**C. Recanati**
**Java Graphique**
**INFO2, 2013**

# Feuille de TP n° 5

## Exercice 1

Ecrire une classe `MonBouton` qui étend `JButton` et définit un type de bouton qui modifie le curseur de souris quand elle entre sur le bouton. Tester votre classe (en définissant par exemple une classe `BoutonCurseur` étendant `JFrame`, dont le panneau de contenu contiendra un `MonBouton`).

☞ Pour créer un curseur de souris, on peut utiliser les constantes de la classe `Cursor`.

☞ ☞ On peut sélectionner les entrées/sorties de la souris sur le bouton en définissant le bouton comme étant son propre écouteur de souris (`MouseListener`). Pour définir cet écouteur, on peut utiliser une classe interne de type `Adapter` (une classe qui étend `MouseAdapter`) que l'on instanciera par `new` au moment de l'enregistrement de l'écouteur.

☞ ☞ ☞ Pour affecter le curseur, on peut utiliser `getSource()` qui retourne l'object source de l'événement et ainsi écrire `((Component) e.getSource()).setCursor(handCursor)` ou encore utiliser `getComponent()` - héritée de `ComponentEvent` qui retourne directement la source comme `Component`.

## Exercice 2 (*suite de Sketcher, logiciel de dessin*).

*On va garder provisoirement le* `JTextArea` *pour afficher des informations sur les événements, mais à terme, on le supprimera pour dessiner dans la fenêtre.*

Les items du menu Elements permettent de sélectionner la couleur et la forme de l'élément que l'on souhaite dessiner. On rajoute donc deux nouveaux membres à la classe `SketcherFrame` : `elementColor` de type Color et `elementType` de type entier. Ces membres seront mis à jour quand on sélectionnera les options du menu Elements.

Pour que les initialisations soient claires et faciles à modifier, on a défini dans un fichier séparé une interface `Constants` qui définit les types d'éléments entiers `LIGNE, RECTANGLE, CERCLE, COURBE` et les conditions initiales des membres `elementType` et `elementColor` (resp. `DEFAULT_ELEMENT_TYPE` et `DEFAULT_ELEMENT_COLOR`). La classe `SketcherFrame`

implémente l'interface `Constants` et se sert de ces constantes pour faire les initialisations.

1) Définir l'interface `Constants` et modifier la création des cases à cocher du sous-menu `Couleurs` pour que la case à cocher marquée initialement corresponde à la couleur de défaut initiale. Modifier également la création des boutons radio pour qu'ils soient sélectionnés au départ conformément au type initial de `elementType`.

2) Pour gérer les types d'options du menu Elements, définir une classe interne `TypeListener` qui implémente `ActionListener` et comporte un membre privé `type` permettant de stocker le type de l'élément écouté. Cette classe aura un constructeur qui prend en argument le type et le mémorise dans son champs privé. Cela permettra d'adapter les écouteurs aux différents items du menu Elements à leur création. Lorsqu'une option du menu sera activée, l'écouteur consultera son type privé et mettra `elementType` à jour.

3) On procèdera de même pour gérer la couleur en implémentant une classe interne `ColorListener` qui comportera un membre privé `color` permettant de définir la couleur de l'élément écouté.

*Si vous avez le temps…*
4) Ajouter des écouteurs anonymes aux boutons de la ToolBar ('anonyme' signifie qu'on définit l'écouteur directement au moment où on l'ajoute, sans le nommer) pour afficher dans le `JTextArea` quel bouton (gauche, milieu, ou droit) a été enfoncé.

**Eléments de correction**

// BoutonCurseur.java

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

public class BoutonCurseur extends JFrame {

     public BoutonCurseur() {
          getContentPane().add(new MonBouton ("Essai de
\Curseur"));
     }

     public class MonBouton extends JButton {

          public MonBouton(String titre){
               super(titre);
```

```
                this.addMouseListener(new MouseHandler());
          }

          class MouseHandler extends MouseAdapter {
                Cursor handCursor = new Cursor
                                    (Cursor.HAND_CURSOR);
                Cursor defaultCursor = new Cursor
                                    (Cursor.DEFAULT_CURSOR);
                public void mouseEntered(MouseEvent e) {
                     e.getSource()).setCursor(handCursor);
                }
                public void mouseExited(MouseEvent e) {

e.getComponent().setCursor(defaultCursor);
                }
          } // fin de la classe MouseHandler
     }    // fin de la classe interne MonBouton

     public static void main(String args[]) {
          BoutonCurseur window = new BoutonCurseur();

          window.setTitle("Bouton + Curseur");
          window.setDefaultCloseOperation
                                    (JFrame.EXIT_ON_CLOSE);
          window.setBounds(100, 200, 300, 200);
          window.setVisible(true);
     }
}


// fichier Sketcher.java
import java.awt.*;

public class Sketcher {
   static SketcherFrame window;
   public static void main(String[] args) {
          window = new SketcherFrame("Sketcher");

          Toolkit leKit = window.getToolkit();
          Dimension wndSize = leKit.getScreenSize();
          window.setBounds(wndSize.width/4,
                           wndSize.height/4,
                           wndSize.width/2,
                           wndSize.height/2);
          window.setVisible(true);
   }
}

// fichier Constants.java
import java.awt.*;
```

```
public interface Constants {

     // Les types d'élements
     int LIGNE = 101;
     int RECTANGLE = 102;
     int CERCLE = 103;
     int COURBE = 104;

     // Les conditions initiales
     int DEFAULT_ELEMENT_TYPE = RECTANGLE;
     Color DEFAULT_ELEMENT_COLOR = Color.green;
}

// fichier SketcherFrame.java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class SketcherFrame extends JFrame
                                    implements constants {

     protected JTextArea output ;
     protected JScrollPane scrollPane;
     private JMenuBar menuBar;

     private Color elementColor =
                              DEFAULT_ELEMENT_COLOR ;
     private int elementType = DEFAULT_ELEMENT_TYPE;

     protected String newline = "\n";

     public SketcherFrame(String titre) {

          JMenu menu, submenu;
          JMenuItem menuItem;
          JRadioButtonMenuItem rbMenuItem;
          JCheckBoxMenuItem cbMenuItem;

          setTitle(titre);
          setDefaultCloseOperation(EXIT_ON_CLOSE);

               //Créer la toolbar.
          JToolBar toolBar = new JToolBar();
          toolBar.setFloatable(false);
          addButtons(toolBar);

          // etc. début de la construction inchangé

          // Construire le menu Elements …
          menu = new JMenu("Elements");
          menu.setMnemonic(KeyEvent.VK_E);
```

```
        menuBar.add(menu);

        ButtonGroup group = new ButtonGroup();

        rbMenuItem = new JRadioButtonMenuItem("Ligne",
                                  elementType == LIGNE);
        rbMenuItem.setMnemonic(KeyEvent.VK_L);
        group.add(rbMenuItem);
        rbMenuItem.addActionListener(new
                                  TypeListener(LIGNE));
        menu.add(rbMenuItem);

        rbMenuItem = new JRadioButtonMenuItem
                      ("Rectangle",
                           elementType == RECTANGLE);
        rbMenuItem.setMnemonic(KeyEvent.VK_R);
        group.add(rbMenuItem);
        rbMenuItem.addActionListener
                      (new TypeListener(RECTANGLE));
        menu.add(rbMenuItem);

        // etc. pour les autres types de formes

        menu.addSeparator();

        submenu = new JMenu("Couleur");
        submenu.setMnemonic(KeyEvent.VK_C);
             // le groupe de check box menu items
        ButtonGroup group2 = new ButtonGroup();

        cbMenuItem = new JCheckBoxMenuItem("Rouge",
                     elementColor.equals(Color.red));
        cbMenuItem.setMnemonic(KeyEvent.VK_R);
        group2.add(cbMenuItem);
        cbMenuItem.addActionListener
                  (new ColorListener(Color.red));
        submenu.add(cbMenuItem);

        // ibid pour bleu, vert, etc.


        //Construire le menu Aide
        menu = new JMenu("Aide");
        menu.setMnemonic(KeyEvent.VK_A);
        // le mettre complètement à droite dans la barre
        menuBar.add(Box.createHorizontalGlue());
        menuBar.add(menu);

        // fin du constructeur
    }

    class TypeListener implements ActionListener {
```

```
          private int type;

          TypeListener(int type) {
               this.type = type;   // mémorise le type
          }

          public void actionPerformed(ActionEvent e) {

               elementType = type; // l'essentiel !

               JMenuItem source
                    = (JMenuItem)(e.getSource());
               String s = "ActionEvent detecte dans
\TypeListener."
                    + newline
                    + "    Event source: "
                    + source.getText()
                    + " (une instance de "
                    + getClassName(source) + ")";
               output.append(s + newline);
          }
     }

     class ColorListener implements ActionListener {
          private Color color;

          ColorListener(Color color) {
               this.color = color; // mémorise color
          }

          public void actionPerformed(ActionEvent e) {

               elementColor = color; // l'essentiel !

               JMenuItem source =
                         (JMenuItem)(e.getSource());
               String s = "ActionEvent detecte dans
\ ColorListener."
                         + newline
                         + "    Event source: "
                         + source.getText()
                         + " (une instance de "
                         + getClassName(source) + ")";
               output.append(s + newline);
          }


     }

    // Retourne juste le nom de la classe
     protected String getClassName(Object o) {
          String classString = o.getClass().getName();
```

```
        int dotIndex = classString.lastIndexOf(".");
        return classString.substring(dotIndex+1);
    }

    protected void addButtons(JToolBar toolBar) {
        JButton button = null;

            // premier bouton
        button = new JButton(new
                ImageIcon("images/left.gif"));
        button.setToolTipText("bouton de gauche");
        button.addActionListener(new
                    ActionListener() {
            public void actionPerformed
                        (ActionEvent e){
              displayResult("ActionEvent bouton1");
            }
        });
        toolBar.add(button);

       // ibid deuxieme et troisieme boutons
    }

    protected void displayResult
                      (String actionDescription) {
        output.append(actionDescription + newline);
    }
}
```