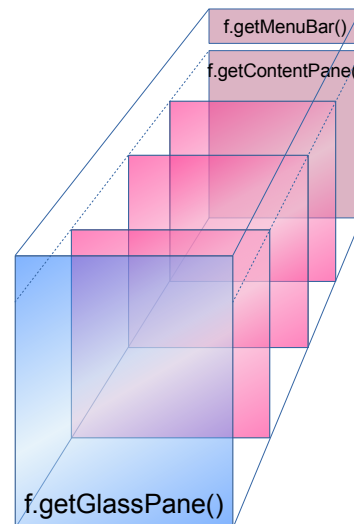
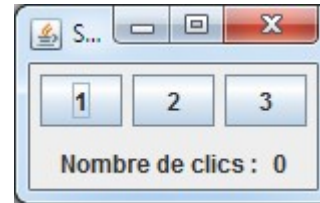
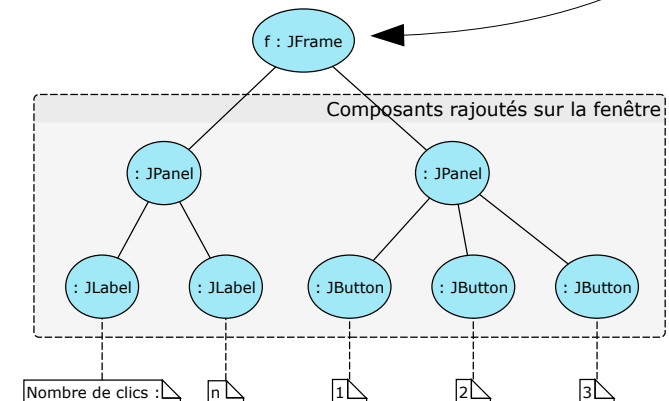
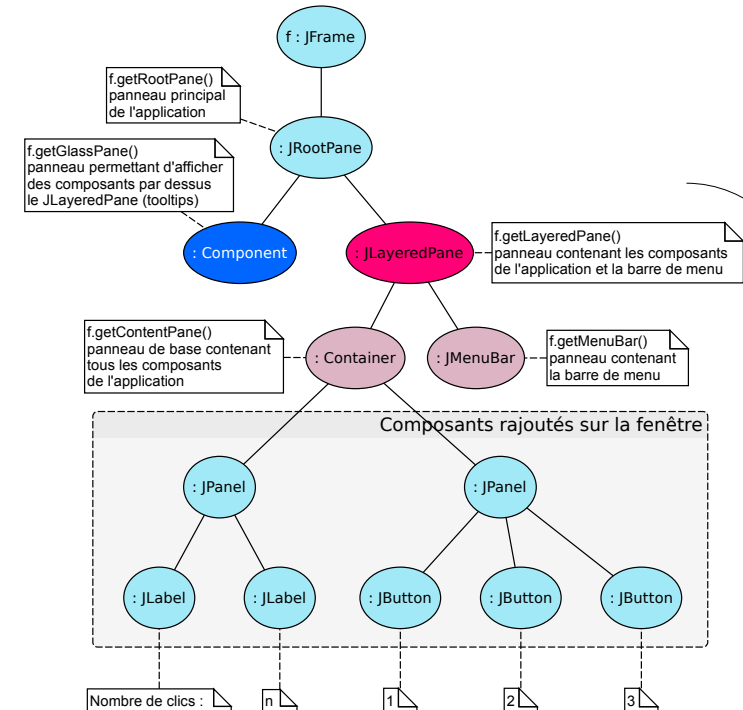


Applications graphiques Java

- I. Types imbriqués
 1. Définition
 2. Type membre statique
 3. Classe interne
 - i. Définition
 - ii. Classe membre non stat.
 - iii. Classe locale
 - iv. Classe anonyme
 - v. Étude de cas : itérateurs
- II. Présentation
 1. Application graphique
 2. Bibliothèque Swing
- III. Programmation événementielle
- IV. Architecture MVC
 1. Définition
 2. Exemple
 - i. Codage du modèle
 - ii. Codage de la vue
 - iii. Codage du contrôleur
 - iv. Organisation du code
- V. Composants graphiques
 1. Classes de base
 2. Méthodes de dessin
 3. Définir un composant
- VI. Placement des composants graphiques
 1. Hiérarchies de contenance
 - i. Présentation
 - ii. Visualisation dans le code
 - iii. Technique de codage
 - iv. Affichage des composants
 2. Gestionnaires de répartition
 - i. Présentation
 - ii. FlowLayout
 - iii. GridLayout
 - iv. BorderLayout



Hiérarchie de contenance :
arborescence des composants graphiques disposés sur un conteneur



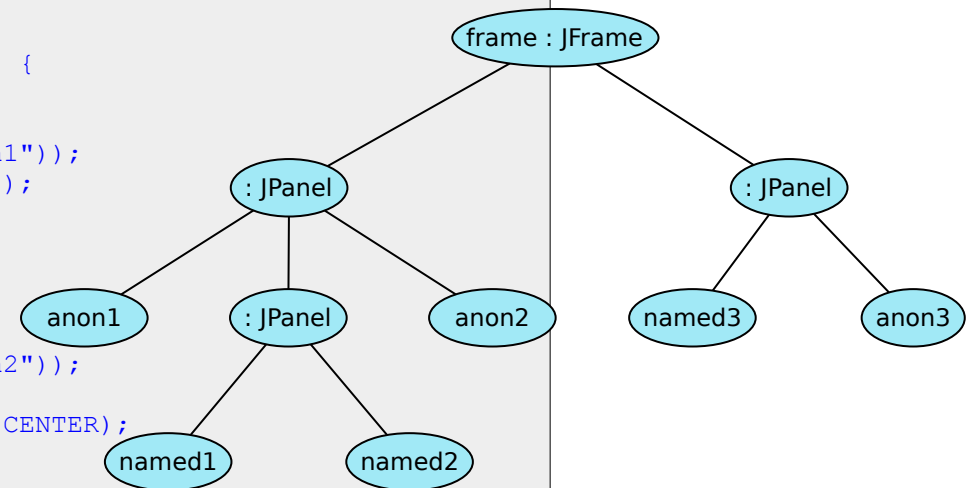
Applications graphiques Java

- I. Types imbriqués
 1. Définition
 2. Type membre statique
 3. Classe interne
 - i. Définition
 - ii. Classe membre non stat.
 - iii. Classe locale
 - iv. Classe anonyme
 - v. Étude de cas : itérateurs
- II. Présentation
 1. Application graphique
 2. Bibliothèque Swing
- III. Programmation événementielle
- IV. Architecture MVC
 1. Définition
 2. Exemple
 - i. Codage du modèle
 - ii. Codage de la vue
 - iii. Codage du contrôleur
 - iv. Organisation du code
- V. Composants graphiques
 1. Classes de base
 2. Méthodes de dessin
 3. Définir un composant
- VI. Placement des composants graphiques
 1. Hiérarchies de contenance
 - i. Présentation
 - ii. Visualisation dans le code
 - iii. Technique de codage
 - iv. Affichage des composants
 2. Gestionnaires de répartition
 - i. Présentation
 - ii. FlowLayout
 - iii. GridLayout
 - iv. BorderLayout

```
public MonAppli() {
    ...
    // VUE
    frame = new JFrame("Graphic Appli");
    named = createNamed();
}

private JButton[] createNamed() {
    JButton[] tab = new JButton[3];
    String[] s = new String[] { "named1", "named2", "named3" };
    for (int i = 0; i < s.length; i++) {
        tab[i] = new JButton(s[i]);
    }
    return tab;
}

private void placeComponents() {
    JPanel p = new JPanel();
    { //--
        p.add(new JLabel("anon1"));
        JPanel q = new JPanel();
        { //--
            q.add(named[0]);
            q.add(named[1]);
        } //--
        p.add(q);
        p.add(new JLabel("anon2"));
    } //--
    frame.add(p, BorderLayout.CENTER);
    p = new JPanel();
    { //--
        p.add(named[2]);
        p.add(new JLabel("anon3"));
    } //--
    frame.add(p, BorderLayout.SOUTH);
}
```

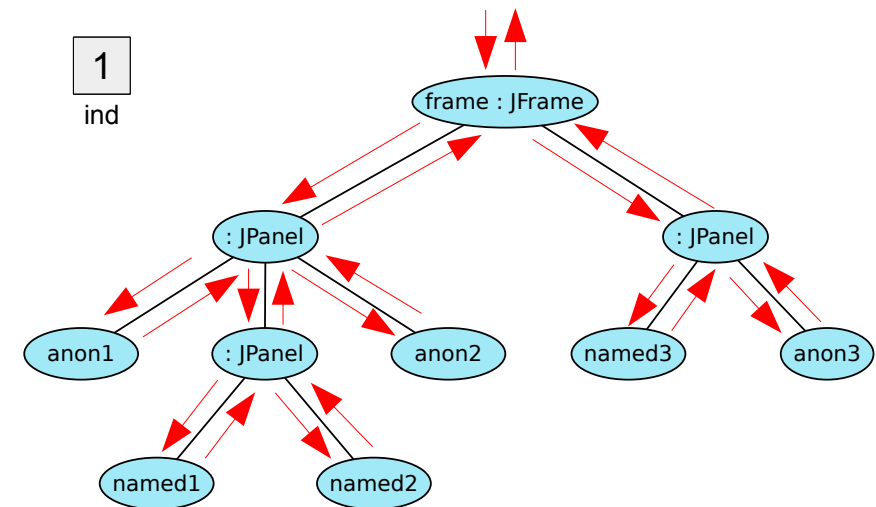


Applications graphiques Java

- I. Types imbriqués
 1. Définition
 2. Type membre statique
 3. Classe interne
 - i. Définition
 - ii. Classe membre non stat.
 - iii. Classe locale
 - iv. Classe anonyme
 - v. Étude de cas : itérateurs
- II. Présentation
 1. Application graphique
 2. Bibliothèque Swing
- III. Programmation événementielle
- IV. Architecture MVC
 1. Définition
 2. Exemple
 - i. Codage du modèle
 - ii. Codage de la vue
 - iii. Codage du contrôleur
 - iv. Organisation du code
- V. Composants graphiques
 1. Classes de base
 2. Méthodes de dessin
 3. Définir un composant
- VI. Placement des composants graphiques
 1. Hiérarchies de contenance
 - i. Présentation
 - ii. Visualisation dans le code
 - iii. Technique de codage
 - iv. Affichage des composants
 2. Gestionnaires de répartition
 - i. Présentation
 - ii. FlowLayout
 - iii. GridLayout
 - iv. BorderLayout

parcours d'une hiérarchie de contenance	arrivée	départ
racine	<code>private void placeComponents() { ind ← ind + 1</code>	<code>ind ← ind - 1 }</code>
noeud interne	<code>[JPanel] p = new JPanel([...]); { //-- ind ← ind + 1</code>	<code>ind ← ind - 1 } //-- parent.add(p[, ...]);</code>
feuille	<i>ne rien faire</i>	<code>p.add(c[, ...]);</code>

```
private void placeComponents() {
    JPanel p = new JPanel();
    { //--
        p.add(new JLabel("anon1"));
        JPanel q = new JPanel();
        { //--
            q.add(named[0]);
            q.add(named[1]);
        } //--
        p.add(q);
        p.add(new JLabel("anon2"));
    } //--
    frame.add(p, BorderLayout.CENTER);
    p = new JPanel();
    { //--
        p.add(named[2]);
        p.add(new JLabel("anon3"));
    } //--
    frame.add(p, BorderLayout.SOUTH);
}
```



Applications graphiques Java

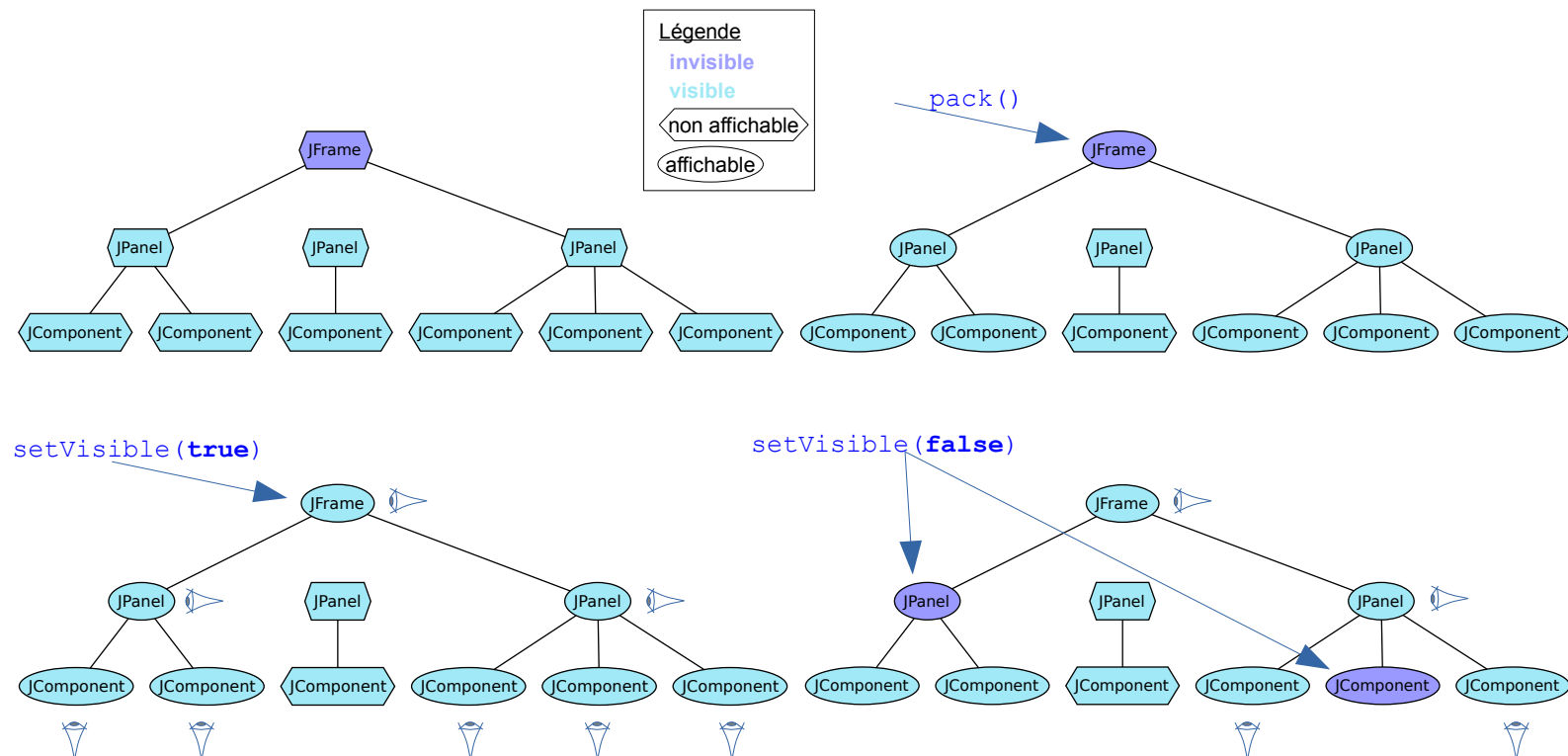
- I. Types imbriqués
 1. Définition
 2. Type membre statique
 3. Classe interne
 - i. Définition
 - ii. Classe membre non stat.
 - iii. Classe locale
 - iv. Classe anonyme
 - v. Étude de cas : itérateurs
- II. Présentation
 1. Application graphique
 2. Bibliothèque Swing
- III. Programmation événementielle
- IV. Architecture MVC
 1. Définition
 2. Exemple
 - i. Codage du modèle
 - ii. Codage de la vue
 - iii. Codage du contrôleur
 - iv. Organisation du code
- V. Composants graphiques
 1. Classes de base
 2. Méthodes de dessin
 3. Définir un composant
- VI. Placement des composants graphiques
 1. Hiérarchies de contenance
 - i. Présentation
 - ii. Visualisation dans le code
 - iii. Technique de codage
 - iv. Affichage des composants
 2. Gestionnaires de répartition
 - i. Présentation
 - ii. FlowLayout
 - iii. GridLayout
 - iv. BorderLayout

Composant graphique affichable : détectable avec `isDisplayable()`
composant dont la hiérarchie de contenance est enracinée dans une fenêtre Swing affichable.

une fenêtre devient affichable après première exécution de `frame.pack()` ou de `frame.setVisible(true)`

Composant graphique visible : détectable avec `isVisible()`
composant dont la propriété de visibilité est vraie.

Composant graphique affiché à l'écran : détectable avec `isShowing()`
composant affichable et visible, dont chaque élément de la hiérarchie de contenance est visible.



Applications graphiques Java

- I. Types imbriqués
 1. Définition
 2. Type membre statique
 3. Classe interne
 - i. Définition
 - ii. Classe membre non stat.
 - iii. Classe locale
 - iv. Classe anonyme
 - v. Étude de cas : itérateurs
- II. Présentation
 1. Application graphique
 2. Bibliothèque Swing
- III. Programmation événementielle
- IV. Architecture MVC
 1. Définition
 2. Exemple
 - i. Codage du modèle
 - ii. Codage de la vue
 - iii. Codage du contrôleur
 - iv. Organisation du code
3. Diagramme de classes
- V. Composants graphiques
 1. Classes de base
 2. Méthodes de dessin
 3. Définir un composant
- VI. Placement des composants graphiques
 1. Hiérarchies de contenance
 - i. Présentation
 - ii. Visualisation dans le code
 - iii. Technique de codage
 - iv. Affichage des composants
 2. Gestionnaires de répartition
 - i. Présentation
 - ii. BorderLayout
 - iii. GridLayout
 - iv. BorderLayout

Gestionnaire de répartition :

objet non graphique qui gère intégralement la taille et la place des composants graphiques placés sur un conteneur

Association d'un gestionnaire à un conteneur

- par défaut :

```
JFrame (ContentPane) → BorderLayout
```

```
JPanel → FlowLayout (centré)
```

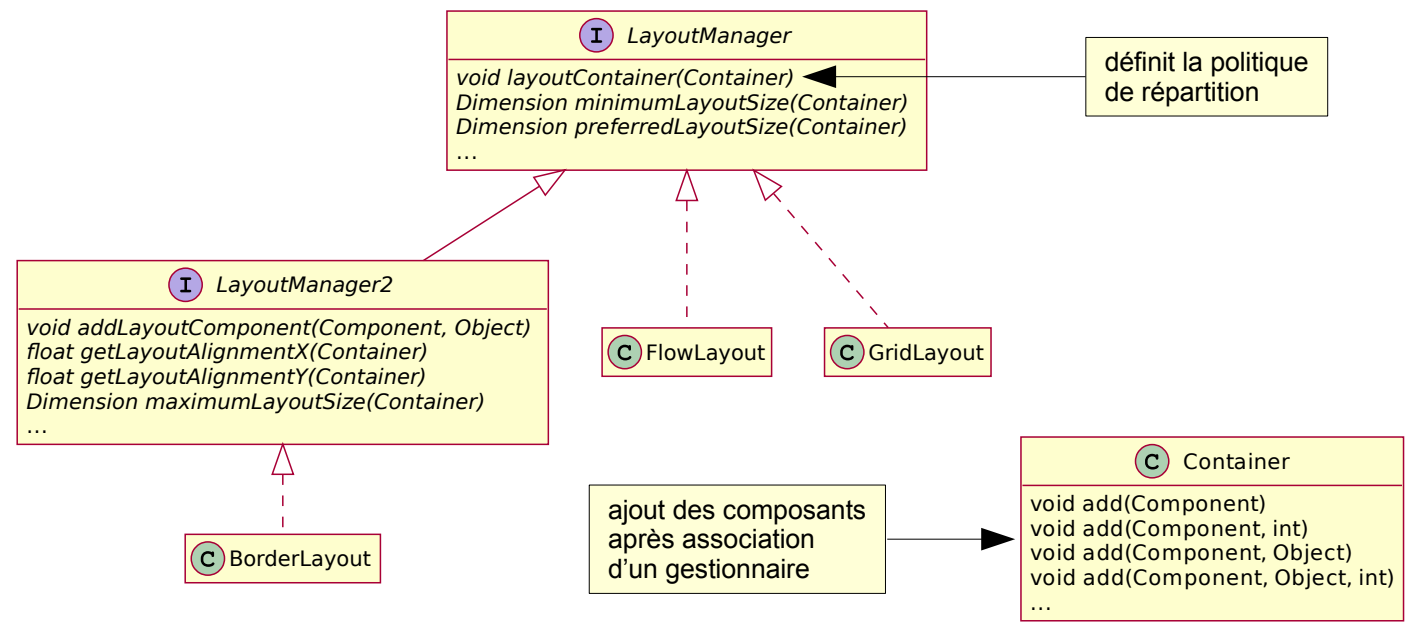
- initialisation :

```
panel = new JPanel(new <TypeGestionnaire>(...))
```

- changement :

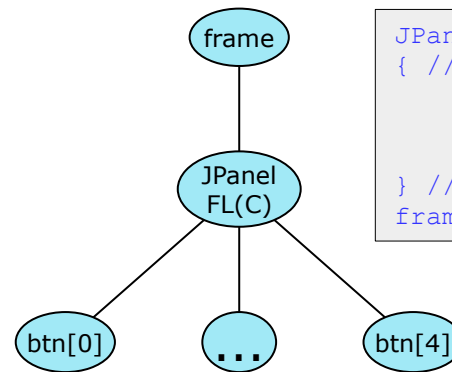
```
frame.setLayout(new <TypeGestionnaire>(...))
```

```
panel.setLayout(new <TypeGestionnaire>(...))
```



Applications graphiques Java

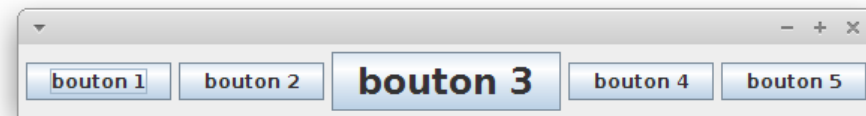
- I. Types imbriqués
 1. Définition
 2. Type membre statique
 3. Classe interne
 - i. Définition
 - ii. Classe membre non stat.
 - iii. Classe locale
 - iv. Classe anonyme
 - v. Étude de cas : itérateurs
- II. Présentation
 1. Application graphique
 2. Bibliothèque Swing
- III. Programmation événementielle
- IV. Architecture MVC
 1. Définition
 2. Exemple
 - i. Codage du modèle
 - ii. Codage de la vue
 - iii. Codage du contrôleur
 - iv. Organisation du code
- V. Composants graphiques
 1. Classes de base
 2. Méthodes de dessin
 3. Définir un composant
- VI. Placement des composants graphiques
 1. Hiérarchies de contenance
 - i. Présentation
 - ii. Visualisation dans le code
 - iii. Technique de codage
 - iv. Affichage des composants
 2. Gestionnaires de répartition
 - i. Présentation
 - ii. FlowLayout
 - iii. GridLayout
 - iv. BorderLayout



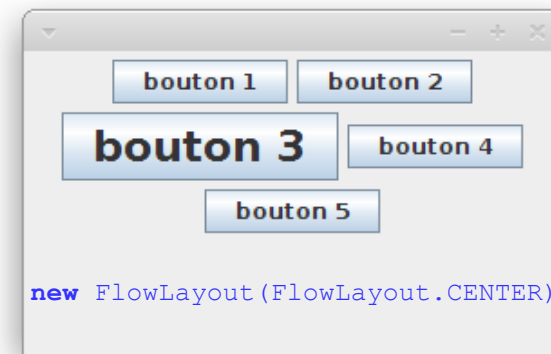
```

JPanel p = new JPanel(new FlowLayout(FlowLayout.CENTER));
{ //--
    for (JButton b : btn) {
        p.add(b);
    }
} //--
frame.add(p);
    
```

après `frame.pack()`



après retailage à la souris



`new FlowLayout(FlowLayout.LEFT)`



`new FlowLayout(FlowLayout.RIGHT)`



Applications graphiques Java

I. Types imbriqués

1. Définition
2. Type membre statique
3. Classe interne
 - i. Définition
 - ii. Classe membre non stat.
 - iii. Classe locale
 - iv. Classe anonyme
 - v. Étude de cas : itérateurs

II. Présentation

1. Application graphique
2. Bibliothèque Swing

III. Programmation événementielle

IV. Architecture MVC

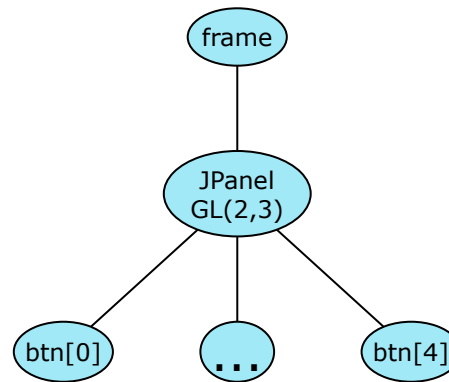
1. Définition
2. Exemple
 - i. Codage du modèle
 - ii. Codage de la vue
 - iii. Codage du contrôleur
 - iv. Organisation du code

V. Composants graphiques

1. Classes de base
2. Méthodes de dessin
3. Définir un composant

VI. Placement des composants graphiques

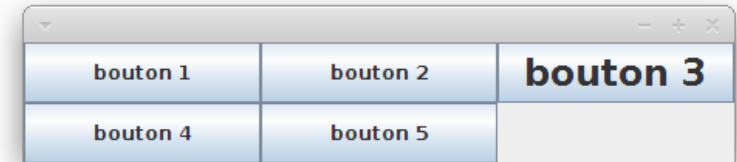
1. Hiérarchies de contenance
 - i. Présentation
 - ii. Visualisation dans le code
 - iii. Technique de codage
 - iv. Affichage des composants
2. Gestionnaires de répartition
 - i. Présentation
 - ii. FlowLayout
 - iii. GridLayout
 - iv. BorderLayout



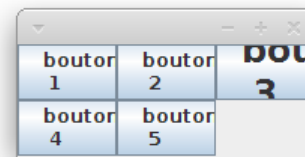
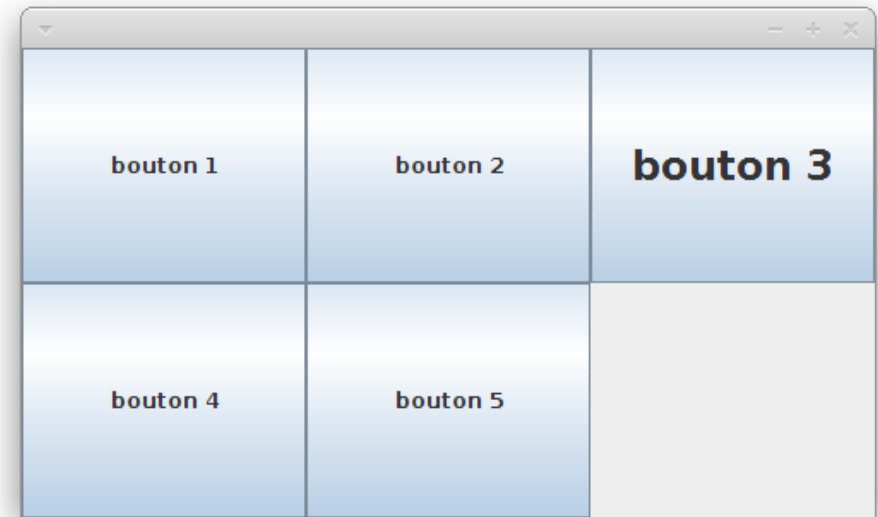
```

JPanel p = new JPanel(new GridLayout(2, 3));
{ //--
    for (JButton b : btn) {
        p.add(b);
    }
} //--
frame.add(p);
    
```

après `frame.pack()`



après retailage à la souris



```

new GridLayout(3, 2)
new GridLayout(3, 0)
new GridLayout(0, 2)
    
```

~~new GridLayout(0, 0)~~

```

new GridLayout(2, 3)
new GridLayout(2, 0)
new GridLayout(0, 3)
    
```


Applications graphiques Java

I. Types imbriqués

1. Définition
2. Type membre statique
3. Classe interne
 - i. Définition
 - ii. Classe membre non stat.
 - iii. Classe locale
 - iv. Classe anonyme
 - v. Étude de cas : itérateurs

II. Présentation

1. Application graphique
2. Bibliothèque Swing

III. Programmation événementielle

IV. Architecture MVC

1. Définition
2. Exemple
 - i. Codage du modèle
 - ii. Codage de la vue
 - iii. Codage du contrôleur
 - iv. Organisation du code

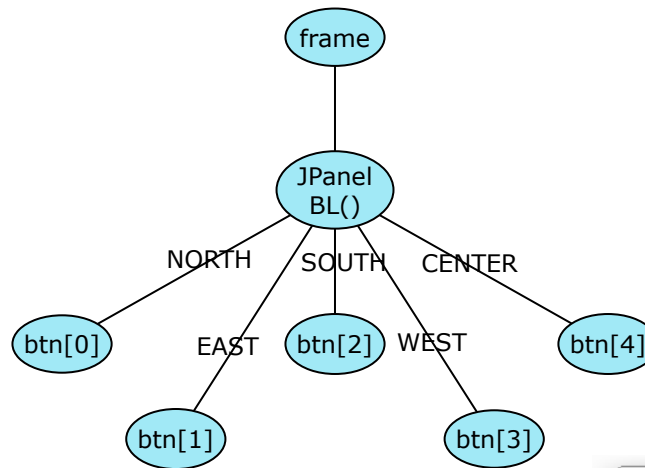
3. Diagramme de classes

V. Composants graphiques

1. Classes de base
2. Méthodes de dessin
3. Définir un composant

VI. Placement des composants graphiques

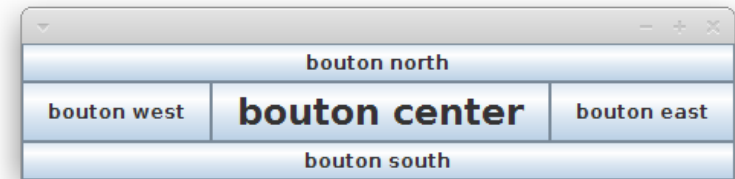
1. Hiérarchies de contenance
 - i. Présentation
 - ii. Visualisation dans le code
 - iii. Technique de codage
 - iv. Affichage des composants
2. Gestionnaires de répartition
 - i. Présentation
 - ii. FlowLayout
 - iii. GridLayout
 - iv. BorderLayout



```

JPanel p = new JPanel(new BorderLayout());
{ //--
    p.add(btn[0], BorderLayout.NORTH);
    p.add(btn[1], BorderLayout.EAST);
    p.add(btn[2], BorderLayout.SOUTH);
    p.add(btn[3], BorderLayout.WEST);
    p.add(btn[4], BorderLayout.CENTER);
} //--
frame.add(p);
    
```

après `frame.pack()`



après retailage à la souris

