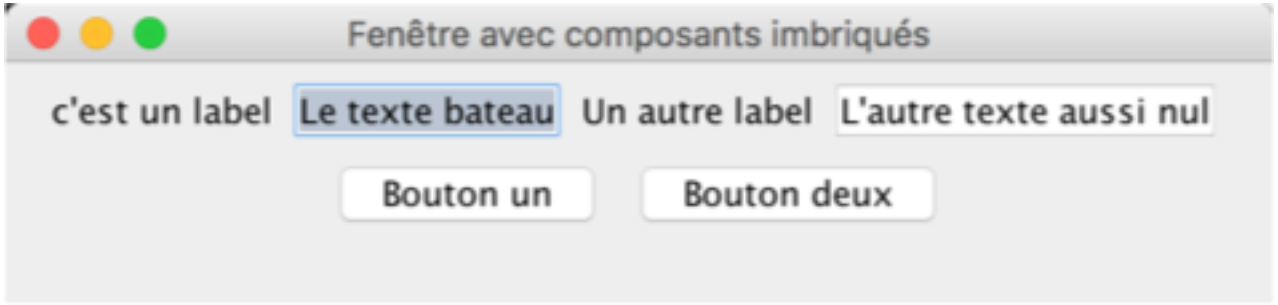
- C'est une classe, avec 3 constructeurs :
  - `FlowLayout()` *(CENTER par défaut)*
  - `FlowLayout(int align)`
    - Paramètre d'alignement (`FlowLayout.LEFT` ou `RIGHT` ou `CENTER`)
  - `FlowLayout(int align, int hgap, int vgap)`
    - hgap est l'espacement horizontal (d'une colonne à une autre)
    - vgap est l'espacement vertical (d'une ligne à une autre)

- On définit le mode de disposition du panneau avec `setLayout()` :
  - `panel.setLayout(new FlowLayout());`
- Puis on transmet le **nom** du composant à ajouter à la méthode `add` :
  - `panel.add(bouton1);`



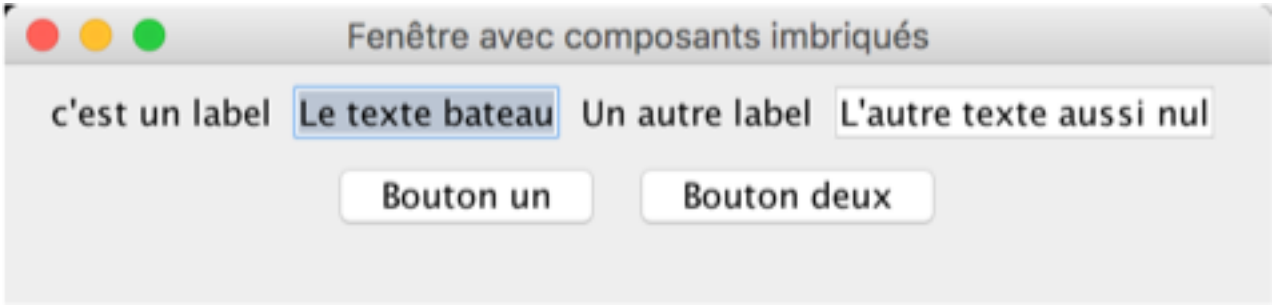
FlowLayout (2/4)

```
public class FenFlowLayout1 extends JFrame {

    private JLabel label1, label2;
    private JTextField texte1, texte2;
    private JButton bouton1, bouton2;

    // Constructeur
    public FenFlowLayout1() {
        initComponents();
        setTitle("Fenêtre avec composants imbriqués");
        //this.setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
```



FlowLayout (3/4)

```
private void initComponents() {

    // Initialisation des composants
    label1 = new JLabel("c'est un label");
    label2 = new JLabel("Un autre label");
    bouton1 = new javax.swing.JButton("Bouton un");
    bouton2 = new javax.swing.JButton("Bouton deux");
    texte1 = new JTextField("Le texte bateau");
    texte2 = new JTextField("L'autre texte aussi nul");

    // on récupère le contentPane de la fenêtre
    JPanel cp = (JPanel) this.getContentPane();
    cp.setLayout( new FlowLayout());
    // necessaire car cast de JPanel ==> pas de layout par defaut

    cp.add(label1);
    cp.add(texte1);
    cp.add(label2);
    cp.add(texte2);
    cp.add(bouton1);
    cp.add(bouton2);

}
```



FlowLayout (4/4)

Appel au constructeur FlowLayout() avec des paramètres :

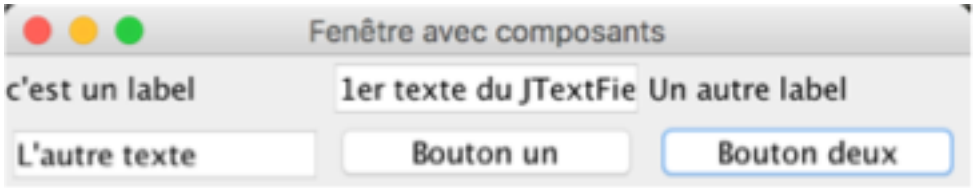
```
// on récupère le contentPane de la fenêtre
JPanel cp = (JPanel) this.getContentPane();
cp.setLayout( new FlowLayout(2, 20, 5));
```

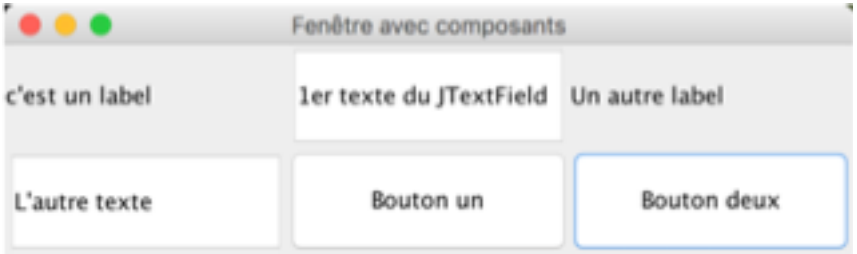
2 ou FlowLayout.RIGHT

20 : espacement horizontal
5 : espacement vertical

GridLayout (1/3)

- Place les composants dans une grille
  - Chaque composant occupe un espace (cellule) de **même dimension**
- Possède 3 constructeurs :
  - `GridLayout()`
  - `GridLayout(int rows, int col)`
    - *rows* = nombre de lignes et *col* = nombre de colonnes
  - `GridLayout(int rows, int col, int hgap, int vgap)`
- On définit le mode de disposition et on transmet le nom du composant à ajouter:
  - `panel.setLayout(new GridLayout(2,3));`
  - `panel.add(bouton1);`

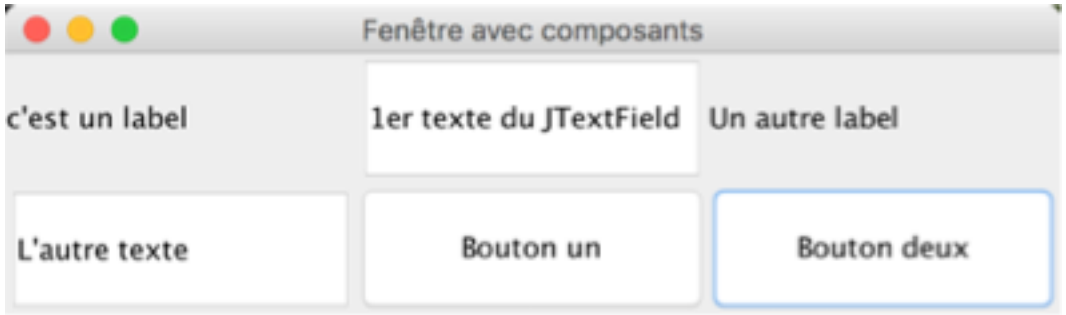




GridLayout(2/3)

```
private void initComponents() {

    // Initialisation des composants
    label1 = new JLabel("c'est un label");
    label2 = new JLabel("Un autre label");
    bouton1 = new javax.swing.JButton("Bouton un");
    bouton2 = new javax.swing.JButton("Bouton deux");
    texte1 = new JTextField("1er texte du JTextField");
    texte2 = new JTextField("L'autre texte");

    // on récupère le contentPane de la fenêtre
    JPanel cp = (JPanel) this.getContentPane();
    cp.setLayout( new GridLayout(2, 3));

    cp.add(label1);
    cp.add(texte1);
    cp.add(label2);
    cp.add(texte2);
    cp.add(bouton1);
    cp.add(bouton2);
}
```

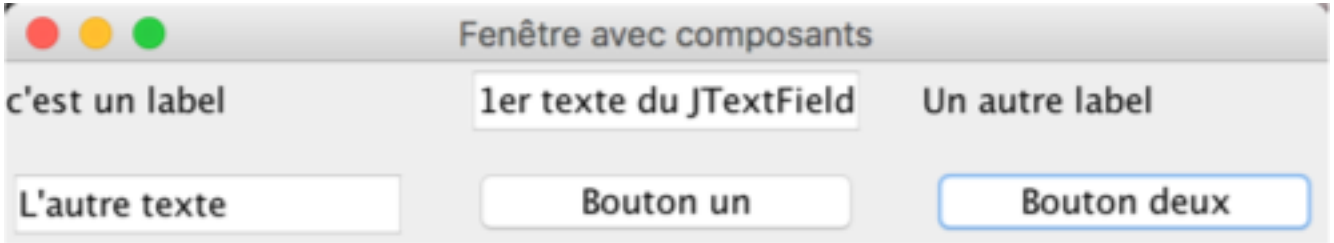
2 lignes et 3 colonnes



GridLayout(3/3)

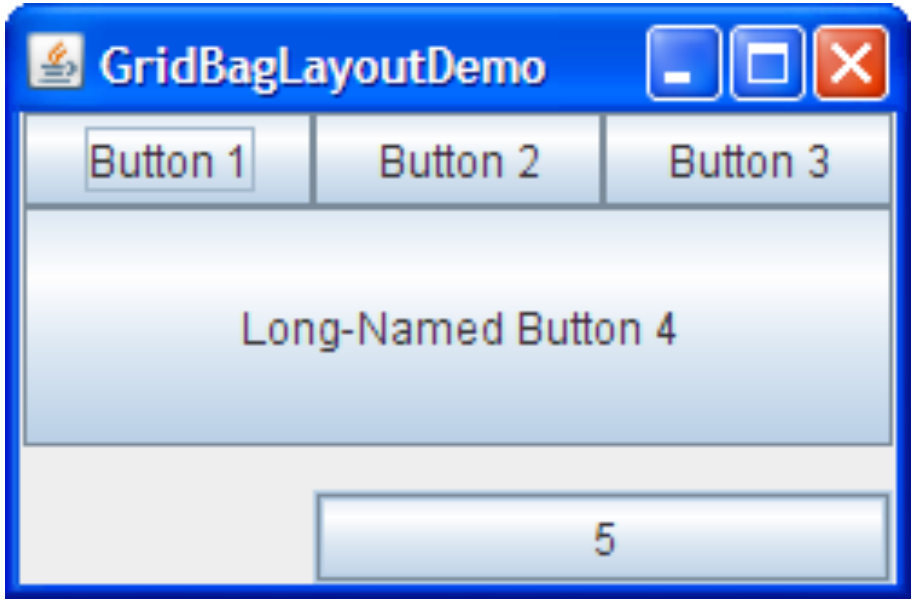
```
private void initComponents() {

    // Initialisation des composants
    label1 = new JLabel("c'est un label");
    label2 = new JLabel("Un autre label");
    bouton1 = new javax.swing.JButton("Bouton un");
    bouton2 = new javax.swing.JButton("Bouton deux");
    texte1 = new JTextField("1er texte du JTextField");
    texte2 = new JTextField("L'autre texte");

    // on récupère le contentPane de la fenêtre
    JPanel cp = (JPanel) this.getContentPane();
    cp.setLayout( new GridLayout(2, 3, 20, 10));

    cp.add(label1);
    cp.add(texte1);
    cp.add(label2);
    cp.add(texte2);
    cp.add(bouton1);
    cp.add(bouton2);
}
```

*2 lignes et 3 colonnes espacées de 20 pixels entre les colonnes (**hgap**) et de 10 pixels entre les lignes (**vgap**)*



GridBagLayout

- Il existe un autre *layout* qui permet d'avoir des composants qui occupent plusieurs cellules de la grille (le *spanning*)
- Classes `GridBagLayout` et `GridBagConstraints` : on définit les contraintes sur chaque composant à placer

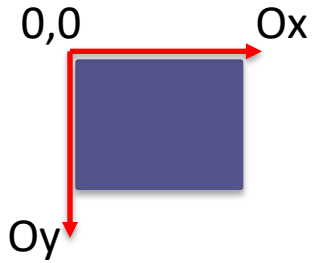


https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html

Absolute layout (1/3)

- Pour placer les composants à un endroit précis, il faut d'abord indiquer **qu'on n'utilise pas** de *LayoutManager* :
  - `ObjetConteneur.setLayout(null);`

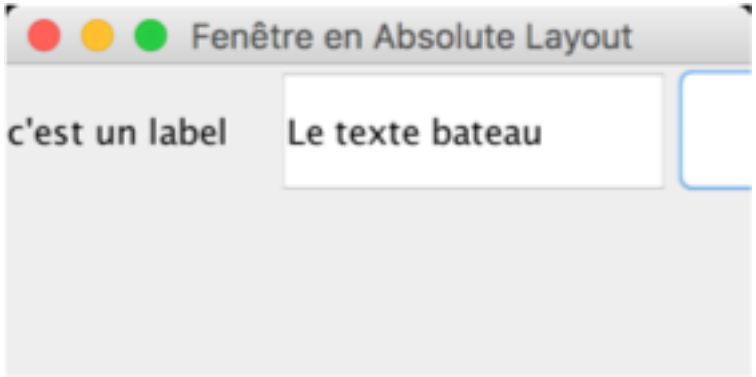
- On indique alors les coordonnées et la taille de chaque composant :
  - `Composant.setBounds(x,y,larg,haut);`
    - `x` et `y` sont les coordonnées du point en haut à gauche du conteneur
    - `larg` est la largeur du composant (axe des `x`)
    - `haut` est la hauteur du composant (axe des `y`)

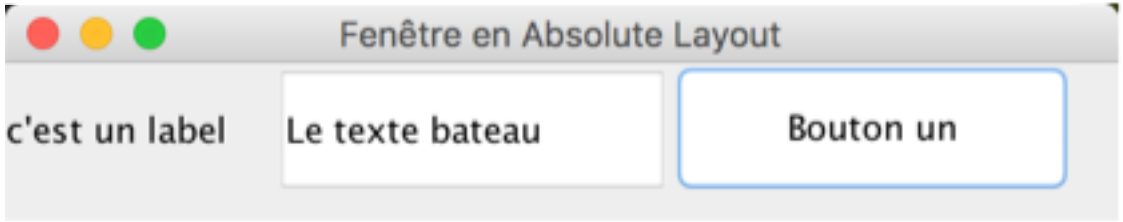


- LIMITES :
  - Les composants **ne sont pas redimensionnés** en cas de modification de la taille de la fenêtre
  - En cas de résolutions d'écran plus petites : le composant peut ne pas apparaître !

Absolute Layout (2/3)

```
private void initComponents() {

    // Initialisation des composants
    label1 = new JLabel("c'est un label");
    label2 = new JLabel("Un autre label");
    bouton1 = new javax.swing.JButton("Bouton un");
    bouton2 = new javax.swing.JButton("Bouton deux");
    texte1 = new JTextField("Le texte bateau");
    texte2 = new JTextField("L'autre texte aussi nul");

    // on récupère le contentPane de la fenêtre
    JPanel cp = (JPanel) this.getContentPane();
    cp.setLayout( null );

    label1.setBounds(0, 0, 100, 50);
    cp.add(label1);
    texte1.setBounds(100, 0, 150, 50);
    cp.add(texte1);
    bouton1.setBounds(250, 0, 150, 50);
    cp.add(bouton1);

}
```



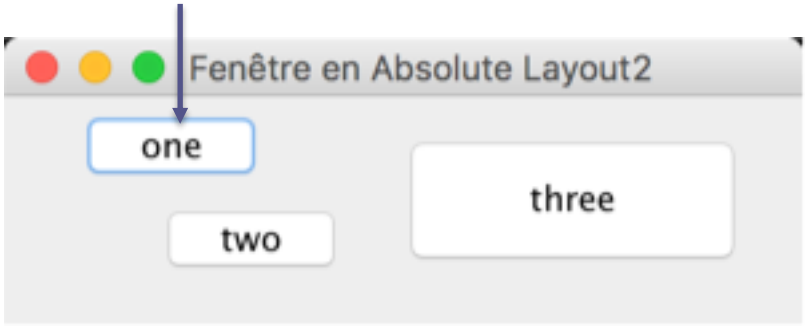


Absolute Layout (3/3)

```
private void initComponents() {
    b1 = new JButton("one");
    b2 = new JButton("two");
    b3 = new JButton("three");

    JPanel cp = (JPanel) this.getContentPane();
    cp.setLayout(null);

    cp.add(b1);
    cp.add(b2);
    cp.add(b3);

    Insets insets = cp.getInsets();

    Dimension size = b1.getPreferredSize();
    b1.setBounds(25 + insets.left, 5 + insets.top,
            size.width, size.height);

    size = b2.getPreferredSize();
    b2.setBounds(55 + insets.left, 40 + insets.top,
            size.width, size.height);

    size = b3.getPreferredSize();
    b3.setBounds(150 + insets.left, 15 + insets.top,
            size.width + 50, size.height + 20);
}
```

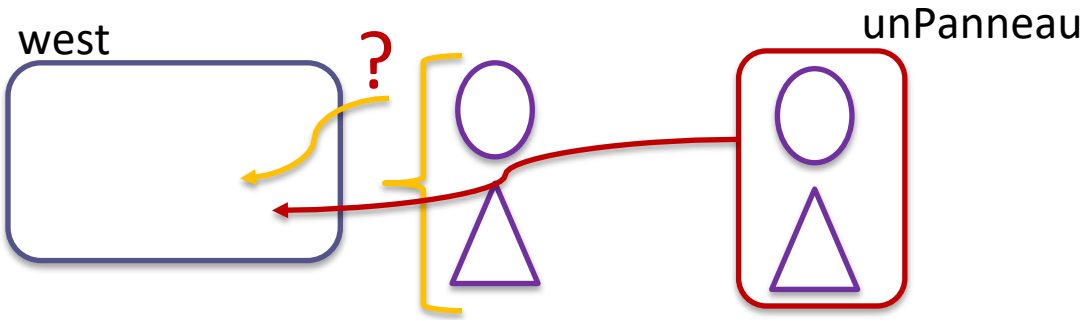
On peut aussi exploiter la taille de la fenêtre et placer les composants par rapport aux côtés

Ici b1 est placé à 25 pixels du bord G et à pixels du haut de la fenêtre



Si on veut mettre plus d'un composant dans une région ?

- Est-ce **possible** ?
- Comment procéder ?



Panneaux Imbriqués

Panneaux imbriqués

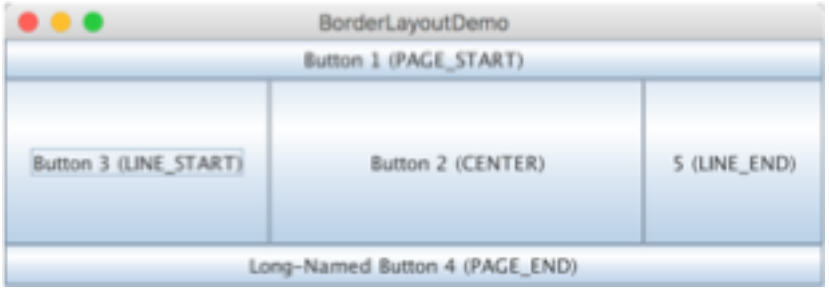
- Pour mettre plus d'un composant dans une zone, on utilise un **panneau** intermédiaire, placé dans la zone
  - dans lequel on positionne les composants

- Un *container* peut être inséré dans un autre *container*, souvent on ajoute des **JPanel**

- C'est la même méthode que pour **ajouter un composant** : on utilise la `add` :
  - `unPanel.add(unAutrePanel);`

BorderLayout (1/4)

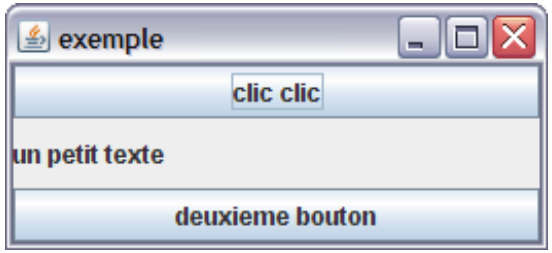
- Le *layout* par défaut d'une `JFrame`



- Découpe l'écran en 5 régions *(anciens noms)* :
  - **PAGE_END** ***(south)*** **PAGE_START** ***(north)***
  - **LINE_START** ***(east)*** **LINE_END** ***(west)***
  - **CENTER**

- On peut n'en utiliser que certaines, mais tout l'espace sera occupé

Il n'y a qu'un seul composant par région



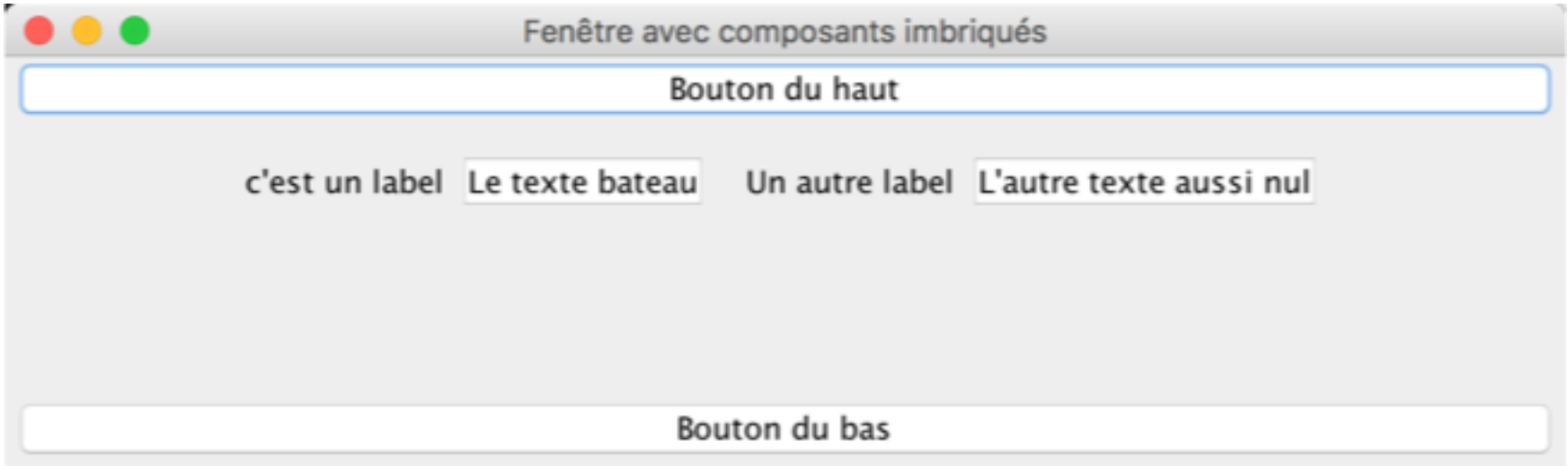
- Possède 2 constructeurs :
  - `BorderLayout()`
  - `BorderLayout(int hgap, int vgap)`
- Définition de la disposition en BorderLayout :

```
c.setLayout(new BorderLayout(10,5));
```

- On transmet à la méthode `add` le **nom** du composant, et **l'emplacement** souhaité :
  - `c.add(bouton1, BorderLayout.PAGE_START);`

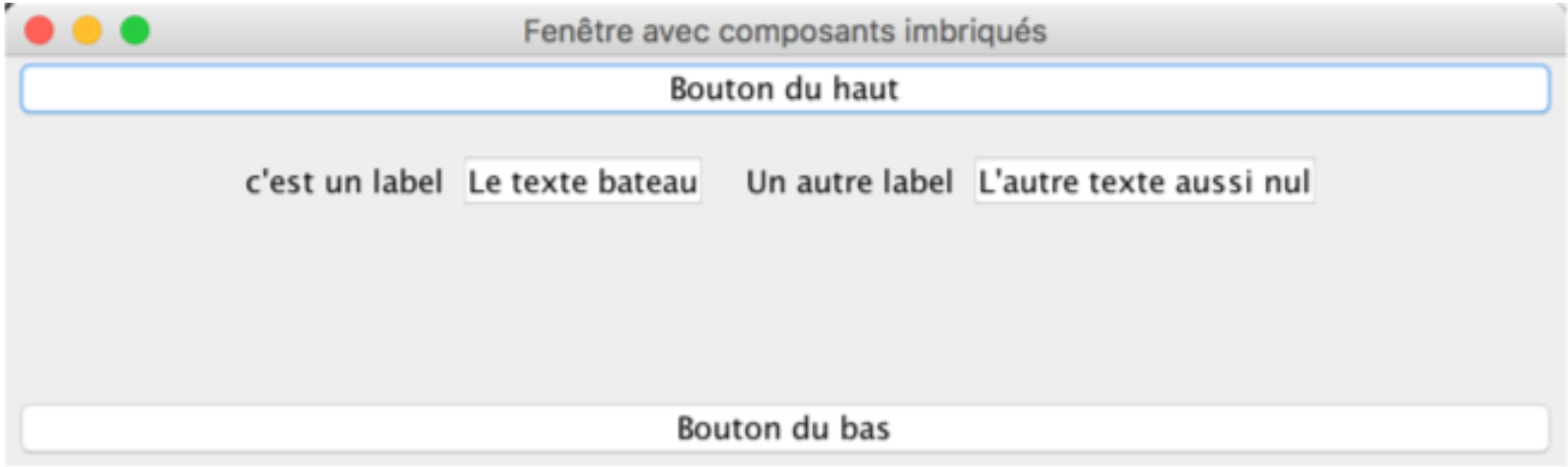
BorderLayout (2/4)

```
public class FenBorderLayout extends JFrame {

    private JLabel lblGauche, lblDroit;
    private JTextField texte1, texte2;
    private JButton enHautBouton, enBasBouton;

    // Constructeur
    public FenBorderLayout() {
        initComponents();
        setTitle("Fenêtre avec composants imbriqués");
        //this.setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     }
```



BorderLayout (3/4)

```
private void initComponents() {

    // Initialisation des composants
    lblGauche = new JLabel("c'est un label");
    lblDroit = new JLabel("Un autre label");
    enHautBouton = new javax.swing.JButton("Bouton du haut");
    enBasBouton = new javax.swing.JButton("Bouton du bas");
    texte1 = new JTextField("Le texte bateau");
    texte2 = new JTextField("L'autre texte aussi nul");

    // on récupère le contentPane de la fenêtre
    JPanel cp = (JPanel) this.getContentPane();
    cp.setLayout( new BorderLayout() );
```



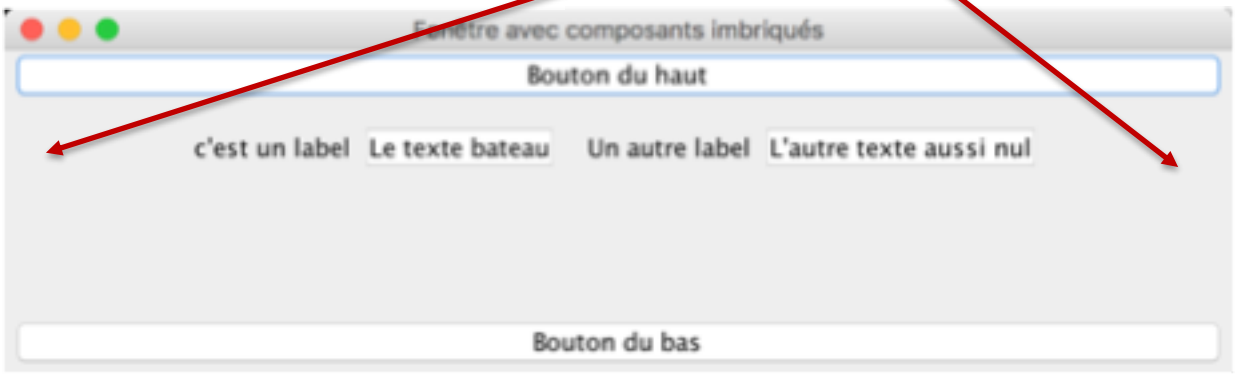
BorderLayout (4/4)

(suite de initComponents())

```
// Besoin de panneaux intermédiaires :
JPanel pannGr1, pannGr2, pannCentre;
// on définit les autres panneaux :
pannGr1 = new JPanel(); // par défaut en FlowLayout Centré
pannGr1.add(lblGauche);
pannGr1.add(texte1);

pannGr2 = new JPanel();
pannGr2.add(lblDroit);
pannGr2.add(texte2);

pannCentre = new JPanel();
pannCentre.add(pannGr1);
pannCentre.add(pannGr2);

cp.add(enHautBouton, BorderLayout.PAGE_START);
cp.add(pannCentre, BorderLayout.CENTER);
cp.add(enBasBouton, BorderLayout.PAGE_END);

}
```

Par défaut, un Panel est en FlowLayout centré : inutile de le préciser

NOTA : ici pas de panneaux latéraux : le CENTER prend toute la place



Autres exemples de panneaux imbriqués

- Des panneaux en *BorderLayout* placés dans un *BorderLayout*
- Des panneaux en *GridLayout* placés dans un *GridLayout*
- Un mix de *BorderLayout* dans un *GridLayout*

BorderLayout imbriqués

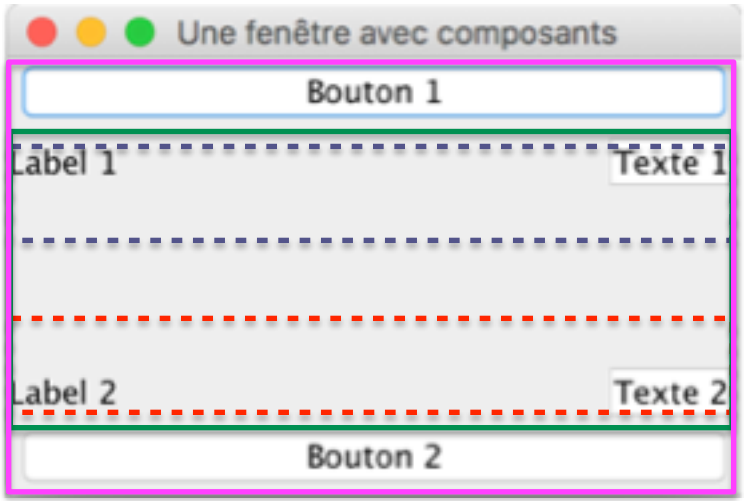
```
public class PanneauxImbriquésBorder extends JFrame{
    JLabel label1, label2;
    JTextField text1, text2;
    JButton bouton1, bouton2;
    JPanel panel1, panel2, panel3, panel4;
    public PanneauxImbriquésBorder(){
        //Création des composants
        label1=new JLabel("Label 1");
        label2=new JLabel("Label 2");
        text1=new JTextField("Texte 1");
        text2=new JTextField("Texte 2");
        bouton1=new JButton("Bouton 1");
        bouton2=new JButton("Bouton 2");
        panel1=(JPanel) getContentPane();
        panel1.setLayout(new BorderLayout());
        panel2= new JPanel(new BorderLayout());
        panel3= new JPanel(new BorderLayout());
        panel4= new JPanel(new BorderLayout());
        //Ajout des composants dans panel2
        panel2.add(label1, BorderLayout.WEST);
        panel2.add(text1, BorderLayout.EAST);
        //Ajout des composants dans panel3
        panel3.add(label2, BorderLayout.WEST);
        panel3.add(text2, BorderLayout.EAST);
        //Ajout des composants dans panel4
        panel4.add(panel2, BorderLayout.NORTH);
        panel4.add(panel3, BorderLayout.SOUTH);
        //Ajout des composants dans panel1
        panel1.add(bouton1, BorderLayout.NORTH);
        panel1.add(panel4, BorderLayout.CENTER);
        panel1.add(bouton2, BorderLayout.SOUTH);
    }
    public static void main (String[] args){
        JFrame fen=new PanneauxImbriquésBorder();
        fen.setSize(300,200);
        fen.setTitle("Une fenêtre avec composants");
        fen.setVisible(true);
    }
}
```

Panel 2

Panel 3

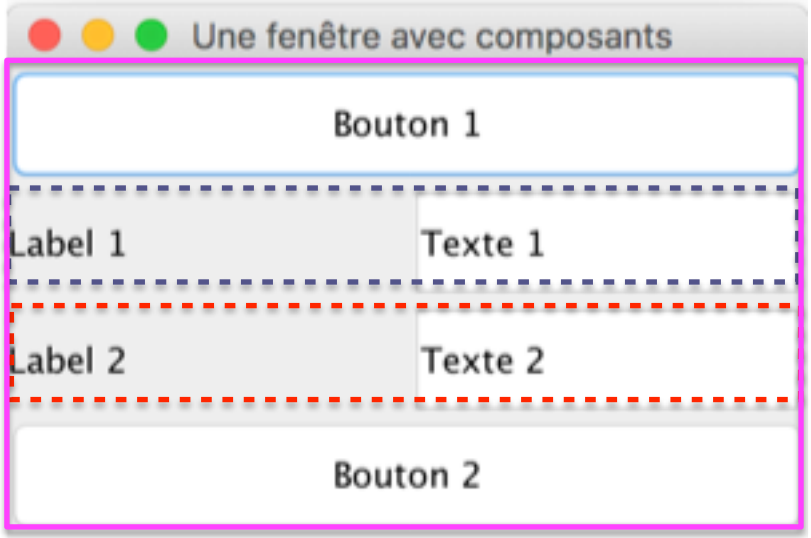
Panel 4

Panel 1



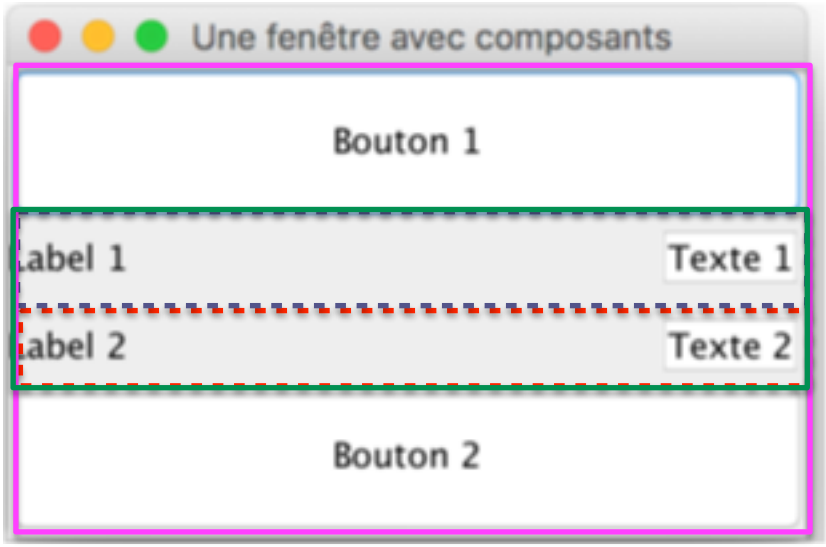
GridLayout imbriqués

```
public class PanneauxImbriquésGrid extends JFrame{
    JLabel label1, label2;
    JTextField text1, text2;
    JButton bouton1, bouton2;
    JPanel panel1, panel2, panel3;
    public PanneauxImbriquésGrid(){
        //Création des composants
        label1=new JLabel("Label 1");
        label2=new JLabel("Label 2");
        text1=new JTextField("Texte 1");
        text2=new JTextField("Texte 2");
        bouton1=new JButton("Bouton 1");
        bouton2=new JButton("Bouton 2");
        panel1=(JPanel) getContentPane();
        panel1.setLayout(new GridLayout(4,1));
        panel2= new JPanel(new GridLayout(1,2));
        panel3= new JPanel(new GridLayout(1,2));
        //Ajout des composants dans panel2
        panel2.add(label1);
        panel2.add(text1);
        //Ajout des composants dans panel3
        panel3.add(label2);
        panel3.add(text2);
        //Ajout des composants dans panel1
        panel1.add(bouton1);
        panel1.add(panel2);
        panel1.add(panel3);
        panel1.add(bouton2);
    }
    public static void main (String[] args){
        JFrame fen=new PanneauxImbriquésGrid();
        fen.setSize(300,200);
        fen.setTitle("Une fenêtre avec composants");
        fen.setVisible(true);
    }
}
```



Grid & BorderLayout

```
public class PanneauxImbriquésBorderGrid extends JFrame{
    JLabel label1, label2;
    JTextField text1, text2;
    JButton bouton1, bouton2;
    JPanel panel1, panel2, panel3, panel4;
    public PanneauxImbriquésBorderGrid(){
        //Création des composants
        label1=new JLabel("Label 1");
        label2=new JLabel("Label 2");
        text1=new JTextField("Texte 1");
        text2=new JTextField("Texte 2");
        bouton1=new JButton("Bouton 1");
        bouton2=new JButton("Bouton 2");
        panel1=(JPanel) getContentPane();
        panel1.setLayout(new GridLayout(3,1));
        panel2= new JPanel(new BorderLayout());
        panel3= new JPanel(new BorderLayout());
        panel4= new JPanel(new BorderLayout());
        //Ajout des composants dans panel2
        panel2.add(label1, BorderLayout.WEST);
        panel2.add(text1, BorderLayout.EAST);
        //Ajout des composants dans panel3
        panel3.add(label2, BorderLayout.WEST);
        panel3.add(text2, BorderLayout.EAST);
        //Ajout des composants dans panel4
        panel4.add(panel2, BorderLayout.NORTH);
        panel4.add(panel3, BorderLayout.SOUTH);
        //Ajout des composants dans panel1
        panel1.add(bouton1);
        panel1.add(panel4);
        panel1.add(bouton2);
    }
    public static void main (String[] args){
        JFrame fen=new PanneauxImbriquésBorderGrid();
        fen.setSize(300,200);
        fen.setTitle("Une fenêtre avec composants");
        fen.setVisible(true);
    }
}
```



Affichage d'une image

- Pour afficher une image sur un panneau, on utilise un label
- 2 méthodes:
  - En utilisant l'objet `ImageIcon` directement dans la déclaration du label:

```
JLabel lbl_image=new JLabel(new ImageIcon("src/gif/IUT.png"));
```

  - La méthode `setIcon()` qui attend un objet `ImageIcon` :
    - `monlabel.setIcon(new ImageIcon(fichier_image));`

```
JLabel lbl_image=new JLabel();
lbl_image.setIcon(new ImageIcon("src/gif/IUT.png"));
```

- Attention sous Windows : comme en Java l'antislash est un marqueur de caractère spécial :'\n', '\t', etc.
  - utiliser le double antislash comme caractère de séparation

    `C:\\Users\\Anna\\workspace\\progIHM\\TP1…`

- Pas de problème sous Unix ou MacOS :
  - `monLabel.setIcon(new ImageIcon("src/gif/image1.gif"));`

Ajout d'une image en icône

- Pour affecter une image comme icône de l'application **SOUS WINDOWS**, 3 méthodes sont utilisées:

  - En utilisant la classe `Toolkit` qui contient la méthode `getDefaultToolkit()`. L'objet `Toolkit` contient la méthode `getImage()` qui prend l'URL de l'image et rend un objet de type `Image`.

```
//Affecter une image comme icône de l'application SOUS WINDOWS
Toolkit tk=Toolkit.getDefaultToolkit();
Image im = tk.getImage("src/gif/image.gif");
setIconImage(im);
```

  - En utilisant `ImageIcon`. La méthode `getImage()` retourne une instance image de `ImageIcon`

```
setIconImage(new ImageIcon("Java_logo.png").getImage());
```

  - En utilisant `ImageIO`. La méthode `read()` de la classe `ImageIO` prend un objet `InputStream` qui pointe vers le fichier de l'image.

```
try{
    setIconImage(ImageIO.read(new FileInputStream("Java_logo.png")));
}
catch(Exception e){}
```

Taille d'une fenêtre, d'un composant ou d'un panel

- Mettre une fenêtre au centre

```
JFrame frame=new JFrame();
frame.setLocationRelativeTo(null);
```

- Mettre une fenêtre sur tout l'écran

```
GraphicsEnvironment gEnv = GraphicsEnvironment.getLocalGraphicsEnvironment();
Rectangle bounds = gEnv.getMaximumWindowBounds();
frame.setBounds(bounds);
frame.setVisible(true);
```

- Pour redimensionner un panel ou un composant, on utilise la méthode `setPreferredSize()`

```
lbl_image.setPreferredSize(new Dimension(500,100));
```

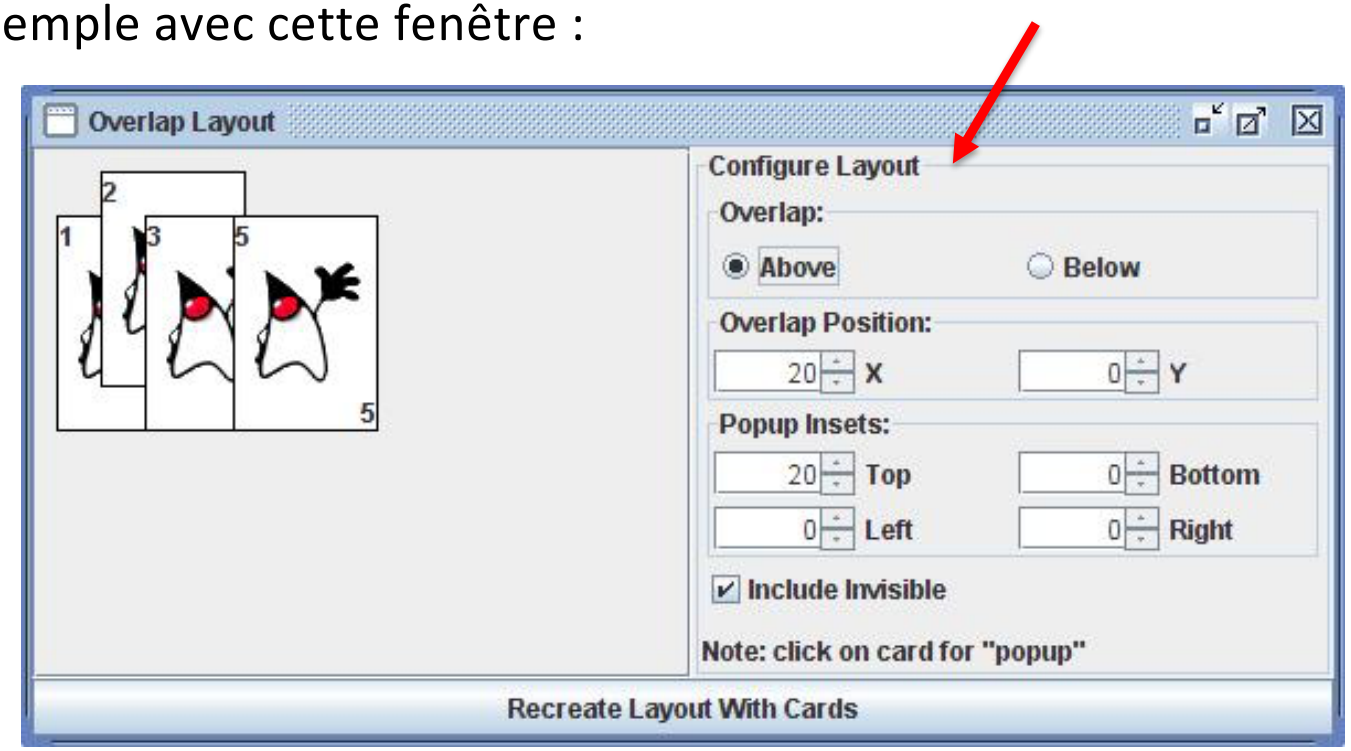
```
JPanel pan=(JPanel) getContentPane();
pan.setPreferredSize(new Dimension(200,300));
```

Bordure et titre d'un JPanel

- Pour ajouter une bordure à un panneau, il faut utiliser la méthode `setBorder()`

```
JPanel pan=(JPanel) getContentPane();
pan.setBorder(BorderFactory.createTitledBorder("Configure Layout"));
```

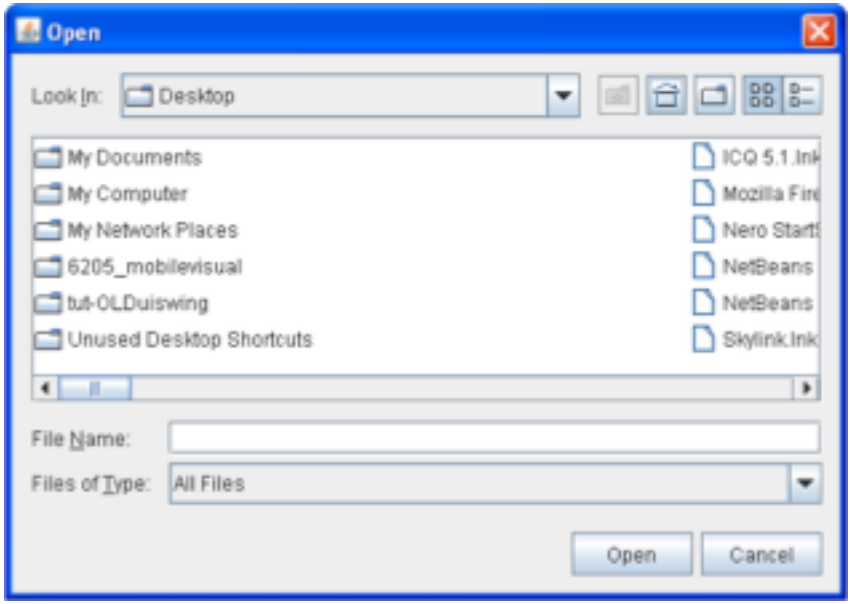
- Un exemple avec cette fenêtre :



Fenêtre de sélection de fichiers

- La **FileDialog** d'AWT : fenêtre de base permettant d'ouvrir ou d'enregistrer un fichier

- Le **JFileChooser** de SWING : fenêtre plus élaborée avec notamment la possibilité de filtrer les fichiers



Fenêtre de sélection de fichiers: FileDialog

Choix du fichier pour ouvrir

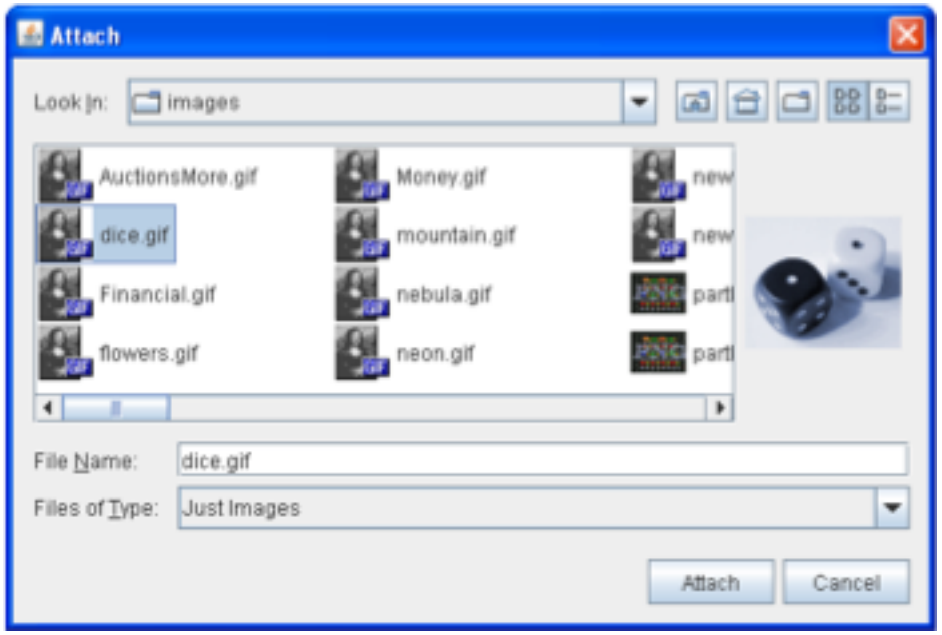
```
String nomFic = new String("");
try {
        // ouvrir un fichier
        FileDialog fd = new FileDialog(this , "Sélectionnez votre fichier...", FileDialog.LOAD);
        fd.setVisible(true);
        nomFic = ((fd.getDirectory()).concat(fd.getFile()));
}
catch (NullPointerException e) {
        System.out.println("Erreur ouverture dossier !");
}
```

Choix du fichier pour enregistrer

```
String nomFic = new String("");
try {
        FileDialog fd = new FileDialog(this , "Sélectionnez votre  fichier...", FileDialog.SAVE);
        fd.setVisible(true);
        nomFic = ((fd.getDirectory()).concat(fd.getFile()));
}
catch(NullPointerException e) {
    System.out.println("Erreur ouverture dossier !");
}
```

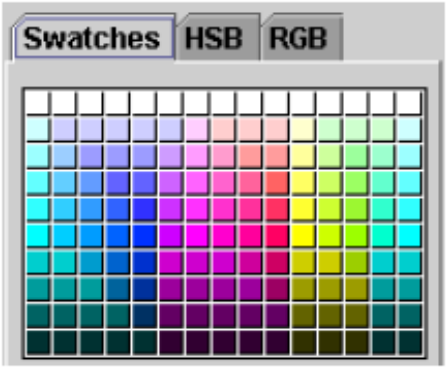
Fenêtre de sélection de fichiers: JFileChooser

- Beaucoup plus complet
- Fenêtre de dialogue permettant de choisir un fichier
- Regarder le tutoriel sur le site de SUN

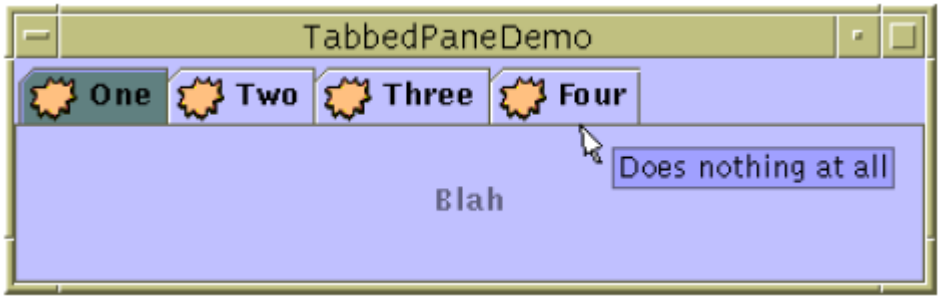


Autres composants utiles

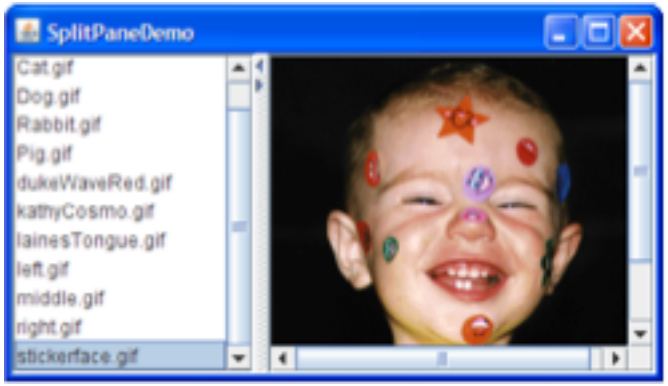
JColorChooser : fenêtre de dialogue permettant de choisir une couleur



JTabbedPane : permet de mettre plusieurs JPanel dans des onglets



JPlitPane : il s'agit d'un double conteneur permettant une interaction entre deux composants



JScrollPane : un conteneur permettant le défilement (ascenseur) si nécessaire