C. Recanati
Interfaces Graphiques
INFO2, 2014

Feuille 7

Pour ceux qui se considèrent comme trop en retard sur le programme Sketcher, je vous propose de traiter l'exercice 3. (Si le dessin d'un rectangle vous pose problème, traitez d'abord l'exercice avec un segment de droite).

Pour les autres, on va maintenant transformer le programme `Sketcher` pour en faire un logiciel de dessin. On va dessiner dans la fenêtre au lieu d'imprimer des infos sur les événements se produisant sur les menus. La `JTextArea` va donc être remplacée par un `JPanel` qui captera les clics de souris, et permettra de dessiner interactivement.

Pour réaliser ce logiciel, on va implémenter l'architecture Modèle/Vue introduite en cours, en définissant les deux classes `SketcherView` et `SketcherModel` (cf. poly).
Le modèle de feuille de dessin a été défini par la classe `SketcherModel` comme une liste d'éléments pouvant être ajouté ou retranché. 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 d'abord de côté la partie concernant le modèle. L'utilisateur va ici dessiner interactivement un élément temporaire avec la souris dans `SketcherView`.

On va gérer les événements souris permettant de dessiner un élément à l'aide d'une classe écouteur interne `MouseHandler` qui étend `MouseInputAdapter`: un premier clic de souris, suivi d'un déplacement de souris bouton enfoncé (drag) suivi du relâchement du bouton, permettront de dessiner interactivement un nouvel élément (ligne, rectangle, etc. selon le type d'élément indiqué dans la variable `elementType`).
- Sur un clic : on mémorise le point de début du dessin (origine d'un élément temporaire `tempElement` dans la classe `MouseHandler`). On prépare aussi un contexte graphique g2D pour dessiner.

- Sur un mouvement, avec bouton de souris enfoncé : on distingue deux cas selon que c'est le premier mouvement ou selon que l'on est en cours mouvement. Si c'est le premier mouvement, on crée un élement temporaire qu'on mémorise dans un membre de la classe `MouseHandler`, et on le dessine ; sinon : a. on efface l'élément temporaire précédemment dessiné; b. on enregistre le point du curseur comme nouvelle extrémité de l'élément temporaire; et c. on trace le nouvel élément temporaire.
- Sur un relâcher, on réinitialise les variables pour une nouvelle utilisation (création d'un autre élément), et on ajoute l'élément temporaire qui vient d'être défini au modèle de dessin.

☞ Pour dessiner (ou effacer, c'est pareil !), on utilise la méthode générique `draw` d'un objet `Shape` de `graphics2D` en mode XOR (utiliser `setXorMode` sur `g2D`, et appeler `draw(getShape(element))`).

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

Exercice 3 :
On va dessiner un rectangle qui suit la souris (comme dans un logiciel de dessin pour créer un rectangle) dans un panneau qu'on placera au centre d'un `JFrame`. Sur l'événement de clic, on mémorisera le sommet initial du rectangle, puis on tracera le rectangle correspondant au coin diamétralement opposé au fur et à mesure des mouvements de souris avec bouton enfoncé. Sur le relâchement, on dessinera simplement le rectangle. Pour gérer ces événements souris, suivre les indications qui ont été données dans l'exercice 1 pour la classe `MouseHandler`.

Elements de correction

```
// 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;  // Objet Application

    public SketcherView(Sketcher theApp) {
        this.theApp = theApp;
```

```
            setBackground(Color.white); setOpaque(true);
        setBorder(BorderFactory.createLineBorder(Color.black));


        MouseHandler handler = new MouseHandler();
        addMouseListener(handler);
        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);
    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 au 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) {
        tempElement = createElement(debut, dernier);
    }
    else { // effacer le precedent
        g2D.draw(tempElement.getShape());
            // modifier l'element
      tempElement.modify(debut, dernier);
    }
   // dessiner le nouveau
    g2D.draw(tempElement.getShape());
}

public void mouseReleased (MouseEvent e) {
    // relachement de la souris
    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;
            }}}
```

Exercice 3 :

```
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;

public class DessinRect extends JFrame {

    public DessinRect(String title) {
        super(title);
        Toolkit tlk = getToolkit();
        Dimension dim = tlk.getScreenSize();
        setBounds(dim.width/4, dim.height/4, dim.width/2,
dim.height/2);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        Panneau panel = new Panneau();
```

```
        getContentPane().add(panel, BorderLayout.CENTER);
    }

    public static void main(String args[]) {
        DessinRect fenetre = new DessinRect("Dessiner un rectangle");
        fenetre.setVisible(true);
    }
}

class Panneau extends JPanel {
    public Panneau() {
            super();
            setBackground(Color.WHITE);
            MouseHandler mh;
            addMouseListener(mh = new MouseHandler() );
            addMouseMotionListener(mh);
    }
    /* la classe interne pour gérer les événements souris */
    class MouseHandler extends MouseInputAdapter {
            private Point debut, fin;
            private Rectangle2D.Double tempRect;
            private Graphics2D g2D;

            public void mousePressed(MouseEvent e) {
                debut = e.getPoint(); // on mémorise l'origine

                // on initialise le contexte graphique
                g2D = (Graphics2D) getGraphics();
                g2D.setStroke((Stroke) new BasicStroke(2));
                g2D.setPaint(Color.black);
                g2D.setXORMode(Color.white);
            }

            public void mouseDragged (MouseEvent e) {
                // deplacement de souris
                fin = e.getPoint(); // dernier point
                if (tempRect == null) {
```

```
                // creer un premier rectangle tempRect à partir de debut
                 tempRect =
                    new Rectangle2D.Double(Math.min(debut.x, fin.x),
                    Math.min(debut.y, fin.y),Math.abs(debut.x - fin.x),
                    Math.abs(debut.y - fin.y) );
                }
                else { // effacer le precedent rectangle tempRect
                    g2D.draw(tempRect);
                    // modifier le rectangle tempRect avant de l'afficher
                      tempRect.x = Math.min(debut.x, fin.x);
                      tempRect.y = Math.min(debut.y, fin.y);
                      tempRect.width = Math.abs(debut.x - fin.x);
                      tempRect.height = Math.abs(debut.y - fin.y);
                }
                // dessiner le nouveau rectangle tempRect
                g2D.draw(tempRect);
            }

            public void mouseReleased (MouseEvent e) {
                // relachement de la souris
                if (tempRect != null)
                     tempRect = null;
                if (g2D != null) {
                    g2D.dispose();
                    g2D = null;
                }
                debut = fin = null;
            }
        }
}
```