TP : Gestion du placement de composants

Objectifs
- se familiariser avec les composants, conteneurs et gestionnaires de placement.

Inscription sur Moodle

Inscrivez-vous au cours d'IHM sur Moodle : http ://moodle.univ-lille1.fr/course/view.php ?id=374 [1] en utilisant la clé d'inscription donnée par votre enseignant de TP. Moodle sera utilisé pour rendre des travaux de TP et vous envoyer des informations tout au long du semestre.

1 Conteneurs de haut niveau : classes `JWindow`, `JFrame`, `JDialog` et `JApplet`

Pour répondre aux questions ci-dessous, vous pouvez vous aider du tutoriel suivant : http ://-java.sun.com/docs/books/tutorial/uiswing/components/frame.html [2] ainsi que des documentations des classes JWindow [3], JFrame [4], JDialog [5] et JLabel [6]

Question 1. En utilisant les classes `JWindow` et `JLabel`, réalisez une interface similaire à celle de la figure 1 (gauche). Utilisez la méthode `getContentPane().add` de `JWindow` pour ajouter le label à la fenêtre. Rendez visible votre fenêtre en utilisant la méthode `setVisible`. Utilisez les méthodes `setSize` pour modifier la taille par défaut de la fenêtre et `setLocation` pour modifier sa position par défaut. Essayez de déplacer et redimensionner la fenêtre en utilisant la souris. Pour quelles applications est-il intéressant d'utiliser la classe `JWindow` ?

Note : pour positionner une fenêtre au centre de l'écran, vous pouvez utiliser la méthode `setLocationRelativeTo(null)` (JFrame). Cette méthode doit-être appelée après la méthode `pack()` pour obtenir un résultat correct. Expliquez pourquoi.

Question 2. Remplacez la classe `JWindow` par la classe `JFrame` de manière à obtenir une interface similaire à la figure 1 (centre). Redimensionnez la fenêtre et observez comment évoluent les positions des textes du label et de la barre de titre.

Question 3. Que se passe-t-il quand on clique sur le bouton de fermeture de l'application ? Utilisez la méthode `setDefaultCloseOperation` de `JFrame` pour quitter l'application quand on clique sur le bouton de fermeture.

Question 4. Utilisez la méthode `setResizable` pour rendre la fenêtre non redimensionable

Question 5. Ajoutez à votre interface précédente une fenêtre utilisant la classe `JDialog` de manière à obtenir une interface similaire à la figure 1 (droite). Vous réaliserez deux versions, l'une en rendant la fenêtre `JDialog` modale avec la `JFrame`, l'autre en version non-modale (notion vue en cours). Dans chaque cas, observez quand il est possible d'interagir avec la fenêtre `JFrame`.

La classe `JApplet` correspond au conteneur de haut niveau utilisé par les applets Java.

Note : si vous voulez utiliser des composants avec du texte contenant des caractères spéciaux (accents par exemple), vous devez compiler votre fichier source avec la commande `javac -encoding ISO8859-1 exemple.java`, si votre fichier source utilise la norme ISO8859-1 pour le formatage des caractères.

1. http://moodle.univ-lille1.fr/course/view.php?id=374
2. http://java.sun.com/docs/books/tutorial/uiswing/components/frame.html
3. http://java.sun.com/javase/6/docs/api/javax/swing/JWindow.html
4. http://java.sun.com/javase/6/docs/api/javax/swing/JFrame.html
5. http://java.sun.com/javase/6/docs/api/javax/swing/JDialog.html
6. http://java.sun.com/javase/6/docs/api/javax/swing/JLabel.html

FIGURE 1 – `JWindow`, `JFrame` et `JDialog`

2 Gestionnaires de placement

Le gestionnaire de placement (layout) a pour rôle de gérer la position des composants d'une interface. Différents gestionnaires de placement existent, chacun représentant une stratégie de placement particulière.

2.1 `BorderLayout`

Ce gestionnaire de placement utilise les points cardinaux pour placer les composants.

Les conteneurs de haut niveau utilisent BorderLayout [7] comme gestionnaire de placement par défaut.

Dans l'exercice précédent, vous avez ajouté le label aux fenêtres en utilisant la méthode `getContentPane().add`. Si vous avez passé le label comme seul argument de la méthode, celui-ci est positionné par défaut au centre du conteneur.

Question 6. En gardant le gestionnaire de placement par défaut, BorderLayout [8], réalisez une interface similaire à la figure 2 gauche. Vous utiliserez ici la classe JButton [9] pour les composants de l'interface. Redimensionnez la fenêtre et observez comment se re-dimensionnent les composants.

Ce gestionnaire de placement est particulièrement utile dans cette situation que l'on retrouve fréquemment : une barre d'état au sud, une barre d'outils au nord, une arborescence à l'ouest et un composant au centre qui prend tout l'espace possible.

2.2 `FlowLayout`

FlowLayout [10] permet de placer les composants les uns à côté des autres, de gauche à droite par défaut.

Question 7. Remplacez le gestionnaire par défaut de `JFrame` par `FlowLayout` en utilisant la méthode `setLayout`. Exemple :

```
JFrame fenetre = new JFrame("JFrame");
fenetre.getContentPane().setLayout(new FlowLayout(FlowLayout.CENTER));
```

Réalisez une interface similaire à celle représentée figure 2 au centre. Testez les différents alignements et orientations possibles. Observez comment évoluent la position et la taille des boutons lors du redimensionnement de la fenêtre.

2.3 `GridLayout`

GridLayout [11] découpe l'espace du conteneur en une grille en remplissant les cases de gauche à droite et de haut en bas.

7. http://java.sun.com/javase/6/docs/api/java/awt/BorderLayout.html
8. http://java.sun.com/javase/6/docs/api/java/awt/BorderLayout.html
9. http://java.sun.com/javase/6/docs/api/javax/swing/JButton.html
10. http://java.sun.com/javase/6/docs/api/java/awt/FlowLayout.html
11. http://java.sun.com/javase/6/docs/api/java/awt/GridLayout.html

Question 8. Réalisez une interface similaire à celle représentée figure 2 à droite en utilisant une grille de 4 lignes et 4 colonnes. Observez comment évoluent la position et la taille des boutons lors du redimensionnement de la fenêtre.

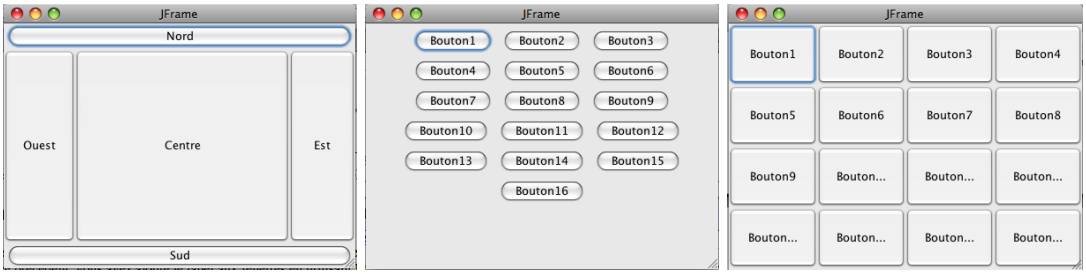

FIGURE 2 – Illustration de `BorderLayout`, `FlowLayout` et `GridLayout`

2.4 `BoxLayout`

BoxLayout [12] permet d'aligner les composants de manière horizontale ou verticale.

Question 9. En utilisant la classe `JFrame` dont vous aurez modifié le gestionnaire de placement pour utiliser `BoxLayout`, réalisez une interface similaire à celle présentée figure 3. Pour obtenir l'espace entre les boutons 2 et 3 vous ajouterez un composant invisible en utilisant la classe interne `Filler`. Vous testerez les différents types de Filler disponibles (rigid area, glue et custom). Etudiez le tutoriel correspondant pour plus d'explications.

Notez que le constructeur de `BoxLayout` prend un objet de type `Container` en premier argument. Pour les conteneurs de haut niveau, la méthode `getContentPane()` permet d'accéder à cet objet.

2.5 `GridBagLayout`

`GridBagLayout` permet la gestion la plus fine (et la plus complexe) des composants. Les composants sont placés sur une grille invisible où chaque composant peut occuper plusieurs lignes et colonnes. Les contraintes sur chaque composant sont spécifiées par l'utilisation de la classe GridBagConstraints [13].

Question 10. (Facultative) Réalisez une interface similaire à celle de la figure 3, en utilisant le gestionnaire de placement GridBag.

2.6 `CardLayout`, `GroupLayout`, `SpringLayout`

CardLayout [14] permet à plusieurs composants d'occuper le même espace à l'écran.
GroupLayout [15] est utilisé par les logiciels de construction d'interfaces comme NetBeans.
SpringLayout [16] est utilisé par les logiciels de construction d'interfaces comme NetBeans. Il est non recommandé de l'utiliser en dehors d'un générateur d'interfaces.

Question 11. (Facultative) Prenez connaissance des tutoriels pour avoir une idée plus précise du fonctionnement de ces gestionnaires de placement.

2.7 Faire sans gestionnaire de placement : positionnement absolu

Le positionnement absolu permet de contrôler la position de chaque composant par rapport aux bords de la fenêtre ainsi que de définir la taille de chaque composant.

Pour utiliser le positionnement absolu, il faut :

12. http://java.sun.com/javase/6/docs/api/javax/swing/BoxLayout.html
13. http://java.sun.com/javase/6/docs/api/java/awt/GridBagConstraints.html
14. http://java.sun.com/javase/6/docs/api/java/awt/CardLayout.html
15. http://java.sun.com/javase/6/docs/api/javax/swing/GroupLayout.html
16. http://java.sun.com/javase/6/docs/api/javax/swing/SpringLayout.html

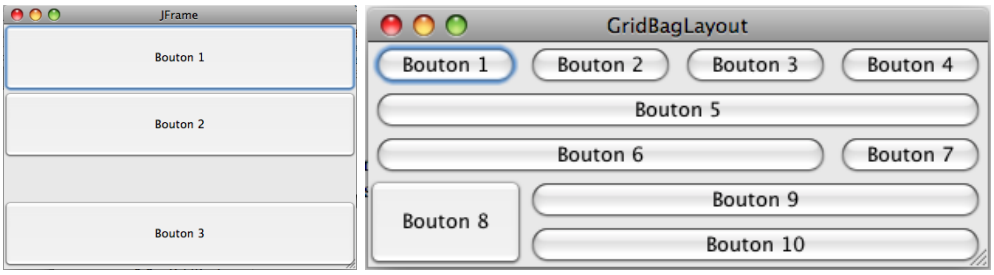

FIGURE 3 – Illustration de `BoxLayout` et `GridBagLayout`

1. Commencer par fixer le gestionnaire de conteneurs à null en appelant `setLayout(null)`
2. Appeler la méthode `setBounds` pour définir la taille et la position de chaque composant
3. Appeler la méthode `repaint()` pour chaque composant

Le réglage de la position de chaque composant se fait en utilisant la méthode `getInsets()` sur le conteneur et le réglage des dimensions se fait en utilisant la méthode `getPreferredSize()` pour chaque composant.

Question 12. Utilisez le positionnement absolu pour créer une interface similaire à celle de la figure 4 (gauche).

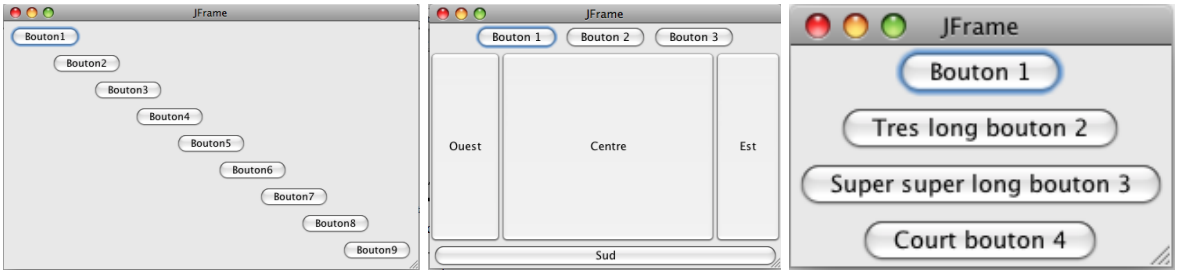

FIGURE 4 – Illustration du positionnement absolu, de l'utilisation de `JPanel` et d'un gestionnaire de positionnement personnalisé.

3 Conteneur de niveau intermédiaire : la classe `JPanel`

JPanel[17] est un conteneur générique qui permet de contenir d'autres composants dont d'autres `JPanel`. Son gestionnaire de placement par défaut est FlowLayout.

Question 13. En reprenant la question 6 sur le `BorderLayout`, remplacez le bouton nord par un JPanel qui contiendra 3 boutons de manière à obtenir une interface similaire à celle de la figure 4.

4 Créer son propre gestionnaire de placement

Nous allons créer un gestionnaire de placement, que nous appelerons `VboxLayout`, qui empile les composants les uns en dessous des autres en les centrant (on peut obtenir un résultat similaire en utilisant un `BoxLayout`).

Question 14. Remplissez les méthodes `setSizes`, `preferredLayoutSize`, `minimumLayoutSize` et `layoutContainer` de la classe VboxLayout.java[18]. Vous testerez `VboxLayout` en utilisant demovbox.java[19] de manière à obtenir une interface proche de celle représentée sur la figure 4.

17. `http://java.sun.com/javase/6/docs/api/javax/swing/JPanel.html`
18. `VboxLayout.java`
19. `demovbox.java`

Pour répondre à cette question, aidez-vous de la démo `DiagonalLayout` du tutoriel.

Question 15. Quelle est l'utilité de la méthode Window.pack [20] ? Comment fonctionne-t-elle ?

5 Pluggable Look-and-feel

Il est possible de changer le look-and-feel de l'ensemble des composants de l'interface en utilisant les instructions suivantes pour utiliser le look-and-feel Motif :

```
try {
        UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
} catch (Exception e) {}
```

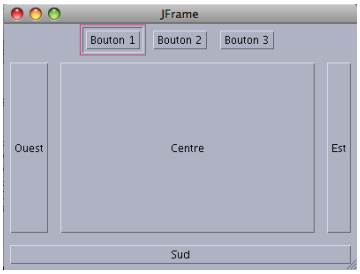

FIGURE 5 – Utilisation du Look-and-Feel Motif.

Question 16. (Facultative) Testez les différents look-and-feel disponibles. (Le look-and-feel Mac OS X n'est pas disponible sous Linux et Windows). Java 7 propose un nouveau Look-and-Feel : Nimbus

6 Un éditeur de texte

Question 17. Proposez une hiérarchie de composants permettant de représenter l'interface ci-dessous.

Question 18. Implémentez la proposition faite à la question précédente.

Pour répondre à cette question, aidez-vous du tutoriel sur la création de menus [21].

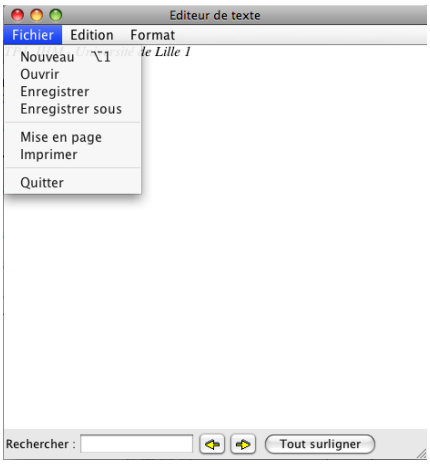

FIGURE 6 – Un éditeur de texte.

20. http://java.sun.com/javase/6/docs/api/java/awt/Window.html
21. http://java.sun.com/docs/books/tutorial/uiswing/components/menu.html

Question 19. Après avoir suivi le tutoriel sur l'utilisation de NetBeans [22], essayez de reproduire à nouveau l'interface en utilisant les outils de création NetBeans.

22. http://java.sun.com/docs/books/tutorial/uiswing/learn/index.html