Programmation Orientée Objet
TP 3 : programmation événementielle.

https ://dpt-info.u-strasbg.fr/~kvanhoey

Introduction

Les programmes que vous avez l'habitude d'écrire sont des programmes séquentiels : on commence dans une méthode *main* puis on exécute une série d'instructions.

Lorsqu'on programme des interfaces graphiques interactives, l'approche séquentielle n'est pas bien adaptée. En effet, l'exécution du programme va dépendre étroitement des actions d'un utilisateur. Le programme devra alors réagir à des événements (clic de souris, entrée au clavier, changement de taille de fenêtre). L'approche événementielle est alors adoptée. ***En programmation événementielle, vous ne contrôlez pas l'exécution du programme. En fait, vous attendez que l'utilisateur fasse une action qui provoquera automatiquement l'appel d'une méthode que vous avez écrit.***

Dans ce TP, nous allons voir comment faire réagir des applets simples à des événements tels que les manipulations à la souris et les frappes au clavier. On parle de gestion des événements, et cela se fait en java en implémentant des interfaces d'écouteurs du package `java.awt.event`. N'oubliez pas d'aller voir l'API pour plus d'informations sur les méthodes des classes utilisées.

Rappel/Indications :

- Pour écrire une applet, une classe doit hériter de la classe `javax.swing.JApplet`. Le cycle de vie d'une applet commence avec l'appel aux méthodes *void init()* pour initialiser les paramètres de l'applet, *void start()* pour démarrer l'applet et *void paint(Graphics g)* pour dessiner le contenu de l'applet. Ces méthodes peuvent être redéfinies pour ajouter des fonctionnalités à chacune des phases correspondantes.
- N'oubliez pas d'écrire une petite page `html` qui appelle votre applet, puis utilisez **appletviewer** pour la lancer.
- Les interfaces s'utilisent avec le mot-clé **implements** et une classe peut implémenter plusieurs interfaces en écrivant leurs noms à la suite, séparés par des virgules.

1 Applet avec MouseListener

Pour réagir aux clics sur les boutons de la souris, on doit implémenter l'interface `MouseListener`. Cette interface contient les profiles suivants :

```
void mouseClicked(MouseEvent e)
void mousePressed(MouseEvent e)
void mouseReleased(MouseEvent e)
void mouseEntered(MouseEvent e)
void mouseExited(MouseEvent e)
```

La classe `MouseEvent` possède notamment des méthodes *int getX()* et *int getY()* qui seront utiles par la suite. Par conséquent, chacune des ces méthodes doit être définie dans la classe qui implémente cette interface (même si le corps de la méthode reste vide) pour que la compilation ne provoque pas d'erreurs. Pour signifier à une applet d'écouter les événements provenant des boutons de la souris, on peut redéfinir sa méthode *init* comme suit :

```
void init()
{
    addMouseListener(this) ;
}
```

1. Écrivez une applet qui « écoute » les clics sur la souris. à chaque clic sur un bouton, affichez les coordonnées du point sur lequel le pointeur de la souris se trouve.
2. En utilisant les méthodes de dessin de la classe `Graphics`, modifiez l'applet pour que l'on puisse dessiner des traits à la souris. Pour dessiner un trait, on clic en maintenant le bouton enfoncé pour donner la première extrêmité, on se déplace puis on relâche le bouton pour définir la deuxième extrémité. On se sert de la méthode *void repaint()* pour rappeler la méthode *paint* et donc mettre à jour le dessin de l'applet.
3. Faire de même, mais cette fois on veut pouvoir dessiner des rectangles, puis des ovales. Est-ce que cela fonctionne quelque soit la position de la deuxième extrêmité. Si ce n'est pas le cas, modifiez le programme pour que ces formes soient dessinées quelque soit le sens du click & drag.

2 Sensibilité au mouvement de la souris

L'interface `MouseMotionListener` donne le profil des méthodes suivantes :

```
void mouseDragged(MouseEvent e)
void mouseMoved(MouseEvent e)
```

Implémenter cette interface pour afficher la position de la souris non plus au clic, mais dès qu'elle est en mouvement sans bouton enfoncé.

3 Saisie des touches

L'interface `KeyListener`, qui est utilisée pour écouter les frappes au clavier, donne le profil des méthodes suivantes :

```
void keyPressed(KeyEvent e)
void keyReleased(KeyEvent e)
void keyTyped(KeyEvent e)
```

Utilisez cette interface pour ajouter des fonctionnalités à votre applet :
- `1` : le click & drag dessine des traits ;
- `2` : le click & drag dessine des ovales ;
- `3` : le click & drag dessine des carrés ;
- `f` : mode fill, dessine ou non les formes avec remplissage ;
- `j,v,b,r,n` : mode couleur, dessine en jaune, vert, bleu, rouge, noir ;
- `c` : mode coordonnées, affiche ou non les coordonnées de la souris pendant son déplacement ;
- `h` : mode help, donne la liste des touches et les actions qui y sont associées ;

4 Historique des dessins

Vous aurez peut-être remarqué que si une partie de la fenêtre de l'applet est recouverte, le dessin qui s'y trouve est effacé. On veut maintenant ajouter un mécanisme de sauvegarde des éléments dessinés dans un tableau (ou plutôt une pile).

1. Écrivez une classe supplémentaire pour représenter et stocker dans un tableau les données nécessaires aux différents dessins vus dans ce TP.

   Question : quel est le terme désignant la technique utilisée ici ?
2. Associez un objet de cette classe à votre applet pour permettre l'ajout des informations à chaque fois qu'un événement débouchant sur un dessin a lieu. L'affichage de l'applet consiste alors à dessiner l'ensemble des figures sauvegardées dans ce tableau.
3. Enfin, ajoutez une action permettant de supprimer le dernier élément sauvegardé du tableau pour permettre d'effacer un ou plusieurs dessins de l'historique.

5 Classe `Vector`

Nous avons jusque là utilisé un tableau classique pour stocker les formes. Les tableaux classiques ne sont pas très pratiques car ils ne peuvent pas se redimensionner. Nous allons utiliser à sa place la classe `Vector` (du package `java.util`) fourni par l'API Java.

1. Regardez dans la documentation de la classe `Vector`. En particulier, vous allez avoir besoin des méthodes *add*, *size* et *get*.
2. Remplacez le tableau de votre programme par un `Vector`. Attention, la méthode get retournant une référence vers un `Object`, il faudra faire un cast.