Utiliser des composants

Table des matières

Les boutons de commande

Comme leur nom l'indique, les boutons de commande sont destinés à lancer une commande (ou une action) lorsque l'utilisateur a cliqué sur leur surface. Grâce à un effet graphique ils donnent l'impression d'avoir été enfoncés, puis relachés.

La classe Button

Les boutons de commande sont représentés par la classe Button du package java.awt. Ils dérivent directement de la classe Component.

```
java.lang.Object
   |
   +----java.awt.Component
           |
           +----java.awt.Button
```

On construit un bouton en indiquant le texte qu'il contiendra. Par exemple, pour obtenir un bouton dont le libellé est "Encore", on écrira :

```
Button b=new Button("Encore");
```

Lorsqu'on appuie sur le bouton on déclenche un évènement de type **ActionEvent**. Cet évènement doit être traité par une classe qui implémente l'interface ActionListener, ce sera pour nous l'applet qui contient le bouton. Pour définir la classe chargée de réagir à l'évènement de type ActionEvent généré par le bouton, on utilise la méthode **addActionListener(ActionListener)**. Enfin pour caractériser l'action que doit déclencher le bouton par une chaîne de caractères on utilise la méthode **setActionCommand(String)**. Si on ne le fait pas, c'est le libellé du bouton qui est utilisé.

Insertion dans une applet

Créons une applet qui contient un bouton "Ajouter" et qui affiche le nombre 0. A chaque fois qu'on appuie sur le bouton "Ajouter", le nombre affiché doit augmenter d'une unité.

Déclaration de l'applet

Comme l'applet sera chargée de réagir aux évènements générés par le bouton, elle devra implémenter l'interface ActionListener qui contient la méthode **actionPerformed(ActionEvent evt)**. La classe ActionEvent contient la méthode **getActionCommand** qui renvoie la chaîne qui caractérise l'action à réaliser.
On obtient :

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

public class bouton1 extends Applet
 implements ActionListener {
```

```
 int nombre;

 public void init() {
  nombre=0;
 }

 public void actionPerformed(ActionEvent evt) {
 }

 public void paint(Graphics g) {
  g.drawString(""+nombre,50,80);
 }

}
```

Insertion du bouton

L'insertion du bouton dans l'applet aura lieu dans la méthode init. Il y a trois opérations à réaliser : créer le bouton, l'insérer, indiquer que la commande qu'il génère sera prise en compte par l'applet. Cela donne :

```
public void init() {
 nombre=0;
 //création
 Button b = new Button("Ajouter");
 //insertion
 add(b);
 //choix de l'écouteur
 b.addActionListener(this);
}
```

Réaction aux clics sur le bouton

La réaction aux clics sur le bouton se fait dans la méthode actionPerformed. On vérifie l'origine de la commande et on l'exécute.

```
public void actionPerformed(ActionEvent evt) {
 if (evt.getActionCommand().equals("Ajouter")) {
  //on augmente nombre d'une unité
  nombre++;
  //on réaffiche
  repaint();
 }
}
```

Test

On obtient l'applet suivante :

Remarques :
1- la méthode paint de l'applet ne fait pas référence au bouton, celui-ci s'affiche automatiquement.
2- nous n'avons à aucun moment précisé la position du bouton, celle-ci est aussi déterminée de façon automatique.

Avec 2 boutons et des couleurs

Complétons l'applet précédente avec un bouton "Retirer" qui enlèvera une unité au nombre affiché et qui utilisera des couleurs. Nous ajouterons un champ fonte utilisé pour l'affichage et nous complèterons les trois méthodes init, paint et actionPerformed.

Méthode init

Après avoir choisi des couleurs pour l'applet, nous ajoutons les boutons.

```
public void init() {
 //pour l'applet
 nombre=0;
 fonte=new Font("Monospace",Font.BOLD,20);
 setBackground(Color.yellow);
 setForeground(Color.blue);
 // ajout du 1er bouton
 Button b = new Button("Ajouter");
 add(b);
 b.setActionCommand("plus");
 b.addActionListener(this);
 b.setBackground(new Color(0,128,0));
 b.setForeground(Color.yellow);
 // ajout du 2ème bouton
 b=new Button("Retirer");
 add(b);
 b.setActionCommand("moins");
 b.addActionListener(this);
 b.setBackground(new Color(128,0,0));
 b.setForeground(Color.yellow);
}
```

Gestion des évènements

La chaîne caractérisant les actions nous permet de déterminer la conduite à adopter.

```
public void actionPerformed(ActionEvent evt) {
 if (evt.getActionCommand().equals("plus")) {
  //c'est le premier bouton
  nombre++;
  repaint();
 }
 else if (evt.getActionCommand().equals("moins")) {
  //c'est le deuxième bouton
  nombre--;
  repaint();
 }
}
```

Méthode paint

Il suffit d'activer la fonte choisie avant d'afficher le nombre.

```
public void paint(Graphics g) {
 g.setFont(fonte);
 g.drawString(""+nombre, 50, 80);
```

Conversion Francs/Euros

Nous allons créer une applet qui pourra convertir une somme en Francs en Euros.

Version 1

Déclarations

L'applet utilise :

- une zone de texte fixe **(classe Label)** pour le titre
- une zone de texte fixe "Somme en Francs"
- une ligne d'édition **(classe TextField)** pour entrer la somme en Francs
- une zone de texte fixe "Equivalent en Euros"
- une ligne d'édition où s'inscrira la somme en euros
- un bouton "Convertir"

Nous allons créer des champs "franc" et "euro" représentant les lignes d'édition pour pouvoir les utiliser lors de l'appui sur le bouton "Convertir".

L'applet sera l'écouteur associé au bouton, elle devra implémenter l'interface ActionListener.

Cela nous donne le code suivant :

```
public class Euro extends java.applet.Applet
 implements ActionListener {

 static double valEuro=6.55957;
 TextField franc;
 TextField euro;
```

Initialisation

La méthode init va nous permettre de définir la couleur de fond, d'insérer les différents composants et de définir l'écouteur des évènements générés par le bouton.

```
 public void init() {
  Label lab;
  //couleur du fond
  setBackground(Color.lightGray);
  //titre
  lab=new Label("CONVERSION FRANCS/EUROS");
  lab.setFont(new Font("SansSerif",Font.BOLD,16));
  add(lab);
  //label et ligne d'édition pour les francs
  lab=new Label("Somme en Francs  : ");
  lab.setFont(new Font("Monospaced",Font.PLAIN,12));
  add(lab);
  franc=new TextField(10);
  add(franc);
  //label et ligne d'édition pour les euros
  lab=new Label("Equivalent Euros : ");
  lab.setFont(new Font("Monospaced",Font.PLAIN,12));
  add(lab);
```

```
        euro=new TextField(10);
        euro.setEditable(false);
        euro.setBackground(Color.yellow);
        add(euro);
        //bouton
        Button bouton=new Button("Convertir");
        bouton.addActionListener(this);
        add(bouton);
       }
```

Action

Il nous reste ensuite à écrire la méthode actionPerformed qui sera exécutée lors de l'appui sur le bouton "Convertir". Nous utiliserons les méthodes **getText** et **setText** de la classe TextField pour lire et écrire dans les lignes d'édition.

```
    public void actionPerformed(ActionEvent e) {
     if (e.getActionCommand().equals("Convertir")) {
       Double FR=new Double(franc.getText());
       double EU=FR.doubleValue()/valEuro;
       //pour arrondir à 2 chiffres
       EU=(Math.floor(EU*100+0.5))/100;
       euro.setText(""+EU);
     }
    }
```

Test

Testons l'applet en lui donnant différentes tailles.

Nous pouvons constater que le positionnement des différents composants dépend de la largeur de l'applet. C'est normal : comme nous n'avons donné aucun renseignement sur la position des composants, ils se mettent l'un derrière l'autre de façon automatique.

Version 2

Dans cette version nous allons apporter les améliorations suivantes :
- regrouper les objets de type Label et de type TextField associés dans des objets de type **Panel**; cela évitera des changements de ligne non désirés.
- ajouter des couleurs
- provoquer le calcul lorsqu'on appuie sur la touche Entrée après avoir écrit la somme en Francs.
- donner le focus à la ligne d'édition de la somme en francs dès le démarrage de l'applet.

Utiliser des objets Panel

L'utilisation des objets Panel pour regrouper des composants et la définition des couleurs se feront dans la méthode init qui deviendra :

```
    public void init() {
     Label lab;
     Panel pan;
     //couleur du fond
     setBackground(Color.blue);
     //titre
```

```
 lab=new Label("CONVERSION FRANCS/EUROS");
 lab.setFont(new Font("SansSerif",Font.BOLD,16));
 lab.setForeground(Color.yellow);
 add(lab);
 //label et ligne d'édition pour les francs dans un panel
 pan=new Panel();
 lab=new Label("Somme en Francs  : ");
 lab.setFont(new Font("Monospaced",Font.BOLD,12));
 lab.setForeground(Color.white);
 pan.add(lab);
 franc=new TextField(10);
 pan.add(franc);
 add(pan);
 //label et ligne d'édition pour les euros dans un panel
 pan=new Panel();
 lab=new Label("Equivalent Euros : ");
 lab.setFont(new Font("Monospaced",Font.BOLD,12));
 lab.setForeground(Color.white);
 pan.add(lab);
 euro=new TextField(10);
 euro.setEditable(false);
 euro.setBackground(Color.yellow);
 pan.add(euro);
 add(pan);
 //bouton
 Button bouton=new Button("Convertir");
 bouton.addActionListener(this);
 add(bouton);
}
```

La méthode utilisée est la suivante :
- on crée un objet de type Panel
- on insère des objets Label et TextField dans ce panel
- on insère l'objet Panel dans l'applet.

Réaction à la touche Entrée

Les objets de type TextField provoquent un évènement de type ActionEvent lorsqu'on appuie sur la touche Entrée. Pour récupérer cet évènement, il nous suffira d'installer un écouteur. Cela se fera en ajoutant la ligne :

```
franc.addActionListener(this);
```

dans la méthode init de l'applet. C'est l'applet qui sera l'écouteur. On doit donc compléter sa méthode actionPerformed en tenant compte du fait que l'évènement de type ActionEvent peut être généré par la ligne d'édition franc. Nous allons utiliser la méthode getSource() de l'évènement, elle nous indique quel objet l'a généré. La méthode actionPerformed devient:

```
public void actionPerformed(ActionEvent e) {
 if (e.getActionCommand().equals("Convertir") ||
     e.getSource()==franc) {
   Double FR=new Double(franc.getText());
   double EU=FR.doubleValue()/valEuro;
   EU=(Math.floor(EU*100+0.5))/100;
   euro.setText(""+EU);
 }
}
```

Demande du focus

Pour être immédiatement utilisable, la ligne d'édition de la somme en Francs doit demander le focus. Cela peut se faire dans la méthode **start** de l'applet. Cette méthode est appelée automatiquement dès que l'initialisation de l'applet est terminée.

Il suffira d'écrire :

```
public void start() {
 franc.requestFocus();
}
```

Boutons radio et cases à cocher

Boutons radio et cases à cocher sont représentés par la classe **CheckBox**. Des objets de type CheckBox regroupés dans un objet de type **CheckBoxGroup** donneront des boutos radio. Par contre, isolés ils donneront des cases à cocher.

Un exemple

Ecrivons une applet qui affiche un cercle. Un groupe de boutons radio permet de choisir le rayon (petit, moyen ou grand) et une case à cocher permet de déterminer si le cercle est plein ou non.

L'interface ItemListener

Les changements d'état des objets de type CheckBox génèrent un évènement de type **ItemEvent**. On définit un écouteur de ce type d'évènement en utilisant l'interface **ItemListener**. Celle-ci impose la présence de la méthode :

```
public void itemStateChanged(ItemEvent e) {
}
```

Notre applet sera l'écouteur associé aux différents onbjets de type CheckBox.

Les champs de l'applet

Pour pouvoir vérifier l'état des boutons radio et de la case à cocher, nous allons créer des champs les contenant.

Ceci nous donne la déclaration suivante :

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

public class BoutonRadio extends Applet
 implements ItemListener {
  // le groupe des boutons radio
  CheckboxGroup groupeRadio = new CheckboxGroup();
  Checkbox petit = new Checkbox("petit",groupeRadio,false);
```

```
Checkbox moyen = new Checkbox("moyen",groupeRadio,true);
Checkbox grand = new Checkbox("grand",groupeRadio,false);
// la case à cocher
Checkbox disque = new Checkbox("disque",false);
```

Notons que le bouton "moyen" est sélectionné grâce au paramètre true, les autres ne le sont pas.

L'initialisation

Nous utiliserons la méthode init pour définir les couleurs utilisées, insérer les objets de type CheckBox et définir leurs écouteurs. Cela donne :

```
public void init() {
 setBackground(Color.yellow);
 setForeground(Color.red);
 add(petit);
 add(moyen);
 add(grand);
 add(disque);
 petit.addItemListener(this);
 moyen.addItemListener(this);
 grand.addItemListener(this);
 disque.addItemListener(this);
}
```

La méthode paint

Avant de dessiner le cercle nous allons tester les boutons radio pour déterminer le rayon. Leur méthode **getState()** indique s'ils sont cochés ou non. Lorsque le rayon est déterminé, nous calculons les coordonnées permettant d'afficher le cercle au centre. Enfin, l'état de la case "plein" nous indique s'il faut dessiner un cercle plein ou non. On obtient le code suivant :

```
public void paint(Graphics g) {
 int rayon=0;
 if (petit.getState()) rayon=10;
 else if (moyen.getState()) rayon=50;
 else if (grand.getState()) rayon=100;
 int x=(getSize().width-2*rayon)/2;
 int y=(getSize().height-2*rayon)/2;
 if (disque.getState())
  g.fillOval(x,y,2*rayon,2*rayon);
 else
  g.drawOval(x,y,2*rayon,2*rayon);
}
```

L'écoute des évènements

Il nous reste à écrire la méthode itemStateChanged qui est applelée lors d'un changement d'état des objets de type CheckBox. Il nous suffira de faire un appel à la méthode repaint de l'applet, puisque la méthode paint tient compte de l'état des boutons. Ceci nous donne :

```
public void itemStateChanged(ItemEvent e) {
 repaint();
}
```

Exercices

1. **Déplacer un texte**
   Créer une applet qui affiche un texte et qui contient deux boutons dont l'effet est de déplacer le texte vers la droite ou vers la gauche.

2. **Cercle de rayon donné**
   Créer une applet qui devra dessiner un cercle de rayon fixé par l'utilisateur. Elle contiendra un bouton "Trace" et une zone de saisie permettant d'entrer le rayon.

3. **Cercle ou carré**
   Créer une applet qui affichera un cercle ou uncarré selon le choix de l'utilisateur qui se fera grâce à deux boutons radio.

4. **Cercle ou carré (bis)**
   Reprendre l'applet précédente en utilisant un composant de type Choice pour permettre à l'utilisateur de choisir entre cercle et carré.

5. **Jeu de Rochambeaux**
   Simuler le jeu de Rochambeaux : le joueur et l'ordinateur choisissent "Pierre", "Papier" ou "Ciseaux". "Papier" gagne sur "Pierre", "Pierre" gagne sur "Ciseaux" et "Ciseaux" gagne sur "Papier".