C. Recanati
Java Graphique
AIR2

TP 5 bis (Sketcher simplifié)

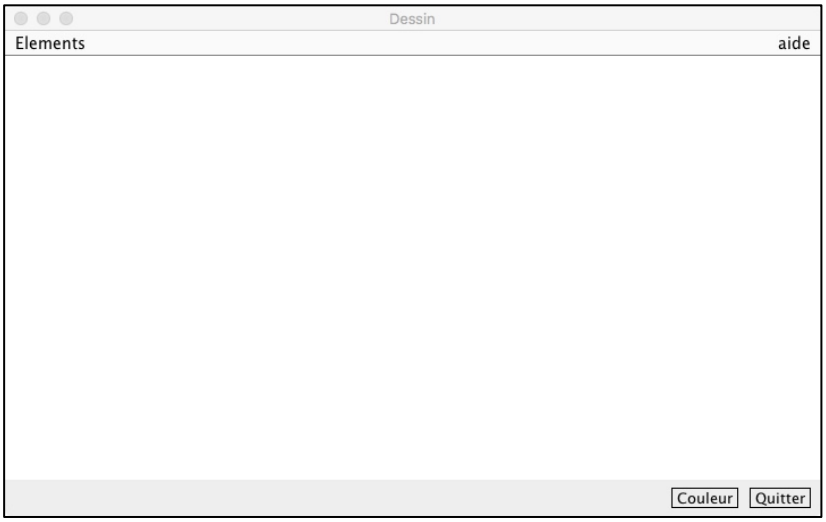
Dans cet exercice simplifié du programme `Sketcher`, on va implémenter le dessin interactif d'un rectangle et d'une ligne (mais sans utiliser la classe abstraite `Element` du programme Sketcher pour simplifier le programme).

On va partir de la classe `SketcherFrame` du logiciel de dessin pour définir une nouvelle classe `Dessin`, extension simple de `JFrame`. Une fenêtre Dessin aura pour titre Dessin, et une barre de menu avec seulement 2 menus : un premier menu `Elements` et un menu `Aide` situé à droite dans la barre de menu. La classe implémentera l'interface `Constants` définissant des constantes permettant d'initialiser le membre `elementType` de la classe Dessin qui définira le type de l'élément courant à dessiner. Mais on pourra aussi bien définir des constantes entières RECTANGLE et LIGNE dans la classe Dessin pour simplifier le programme et on se passera alors de l'interface `Constants`.

Le menu Elements de la fenêtre Dessin contiendra deux items de menu qui formeront un groupe de boutons radio (type `JRadioButtonMenuItem`). Ces boutons radios permettront de sélectionner le type de l'élément courant à dessiner en modifiant le membre elementType de la classe avec les constantes entières définies précédemment.

Question 1: Afficher d'abord une instance de la classe Dessin en implémentant la barre de menu avec ses deux menus. Faites un main dans lequel on crée une instance de cette classe, et où on affiche sur la console les valeurs de elementType quand elles sont modifiées via les boutons radios du menu Elements.

Question 2: Ajouter au centre du cadre de Dessin un panneau dans lequel on pourra dessiner, et en dessous, un autre panneau contenant deux boutons alignés à droite : un bouton Couleur et un bouton Quitter. L'allure de la fenêtre sera alors semblable à celle de la figure suivante.



Question 3 : Implémenter l'interface `ActionListener` dans le cadre Dessin pour que l'application termine lorsque l'utilisateur clique sur le bouton Quitter. La procédure `actionPerformed` pourra afficher une boite de confirmation standard permettant à l'utilisateur de confirmer sa demande. Rappel : il ya dans la classe `JOptionPane` des méthodes pour créer des boîtes standards).

Question 4 : Pour le bouton Couleur, on implémentera l'apparition d'une boite de dialogue permettant de choisir une couleur (avec un `JColorChooser`) et on affectera la couleur sélectionnée au fond du panneau central.

Question 5 : On va dessiner maintenant dans le panneau situé au centre de la fenêtre principale. Ce Panneau sera une extension de `JPanel`. Il captera les clics et les mouvements de la souris et permettra de dessiner interactivement un élément (Rectangle ou Ligne).

L'utilisateur pourra dessiner interactivement un rectangle ou une ligne selon la valeur du membre `elementType` qui détermine le type de l'élément courant dessiné dans le panneau de la fenêtre de l'application quand l'utilisateur clique sur la souris.

On gèrera les événements souris permettant de dessiner un élément à l'aide d'une classe interne `MouseHandler` qui étendra `MouseInputAdapter`. Pour faire ces dessins, des attributs `tempRect` et `tempLine` (de type `Rectangle2D.Double` et `Line2D.Double`) y seront déclarés et initialisés sur un clic. Ils permettront de dessiner un élément temporaire directement dans la classe `MouseHandler`.

Principe du dessin interactif d'un élément : un premier clic de souris, suivi ensuite d'une succession de déplacements de la souris avec ce bouton enfoncé (un drag), suivi enfin (pour terminer le dessin) d'un relâchement du bouton de la souris, sera la séquence d'événements permettant de dessiner interactivement le nouvel élément à venir (ligne ou rectangle, selon la valeur de `elementType` au moment du clic initial).

On mémorisera l'élément temporaire en cours de dessin dans un des membres de la classe `MouseHandler` (par exemple dans `tempRect ou tempLine`) ainsi que les points de début et de fin caractérisant cet élément courant, afin de le dessiner, de mouvement de souris en mouvement de souris. Ainsi :

* Sur un clic : on enregistre le point de début du dessin (origine de élément temporaire `tempRect ou tempLine`).
* Sur un mouvement de drag (bouton de souris enfoncé) :
    a. on efface l'élément temporaire précédemment dessiné (lors du mouvement précédent);
    b. on enregistre le point du curseur (la nouvelle extrémité de l'élément temporaire);
    c. on trace le nouvel élément temporaire.
* Sur un relâcher du bouton, on efface le dernier élément temporaire et on réinitialise les variables locales du MouseHandler en vue d'une future utilisation pour dessiner un nouvel élément.

Remarques 1. Pour dessiner ou effacer ! …c'est pareil, car on utilise la méthode générique `draw(Shape)` de `graphics2D` **en mode XOR** (utiliser `setXorMode`) . Cette remarque explique pourquoi on redessine l'ancien élément (ça l'efface) avant de dessiner le nouveau dans la partie gérant un mouvement de Drag.
2. Pour mettre au point le dessin d'un rectangle donné par les deux sommets de sa diagonale, vous pouvez regarder dans le corrigé de la feuille 3 l'implantation de sa méthode `modify` dans la classe `Element`.

Elements de correction

```
public class Dessin extends JFrame implements Constants {
      private JMenuBar menuBar; // la barre de menu
      JRadioButtonMenuItem rbMenuItem;
      JMenu menu;

      private int elementType = DEFAULT_ELEMENT_TYPE;
      private Panneau panel; // une classe de panneau spécifique
// pour la question 2, positionner au sud un bouton Quitter
      private JButton quit = new JButton("Quitter");

      public Dessin (String title) {
```

```
      super(title);

      // creer la barre de menu de Dessin
      ...
      ...
      // creer son menu Elements
      menu = new JMenu("Elements");
      menu.setMnemonic(KeyEvent.VK_E);
      menuBar.add(menu);

// lui mettre des boutons radio "Ligne" et "Rectangle"
// comme dans SketcherFrame: ils sont initialises
// avec Constants ou d'une autre manière
// et permettent de modifier
// elementType via un ecouteur qu'on pourra appeler TypeListener
      ...
      ...
      ...
      menu = new JMenu("Aide");
// ajouter le menu Aide a droite dans la barre de menu
      ...
      panel = new Panneau(this); // notre classe Panneau
// ajouter le panel au centre
      getContentPane().add(panel, BorderLayout.CENTER);

// question 2
// compléter la classe pour gérer le bouton Quitter
// et le bouton Couleur
      ...
// dimensionner la fenetre de Dessin pour l'affichage initial
      ...

} // fin du constructeur de Dessin

// classe interne pour gérer les boutons radio
class TypeListener implements ActionListener {
      private int type;
      TypeListener(int type) {
      ...
      }
      public void actionPerformed(ActionEvent e)  {
            ...
      }
}
int getElementType() {
      return elementType;
}


public static void main(String args[]) {
      Dessin fenetre = new Dessin ("Dessiner");
      fenetre.setVisible(true);
```

```
	}
} // fin de la classe Dessin

class Panneau extends JPanel implements Constants {
	Dessin app;  // pour récupérer ensuite elementType

	public Panneau(Dessin inst) {
		super();
		app = inst;

		setBackground(Color.WHITE);

		MouseHandler mh = new MouseHandler();
	// l'enregistrer comme MouseListener et MouseMotionListener
		...
	}
	// la classe interne pour gerer les evenements souris
	class MouseHandler extends MouseInputAdapter {
		// pour gerer l'element temporaire en cours de dessin
		int elementType;
		private Point debut, fin;
		private Rectangle2D.Double tempRect;
		private Line2D.Double tempLine;
		private Graphics2D g2D;

		public void mousePressed(MouseEvent e) {
			//récupérer le type de l'élément courant dessiné
			elementType = ...

			debut = e.getPoint(); // on memorise l'orig. du clic
			// et on initialise le contexte de dessin g2D
			g2D = (Graphics2D) getGraphics();
			g2D.setStroke((Stroke) new BasicStroke(2));
			g2D.setPaint(...);
			g2D.setXORMode(...);
		}
		public void mouseDragged (MouseEvent e) {
			// deplacement de la souris bouton enfoncé
			fin = ... // le "dernier" point finalement
			switch (elementType) {
				case LIGNE:
					if (tempLine == null) { // 1ere fois
					// creer une ligne avec debut et fin
						tempLine = new Line2D.Double
								(...);
					}
					else {
						// effacer la ligne precedente
						// en la redessinant
						g2D.draw(tempLine);

						// definir la nouvelle ligne
```

```
                                    tempLine.x1 = ...;
                                    tempLine.y1 = ...;
                                    tempLine.x2 = ...;
                                    tempLine.y2 = ...;
                              }
                              ... // dessiner la nouvelle ligne
                              break;

                        case RECTANGLE:
                              if (tempRect == null) {
                                    // creer un premier rectangle
                                    tempRect = new
                                          Rectangle2D.Double(... );
                              }
                              else {
                                    // effacer l'ancien rectangle
                                    g2D.draw(tempRect);
                                    // modifier le rectangle
                                    tempRect.x = ...;
                                    tempRect.y = ...;
                                    tempRect.width =
                                          ...;
                                    tempRect.height =
                                          ...;
                              }
                              g2D.draw(tempRect);// dessiner le nouveau
                              break;
                  } // fin du switch
            } // fin de mouseDragged()

            public void mouseReleased (MouseEvent e) {
                        // on reinitialise le Handler
                        // sur le relacher de souris
                  switch(elementType) {
                        case RECTANGLE:
                                    if (tempRect != null)
                                          tempRect = null;
                                    break;
                        case LIGNE:
                                    if (tempLine != null)
                                          tempLine = null;
                                    break;
                  }
                  if (g2D != null) {
                        g2D.dispose();
                        g2D = null;
                  }
                  debut = fin = null;
            } // fin de mouseReleased()
      } // fin de MouseHandler
} // fin de Panneau
```