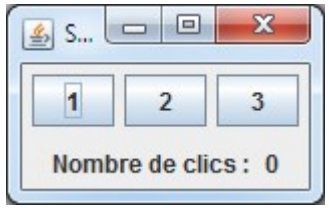
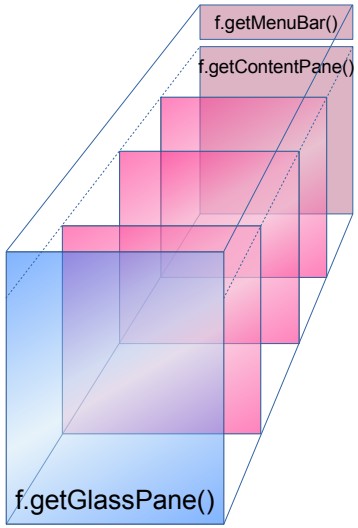
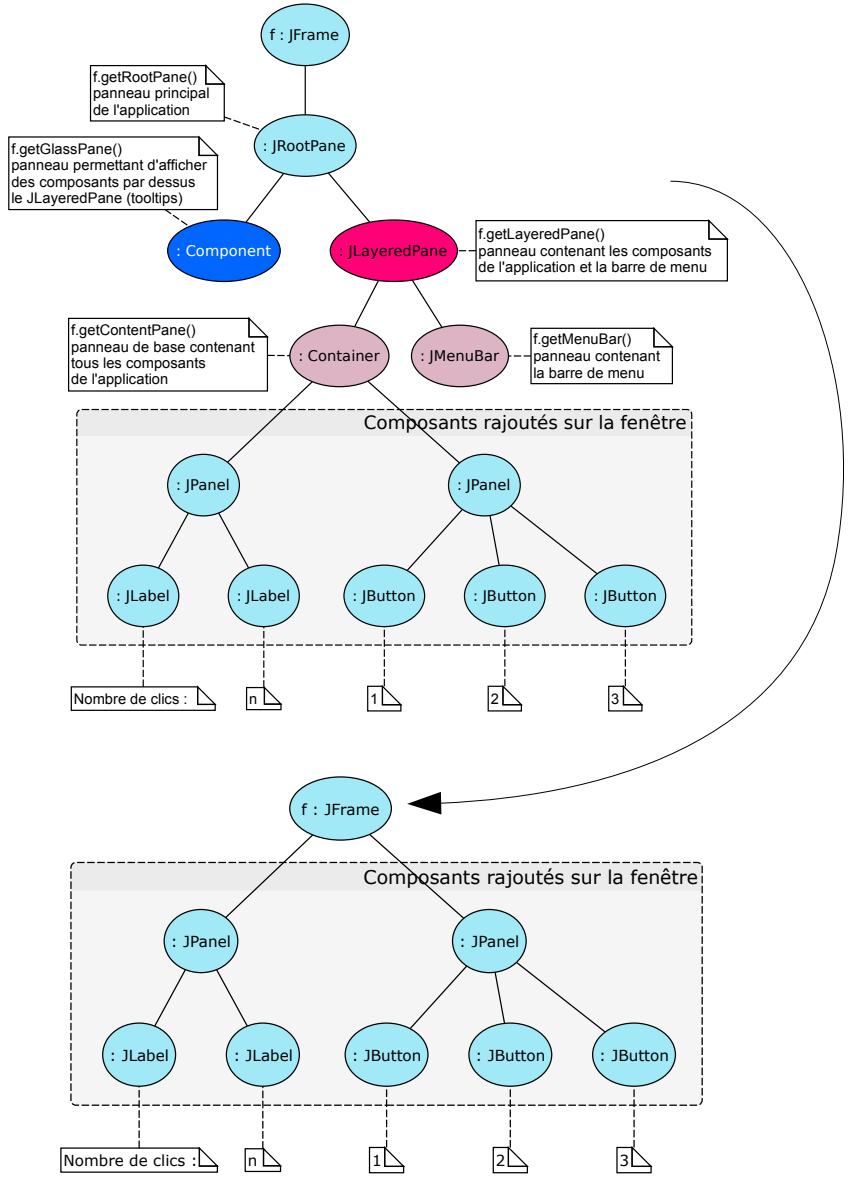
Applications graphiques Java

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





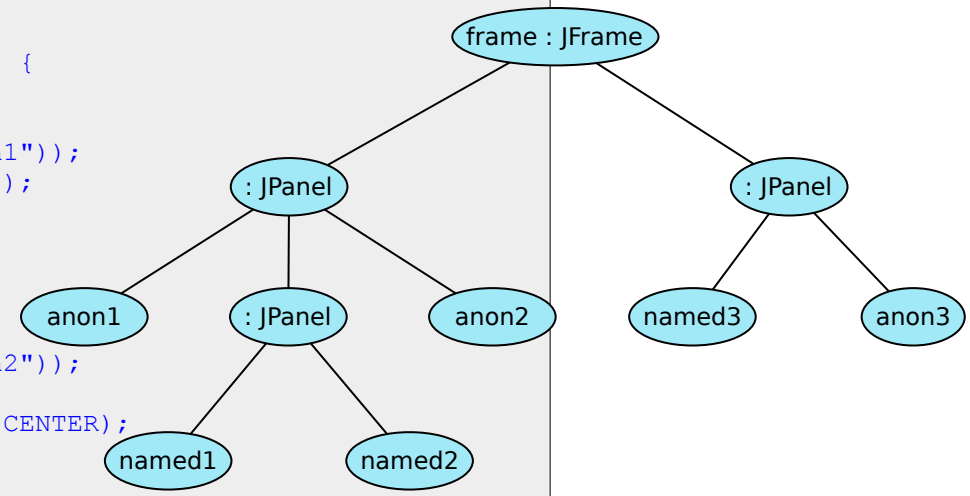


Applications graphiques Java


```
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);
}
```

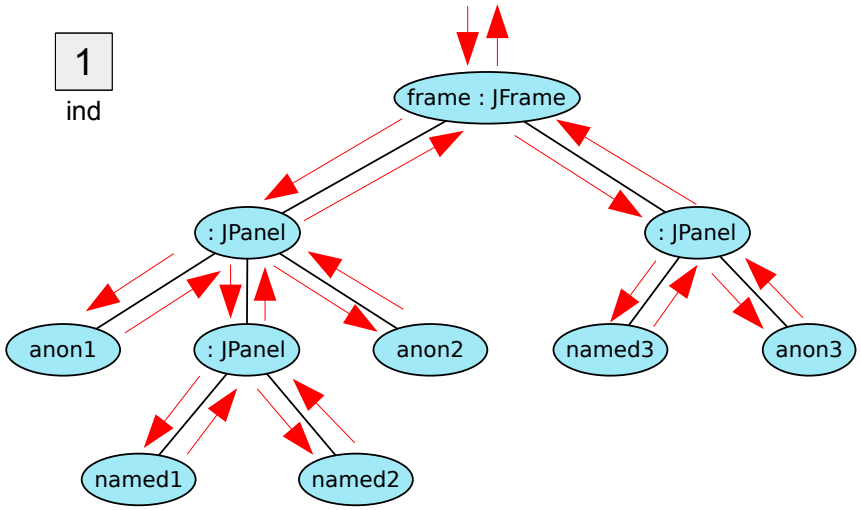


parcours d'une hiérarchie de contenance	arrivée	départ
racine	`private void placeComponents() {` ind ← ind + 1	ind ← ind - 1 `}`

noeud interne	`[JPanel ]p = new JPanel([...]);`
	`{ //--`
	ind ← ind + 1
		ind ← ind - 1
		`} //--`
		`parent.add(p[, ...]);`

feuille	*ne rien faire*	`p.add(c[, ...]);`

```
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);
}
```



Composant graphique affichable : composant dont la hiérarchie de contenance est enracinée dans une fenêtre Swing affichable.

détectable avec `isDisplayable()`

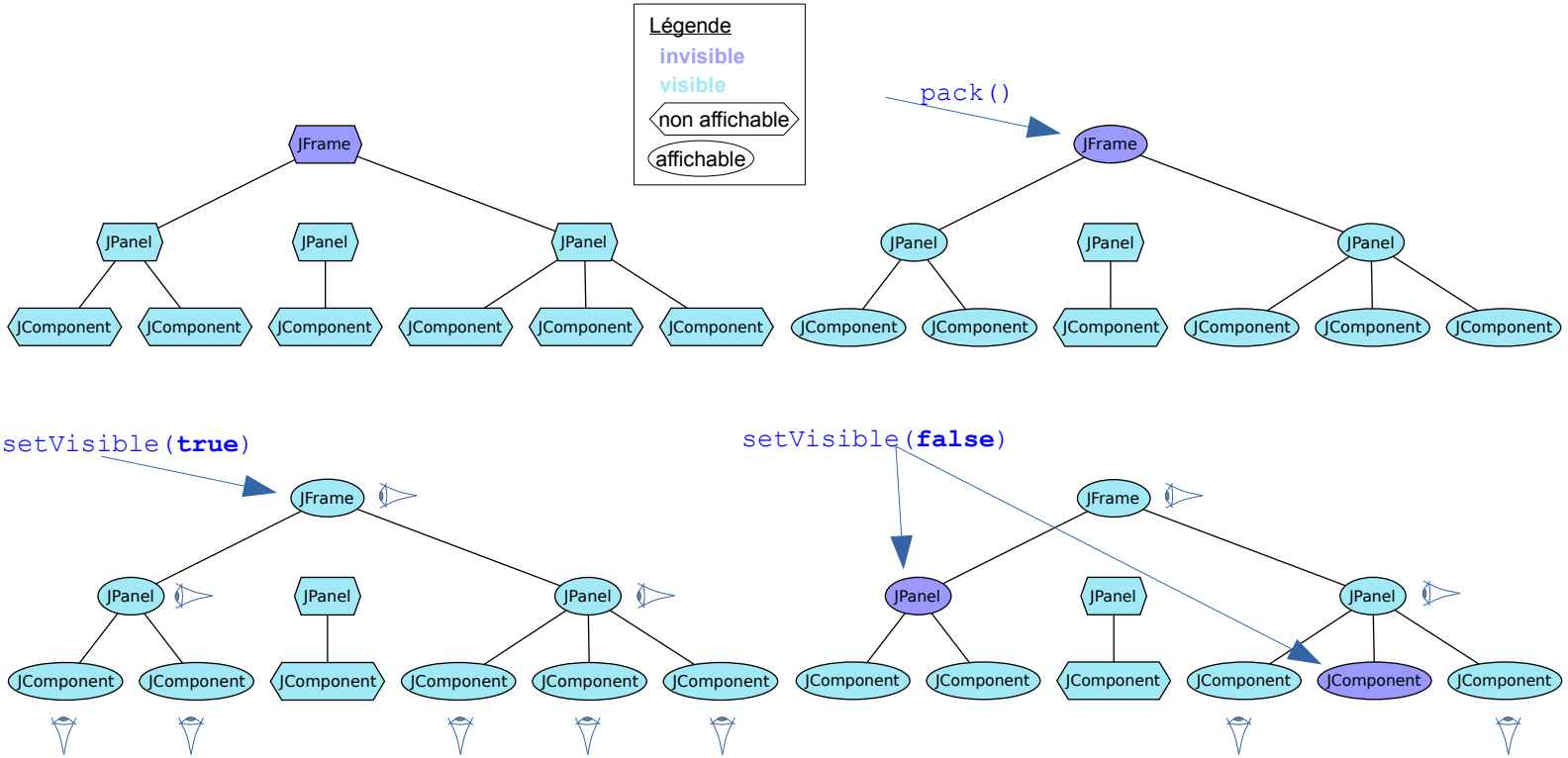
une fenêtre devient affichable après première exécution de `frame.pack()` ou de `frame.setVisible(true)`

Composant graphique visible : composant dont la propriété de visibilité est vraie.

détectable avec `isVisible()`

Composant graphique affiché à l'écran : composant affichable et visible, dont chaque élément de la hiérarchie de contenance est visible.

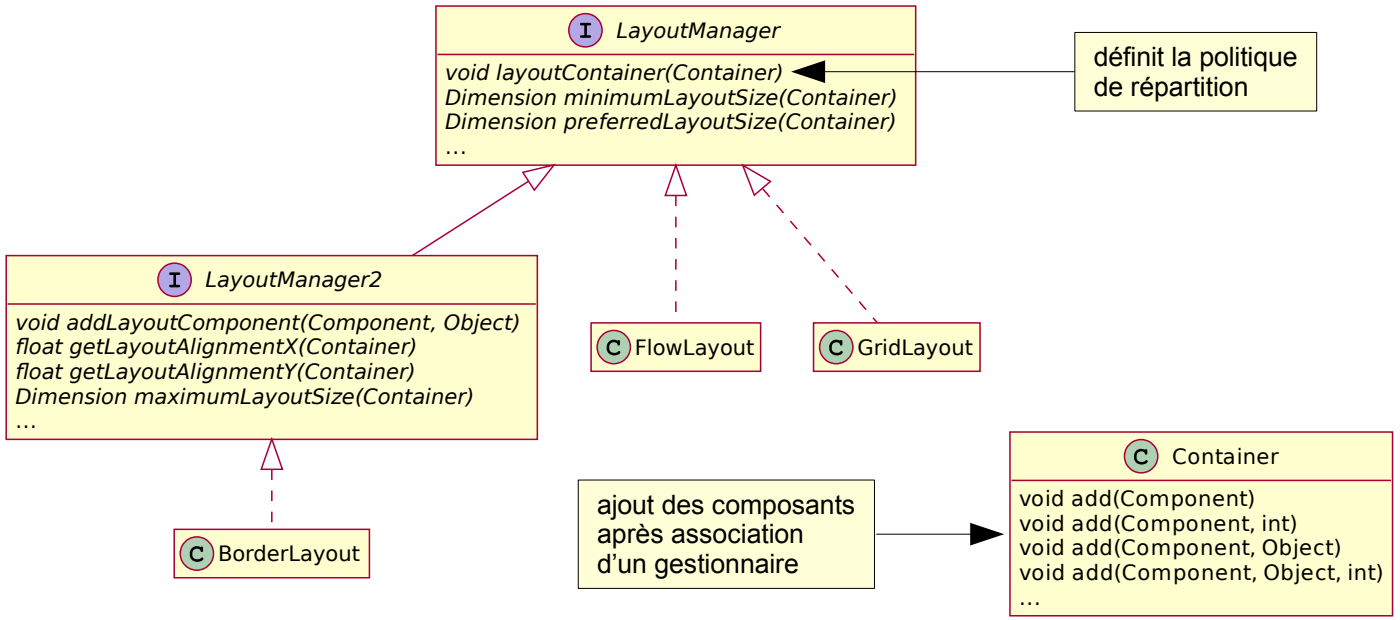
détectable avec `isShowing()`



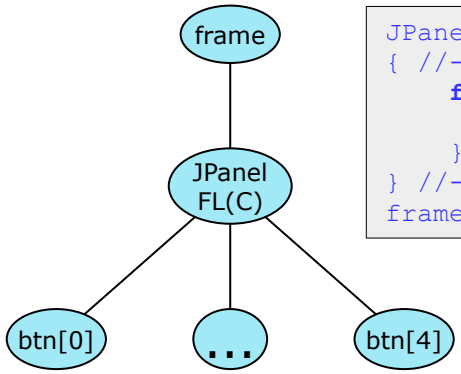
Gestionnaire de répartition :
objet non graphique qui gère intégralement la taille et la place des composants graphiques placés sur un conteneur

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>(...))
```

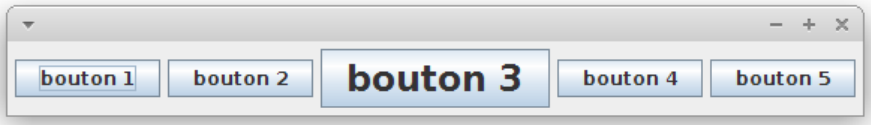


Applications graphiques Java

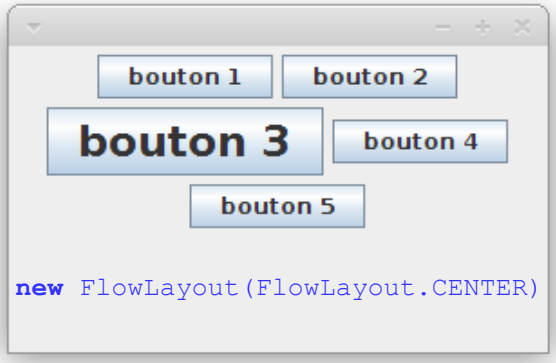



```
JPanel p = new JPanel(new FlowLayout([FlowLayout.CENTER]));
{ //--
    for (JButton b : btn) {
        p.add(b);
    }
} //--
frame.add(p);
```

après `frame.pack()`

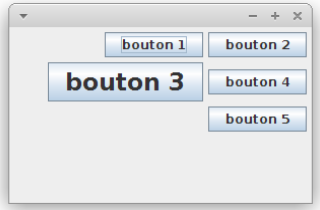


après retaillage à la souris

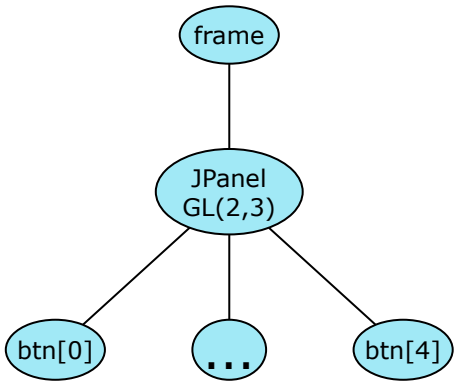


`new FlowLayout(FlowLayout.LEFT)`

`new FlowLayout(FlowLayout.RIGHT)`

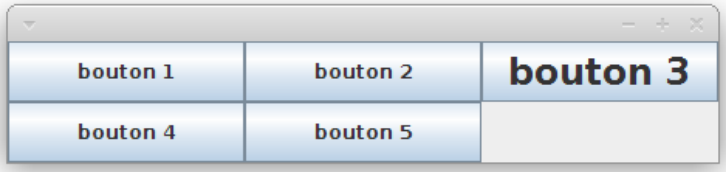


Applications graphiques Java

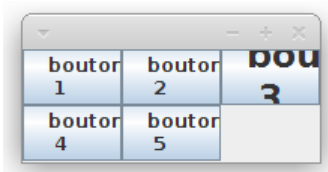



```
JPanel p = new JPanel(new GridLayout(2, 3));
{ //--
    for (JButton b : btn) {
        p.add(b);
    }
} //--
frame.add(p);
```

après `frame.pack()`

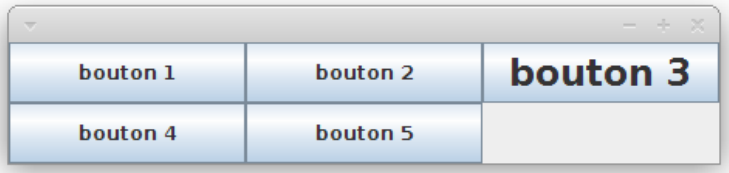


après retaillage à la souris



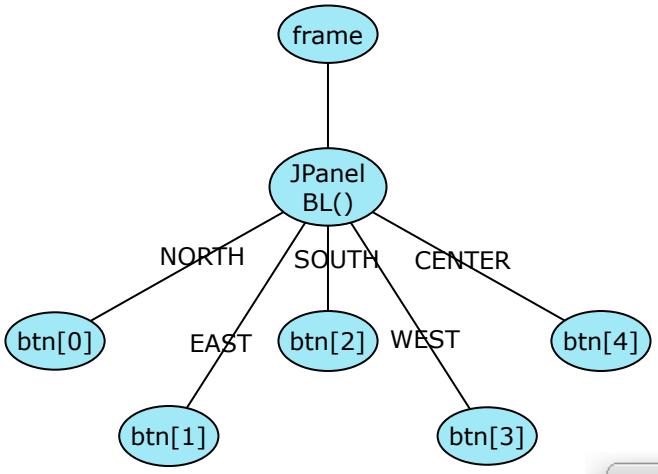

```
new GridLayout(3, 2)
new GridLayout(3, 0)
new GridLayout(0, 2)
```

~~`new GridLayout(0, 0)`~~



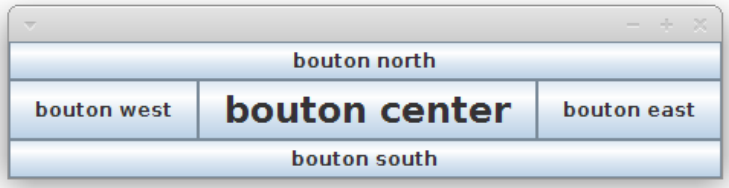

```
new GridLayout(2, 3)
new GridLayout(2, 0)
new GridLayout(0, 3)
```

Applications graphiques Java




```
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 retaillage à la souris