Université de Nice - Sophia Antipolis
Faculté des Sciences

DEUG MIAS MP1

Programmation 2000-01

10. DESSINE MOI UN RECTANGLE

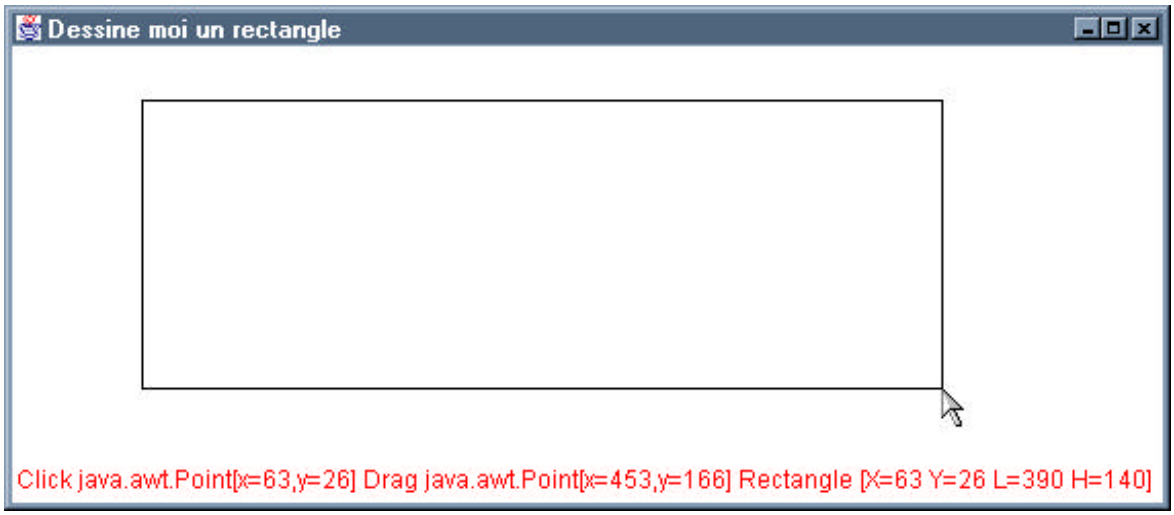
A. Les MouseListener

Nous allons voir dans ce TP comment il est possible de faire réagir une interface graphique aux actions de la souris sur des composants graphiques autres que des boutons ou les petits contrôles des fenêtres. Java utilise des objets appelés `MouseListener` dérivant de `EventListener` exactement comme les `WindowListener` que vous avez vus dans le TP 7. Le `WindowListener` s'occupe d'écouter les événements générés par une fenêtre (décrits par des objets `WindowEvent`) alors que le `MouseListener` écoute les événements générés par la souris (décris par des objets `MouseEvent`). Dans ce cas aussi vous pouvez remarquer deux classes abstraites utiles :

```
public abstract class MouseMotionAdapter implements MouseMotionListener
{
    void mouseDragged(MouseEvent e) // invoquée lorsque la souris est déplacée
                                                              avec le bouton enfoncé
    void mouseMoved(MouseEvent e)   // invoquée lorsque la souris est déplacée
                                                          sans que le bouton soit enfoncé
}

public abstract class MouseAdapter implements MouseListener
{   public void mouseClicked(MouseEvent e) {};  // click de souris
    public void mouseEntered(MouseEvent e) {};  // la souris entre dans le composant
    public void mouseExited(MouseEvent e) {};   // la souris sort du composant
    public void mousePressed(MouseEvent e) {};  // le bouton est enfoncé
    public void mouseReleased(MouseEvent e) {}; // le bouton est relâché
}
```

Nous allons explorer la gestion des événements en écrivant un programme permettant de dessiner un rectangle noir dans un `Canvas` blanc, et qui affiche ensuite les coordonnées de ce rectangle dans un `Label`. Le premier coin du rectangle sera donné par la position de la souris lorsque l'on va appuyer sur le bouton de la souris. Le coin opposé sera fixé lorsque l'on va relâcher le bouton. Tant que le bouton de la souris restera appuyé, le rectangle changera de dimensions, il faudra donc le rafraîchir l'affichage correctement en redessinant le contenu du `Canvas`.



B. Application au dessin d'un rectangle

Pour cet exercice il est utile de se servir d'un objet `Point` fourni par Java (c'est le moment de se rafraîchir la mémoire en jetant un coup d'œil à son API dans la documentation Java). Vous pouvez récupérer à partir de la page Web du cours, et ensuite compléter le fichier `DessinRect.java` qui contient deux classes.

Exercice 1.1 Commencez par compléter la partie du code concernant le `Canvas` (dans le fichier `DessinRect.java`), de façon à ce que ce dernier soit correctement affiché lorsque l'on exécute le programme. (transformez les commentaires 1 et 2 en code Java). Optez pour un canevas de taille 700x300. Sauvez et compilez pour vérifier le résultat (la souris ne fonctionne pas encore !).

• Maintenant, pour dessiner le rectangle, nous souhaiterions capter deux types d'événements : une pression sur un bouton et un mouvement de la souris avec le bouton enfoncé (*Drag*). Pour cela nous allons utiliser les deux classes `MouseAdapter` et `MouseMotionsAdapter` qui implémentent *trivialement* les interfaces `MouseListener` et `MouseMotionListener` (cf l'API).

Exercice 1.2 Dans le constructeur du canevas, redéfinissez les méthodes associées aux événements de la souris, pour indiquer ce que l'on veut qu'il arrive lorsque l'on appuie sur le bouton et lorsque l'on bouge la souris avec le bouton appuyé. Pour cela remplacez le commentaire 3 par le code suivant en le commentant et en le complétant. Toutes les variables de la classe nécessaires pour noter le point où a eu lieu le premier click et le point où la souris a glissé sont déclarées.

```
this.addMouseListener( new MouseAdapter()
  {
    public void mousePressed(MouseEvent e)
    {
          ...    // Mettre à jour les coordonnées et rafraîchir l'affichage
    }
  } );

this.addMouseMotionListener( new MouseMotionAdapter()
  {
    public void mouseDragged(MouseEvent e)
    {
          ...    // Mettre à jour les coordonnées et rafraîchir l'affichage
    }
  } );
```

Exercice 1.3 La dernière chose à faire maintenant est de dessiner le rectangle dans la méthode `paint(…)` du canevas. Pour cela remplacez le commentaire 4 par un algorithme qui calcule les coordonnées du coin supérieur gauche et les dimensions du rectangle à dessiner (les variables **entières** sont déjà déclarées) en utilisant les deux points (`clickPoint` et `dragPoint`) mis à jour par les procédures de l'écouteur de la souris.