C. Recanati
Java Graphique
AIR2

Feuille de TP 5 bis

On va maintenant transformer le programme `Sketcher` logiciel de dessin. On va dessiner dans la fenêtre au lieu d'imprimer des infos sur les événements. La `JTextArea` va donc être remplacée par un `JPanel` (classe SketcherView) qui affichera le dessin et captera les clics de souris pour permettre de dessiner un élément.

Pour réaliser cela, on va implémenter l'architecture Modèle/Vue introduite en cours, en définissant les deux classes `SketcherView` et `SketcherModel`. Le modèle d'une feuille de dessin a été défini par la classe `SketcherModel` comme une liste d'éléments pouvant être ajoutés ou retranchés à la feuille de dessin. La vue `SketcherView` gèrera l'affichage des éléments du modèle mais aussi l'interaction de l'utilisateur avec le logiciel (la partie « contrôleur »). On va donc implanter dans `SketcherView` la création interactive d'un nouvel élément, puis son ajout au modèle à la fin de l'interaction de l'utilisateur avec la souris. Cette « vue » (ici constituée d'un `JPanel` étendu) sera insérée au centre du panneau du cadre de l'application.

Exercice 1:
Dans cet exercice, on va implémenter la création interactive d'un élément dans la vue (SketcherView.java), et laisser en commentaire la partie concernant le modèle. L'utilisateur va ici pouvoir dessiner interactivement un élément temporaire avec la souris dans `SketcherView`.

On gèrera l'enchaînement des événements souris permettant de dessiner un élément à l'aide d'une classe interne `MouseHandler` qui étend `MouseInputAdapter`. Un premier clic de souris, suivi d'un déplacement de souris bouton enfoncé (un drag), suivi du relâchement du bouton, permettront de dessiner interactivement un nouvel élément (ligne, rectangle, etc. selon le type d'élément courant sélectionné).

- Sur un clic : on enregistre le point de début du dessin (origine d'un élément temporaire `tempElement`).

- Sur un mouvement de drag (bouton de souris enfoncé) :
    a. on efface l'élément temporaire précédemment dessiné;
    b. on enregistre le point du curseur comme nouvelle extrémité de l'élément temporaire;
    c. on trace le nouvel élément temporaire.
- Sur un relâcher, on enregistre l'élément temporaire qui vient d'être défini dans le modèle et on réinitialise les variables en vue d'une future utilisation.

☞ 1. Pour dessiner (ou effacer !... ça sera pareil), on utilise la méthode générique `draw(Shape)` de `graphics2D` en mode XOR (utiliser `setXorMode`) . On mémorisera l'élément temporaire dans un membre de la classe `MouseHandler` ainsi que les points de début et fin pour pouvoir continuer de dessiner, de mouvement en mouvement.

2. Pour faire référence aux constantes qui identifient le type d'éléments, on utilisera l'interface `Constants`.

Exercice 2:
Terminez le logiciel en complétant l'architecture modèle/vue.

Elements de correction

```
// fichier SketcherView.java
import javax.swing.*;
import javax.swing.event.*;
import java.util.*;
import java.awt.*;
import java.awt.geom.*;
import java.awt.event.*;

class SketcherView extends JPanel
                                implements Observer, Constants {
    private Sketcher theApp;  // l'objet instance d'application

    public SketcherView(Sketcher theApp) {
        this.theApp = theApp;
        setBackground(Color.white);
        setOpaque(true);
        setBorder(BorderFactory.createLineBorder(Color.black));

        MouseHandler handler = new MouseHandler();
```

```
        addMouseListener(handler);   // il faut l'enregistrer 2 fois
        addMouseMotionListener(handler);
    }

    public void update(Observable o, Object rectangle) {
        // code correspondant aux modifs du modèle
            if (rectangle == null)
                repaint();
            else repaint( (Rectangle) rectangle);
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g); // toujours appeler en premier
        Graphics2D g2D = (Graphics2D) g;
        Iterator elements = theApp.getModel().getIterator();
        Element element;

        while (elements.hasNext()) {
            element = (Element) elements.next();
            g2D.setPaint(element.getColor());
            g2D.draw(element.getShape());
        }
     }

     class MouseHandler extends MouseInputAdapter {
        private Point debut;     // position du clic
        private Point dernier;  // position en drag
        private Element tempElement ;

        private Graphics2D g2D;

        public void mousePressed (MouseEvent e) {
            // clic du bouton de souris
            debut = e.getPoint();
            System.out.println("premier clic");

            g2D = (Graphics2D) getGraphics();
```

```
    g2D.setPaint(theApp.getWindow().getElementColor());
    g2D.setXORMode(Color.white);
}

public void mouseDragged (MouseEvent e) {
            // deplacement de souris
    dernier = e.getPoint();
    if (tempElement == null) { // la premiere fois
        tempElement = createElement(debut, dernier);
    }
    else { // effacer le precedent en dessinant en Xor
        g2D.draw(tempElement.getShape());
            // modifier l'element
        tempElement.modify(debut, dernier);
    }
    // dessiner le nouvel element tjs en mode Xor
    g2D.draw(tempElement.getShape());
}

public void mouseReleased (MouseEvent e) {
    System.out.println("souris relachee");

    if (tempElement != null) {
        theApp.getModel().add(tempElement);
        tempElement = null;
    }
    if (g2D != null) {
        g2D.dispose();
        g2D = null;
    }
    debut = dernier = null;
}

private Element createElement(Point debut, Point fin) {
    switch(theApp.getWindow().getElementType()) {
        case LIGNE: return new Element.Ligne(debut, fin,
                theApp.getWindow().getElementColor());
```

```
                case RECTANGLE:
                        return new Element.Rectangle(debut, fin,
                        theApp.getWindow().getElementColor());
                case CERCLE:
                        return new Element.Cercle(debut, fin,
                        theApp.getWindow().getElementColor());
                case COURBE:
                        return new Element.Courbe(debut, fin,
                        theApp.getWindow().getElementColor());
            }
            return null;
        }
    }
}
```