Positionner des composants

Table des matières

Les gestionnaires de positionnement

Tous les composants susceptibles de contenir d'autres composants (par exemple Applet ou Panel) sont dérivés de la classe Container. Ils possèdent une méthode **setLayout** qui permet de définir un gestionnaire de positionnement qui se chargera de placer correctement les composants insérés.

FlowLayout

Les gestionnaires de type FlowLayout se contentent de placer les composants les uns derrière les autres. Ce sont les gestionnaires utilisés par défaut, c'est à dire dans le cas où la méthode setLayout n'a pas été employée.
Le constructeur d'un gestionnaire de type FlowLayout peut utiliser un paramètre indiquant le type d'alignement à utiliser : FlowLayout.RIGHT pour droite, FlowLayout.LEFT pour gauche et FlowLayout.CENTER pour centré qui est l'alignement par défaut.

Exemple de méthode init pour une applet :

```
public void init() {
 setLayout(new FlowLayout(FlowLayout.LEFT));
 add(new Button("Pomme"));
 add(new Button("Poire"));
 add(new Button("Abricot"));
 add(new Button("Pêche"));
 add(new Button("Fraise"));
}
```

BorderLayout

Les gestionnaires de type BorderLayout permettent de placer les composants en faisant référence à une position de type géographique : nord, sud, est, ouest ou centre. Les composants placés au nord, au sud, à l'est et à l'ouest vont prendre leur taille préférée dans un sens et la taille maximale dans l'autre, l'élément placé au centre va occuper tout l'espace restant.

Pour ajouter un composant, on utilisera la méthode add en précisant la position géographique par son nom anglais : North, South, East, West ou Center.

Exemple de méthode init pour une applet :

```
public void init() {
 //on déclare le gestionnaire de positionnement
 setLayout(new BorderLayout());
 //on place les composants
 add("Center", new Button("Centre"));
 add("North", new Button("Nord"));
 add("South", new Button("Sud"));
 add("East", new Button("Est"));
 add("West", new Button("Ouest"));
}
```

GridLayout

Les gestionnaires de type GridLayout permettent de placer les composants dans une grille formée de cases de même taille. Chaque composant prend la taille d'une case. Le constructeur d'un gestionnaire de type GridLayout attend deux paramètres entiers qui sont le nombre de lignes et le nombre de colonnes de la grille.

Pour ajouter des composants on utilise la méthode add. Les composants remplissent alors la grille ligne par ligne.

Exemple de méthode init pour une applet :

```
public void init() {
 setLayout(new GridLayout(2, 3));
 add(new Button("Pomme"));
 add(new Button("Poire"));
 add(new Button("Abricot"));
 add(new Button("Pêche"));
 add(new Button("Fraise"));
 add(new Button("Ananas"));
}
```

CardLayout

Exemple d'applet utilisant un gestionnaire de positionnement de type CardLayout :

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class CardLayoutExemple extends Applet
 implements ActionListener {
 CardLayout carte;
 Panel fruitEtLegume;
 Button bouton=new Button("Fruits ou Légumes");

 public void init() {
  //panel de la première carte (les fruits)
  Panel p1 = new Panel();
  p1.add(new Button("Pomme"));
  p1.add(new Button("Poire"));
  p1.add(new Button("Abricot"));
  p1.setBackground(Color.orange);
  //panel de la deuxieme carte (les legumes)
  Panel p2 = new Panel();
  p2.add(new Button("Tomate"));
  p2.add(new Button("Poireau"));
  p2.add(new Button("Haricot"));
  p2.setBackground(Color.green);
  //panel contenant les deux précédents
  fruitEtLegume=new Panel();
  carte = new CardLayout();
  fruitEtLegume.setLayout(carte);
  fruitEtLegume.add("fruit", p1);
  fruitEtLegume.add("legume", p2);
  setLayout(new BorderLayout());
```

```
 add("North", bouton);
 add("Center", fruitEtLegume);
 bouton.addActionListener(this);
}

public void actionPerformed(ActionEvent e) {
 carte.next(fruitEtLegume);
}
}
```

GridBagLayout

Les gestionnaires de positionnement de type GridBagLayout permettent d'utiliser des tableaux dont les cases n'ont pas toutes les mêmes dimensions. Pour ajouter un composant on devra lui associer un objet de type **GridBagConstraints** qui permet de préciser sa taille et sa position. Cela se fait en deux temps. Pour ajouter un composant c associé à l'objet gbc de type GridBagConstraints avec le gestionnaire g, on écrira :

```
add(c);
g.setConstraints(c,gbc);
```

GridBagConstraints

La classe GridBagConstraints possède les champs suivants qui permettent de préciser la position et la taille d'un composant :

- **gridx** et **gridy** : coordonnées de la cellule; la valeur par défaut est GridBagConstraints.RELATIVE.
- **gridwidth** et **gridheight** : nombre de cellules; la valeur GridBagConstraits.REMAINDER permet le remplissage de toute une ligne et donc un passage à la ligne suivante. occupées en colonne et en ligne; la valeur par défaut est 1.
- **fill** : pour indiquer comment utiliser l'espace disponible lorsque la taille du composant ne correspond pas à celle qui est offerte; on pourra utiliser les constantes suivantes :
  - GridBagConstraints.NONE : pas de redimensionnement; c'est la valeur par défaut.
  - GridBagConstraints.HORIZONTAL : remplir l'espace horizontal.
  - GridBagConstraints.VERTICAL : remplir l'espace vertical.
  - GridBagConstraints.BOTH : remplir tout l'espace offert.
- **ipadx** et **ipady** : espace entre les composants

Un exemple

Pour obtenir l' exemple proposé, on pourra utiliser la méthode init suivante :

```
public void init() {
 //installer le gestionnaire
 GridBagLayout g=new GridBagLayout();
 setLayout(g);
 //créer un objet de type GridBagConstraints
 GridBagConstraints c=new GridBagConstraints();
 //on utilise tout l'espace d'une cellule
 c.fill=GridBagConstraints.BOTH;
 c.weightx=1.0;
 Button peche=new Button("Pêche");
 add(peche);
 g.setConstraints(peche,c);
```

```
 Button poire=new Button("Poire");
 add(poire);
 g.setConstraints(poire, c);
 //on va terminer la ligne avec ce composant
 c.gridwidth=GridBagConstraints.REMAINDER;
 Button pomme=new Button("Pomme");
 add(pomme);
 g.setConstraints(pomme,c);
 //réinitialisation
 c.weightx=0.0;
 c.weighty=1.0;
 c.gridwidth=1;
 c.gridheight=2; //sur 2 lignes
 Button prune=new Button("Prune");
 add(prune);
 g.setConstraints(prune,c);
 //nouvelle réinitialisation
 c.weighty=0.0;
 c.gridwidth=GridBagConstraints.RELATIVE;
 c.gridheight=1;
 Button fraise=new Button("Fraise");
 add(fraise);
 g.setConstraints(fraise,c);
 //on termine la ligne
 c.gridwidth=GridBagConstraints.REMAINDER;
 Button cerise=new Button("Cerise");
 add(cerise);
 g.setConstraints(cerise,c);
 Button ananas=new Button("Ananas");
 add(ananas);
 g.setConstraints(ananas,c);
}
```

Espacer les composants

La méthode getInsets

Par défaut, les composants s'insèrent dans tout l'espace offert par leur conteneur. Il est possible de définir des marges sur les quatre bords en utilisant la méthode getInsets qui renvoie un objet de type **Insets** qui a quatre champs : top, left, bottom et right. Le constructeur des objets de type Insets attend en paramètre les valeurs entières à attribuer à ces champs.

Ainsi, pour que les composants s'insèrent, par exemple dans une applet, en respectant une marge de 4 pixels sur les bords, on ajoutera la méthode suivante :

```
public Insets getInsets() {
 return new Insets(4,4,4,4);
}
```

Espace entre les composants

La plupart des gestionnaires de positionnement admettent des constructeurs qui permettent de définir l'espace horizontal et l'espace vertical qui séparent les composants. Ces deux paramètres sont donnés en dernière place.

Par exemple, pour un gestionnaire de positionnement de type BorderLayout, on imposera un espacement de 4 pixels dans les deux sens en utilisant le code :

```
BorderLayout bl=new BorderLayout(4,4);
```

Pour un gestionnaire de positionnement de type GridLayout qui définit une grille de 4 lignes et 2 colonnes, on écrira:

```
GridLayout gl=new GridLayout(4,2,4,4);
```

Un exemple

On désire construire une applet contenant une zone de saisie et des boutons représentant les 10 chiffres.

Version 1

L'applet se décompose en deux parties : la zone de saisie et l'ensemble des boutons. On peut donc utiliser un gestionnaire de positionnement de type BorderLayout avec la zone de saisie au nord et les boutons au centre.

Pour regrouper les boutons dans un même ensemble on utilise un composant de type **Panel** qui est un dérivé de Container et dont le rôle est donc de contenir d'autres composants. Dans ce Panel, les boutons forment une grille de deux lignes et de cinq colonnes. On utilisera donc un gestionnaire de type GridLayout pour définir cette grille.

Ceci nous donne le code suivant :

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Calcul1 extends Applet {
  TextField saisie;

  public void init() {
   setLayout(new BorderLayout());
   saisie=new TextField(10);
   Panel bas =new Panel();
   bas.setLayout(new GridLayout(2,5));
   ajouteBouton("0",bas);
   ajouteBouton("1",bas);
   ajouteBouton("2",bas);
   ajouteBouton("3",bas);
   ajouteBouton("4",bas);
   ajouteBouton("5",bas);
   ajouteBouton("6",bas);
   ajouteBouton("7",bas);
   ajouteBouton("8",bas);
   ajouteBouton("9",bas);
   add("North",saisie);
   add("Center",bas);

  }
```

```
 void ajouteBouton(String S, Panel p) {
  Button but=new Button(S);
  p.add(but);
 }

}
```

Nous avons créé une méthode ajouteBouton pour regrouper les opérations de création et d'insertion des boutons.

Le résultat obtenu est bien celui attendu, mais les composants sont collés les uns aux autres.

Version 2

Introduisons la méthode getInsets pour obtenir une marge de 10 pixels sur les bords.

```
public Insets getInsets() {
 return new Insets(10,10,10,10);
}
```

Le résultat obtenu a bien une marge, mais les composants sont toujours collés les uns aux autres.

Version 3

Utilisons les constructeurs de gestionnaires de positionnement générant des espaces.

Pour le gestionnaire de type BorderLayout de l'applet, écrivons :

```
setLayout(new BorderLayout(0,5));
```

Pour le gestionnaire de type GridLayout du composant Panel contenant les boutons, écrivons :

```
bas.setLayout(new GridLayout(2,5,5,5));
```

On obtient alors le résultat attendu.

Exercices

1. **Calculette Euro**
   Compléter l'applet créée dans la partie Compléments pour en faire une calculette Euro permettant de convertir des Francs en Euros et des Euros en Francs.

2. **Menu restaurant**
   Créer une applet qui permet de choisir une entrée, une viande et un dessert au restaurant.

3. **GridBagLayout**
   Utiliser un gestionnaire de positionnement GridBagLayout pour reproduire l'applet. Peut-on obtenir le même résultat en n'utilisant que des gestionnaires GridLayout et des Panels ?