hoenen@icps.u-strasbg.fr

TP4

Dans ce TP, nous allons voir comment faire réagir des applets simples à des évènement tels que les manipulations à la souris et les frappes au clavier. On parle de gestion des évènements, 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.

Rappels :

- les interfaces s'utilisent avec le mot-clé `implements` et une classe peut implémenter plusieurs interface en écrivant leurs noms à la suite séparés par des virgules.
- 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 l'utiliser.

Applet avec `MouseListener`

Pour pouvoir réagir aux clicks sur les boutons de la souris, on doit implémenter l'interface `MouseListener`. Cette interface contient le profile des méthodes 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 dans 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 par d'erreur. Pour dire à une applet d'écouter les évènements 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 clicks sur la souris. À chaque click 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 click en maintenant le bouton enfoncé pour donner la première extrêmité, puis on se déplace et 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 extrêmitée. Si ce n'est pas le cas, modifiez le programme pour que ces formes soient dessinées quelque soit le sens du *click* & *drag*.

D'autres écouteurs d'énèvements

1. L'interface `MouseMotionListener` donne le profile des méthodes suivantes :

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

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

2. L'interface `KeyListener`, qui est utilisée pour écouter les frappes au clavier, donne le profile 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 rectangles
- `3` : le *click* & *drag* dessine des ovales
- `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

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 se voit effacé. On veut maintenant ajouter un mécanisme de sauvegarde des éléments dessiné 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.
2. Associez un objet de cette classe à votre applet pour permettre l'ajout des informations à chaque fois qu'un évènement débouchant sur un dessin a lieu. Le dessin de l'applet consistera 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.