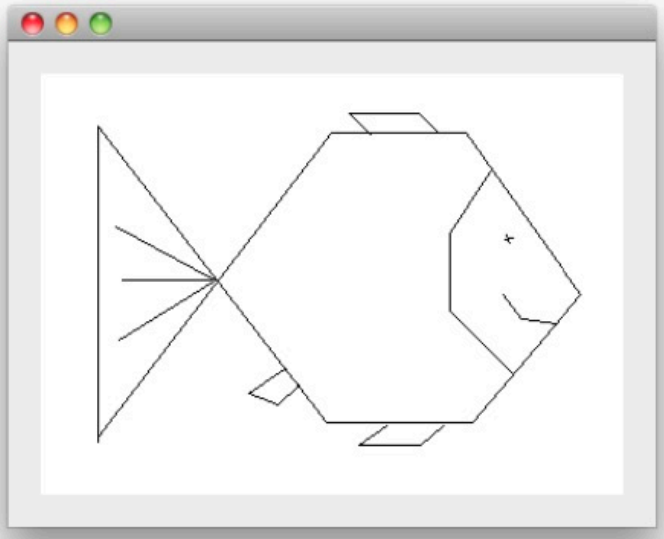
Exercice E.2: Traceur de lignes 1



Sachant que :

- l'événement **mousePressed** intervient lorsque l'utilisateur enfonce le bouton de la souris,
- l'événement **mouseDragged** intervient lorsque l'utilisateur bouge la souris **avec un bouton maintenu enfoncé**,
- l'événement **mouseReleased** intervient lorsque l'utilisateur relâche un bouton de la souris,
- un objet du type **MouseEvent** possède une méthode **getPoint()** qui retourne la position actuelle de la souris. Le résultat est du type **Point**.

Créez une application permettant de tracer à l'aide de la souris des lignes sur un composant du type **JPanel**.

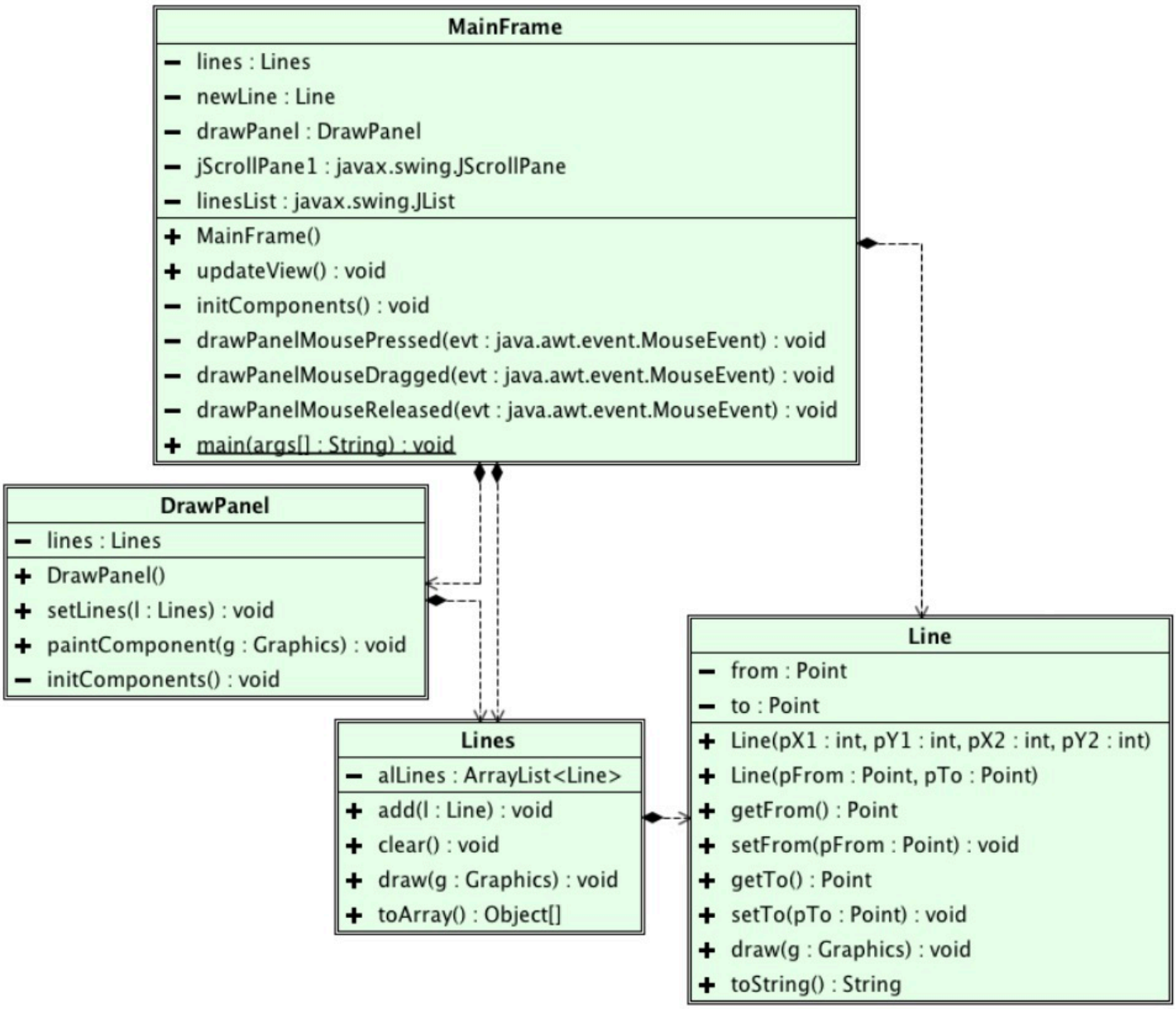
Principe de fonctionnement (Voir aussi le diagramme UML à la prochaine page)

Procédez de façon analogue à l'exercice **Turtles** :

- La classe **Line** sait représenter une ligne. La classe **Line** possède une méthode **draw(Graphics g)** pour se dessiner sur un canevas **g**.
- La classe **Lines** sait gérer une liste de lignes dans une **ArrayList** nommée **allLines.** La classe **Lines** possède une méthode **draw(Graphics g)** pour dessiner toutes les lignes sur un canevas **g**.
- La fiche principale **MainFrame** est le contrôleur : elle possède un attribut du type **Lines** qu'elle initialise au démarrage. Elle possède aussi un objet **drawPanel** du type **DrawPanel**. **MainFrame** réagit aux événements de la souris qui sont effectués sur **drawPanel**. Les lignes sont créées dans **MainFrame** qui les ajoute immédiatement à l'objet **lines**. Après chaque modification dans **lines**, elle fait redessiner **drawPanel**.
- **DrawPanel** est la vue : elle possède un attribut du type **Lines** dont elle a besoin pour dessiner les lignes. **DrawPanel** obtient les lignes de la fiche principale par un modificateur **setLines**. **DrawPanel** dessine toutes les lignes dans sa méthode **paintComponent(Graphics g)**.
- Une nouvelle ligne est créée lorsque le bouton est enfoncé. Les coordonnées de la ligne sont modifiées aussi longtemps que le bouton reste enfoncé et elle est redessinée chaque fois que ses coordonnées changent. (→ De cette façon, nous voyons la ligne déjà lorsque nous sommes en train de la tracer avec la souris)

On a donc deux fois un attribut **lines** : une fois dans **DrawPanel**, l'autre dans **MainFrame**, mais les deux doivent référencer (pointer sur) le même objet ! **MainFrame** effectue la gestion et le contrôle de **lines**, tandis que **DrawPanel** fait dessiner les lignes sur son canevas.

Amélioration (déjà intégrée dans le diagramme UML) :



Lorsque vous savez dessiner des lignes, ajoutez une **`JList`** à **`MainFrame`** (nommée **`linesList`**) pour afficher les coordonnées des lignes contenues dans **`lines`**. Pour cela, vous devez définir :

`Line.toString()`,

`Lines.toArray()`.

En plus, c'est pratique de définir dans **`MainFrame`** une méthode **`updateView`** qui redessine **`drawPanel`** et ré-affiche le contenu de **`lines`** dans **`linesList`**. Cette méthode **`updateView`** est alors appelée après chaque changement de **`lines`** (au lieu de **`repaint`**).

```java

import java.awt.Color;
import java.awt.Graphics;

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.
txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JPanel.java to
edit this template
 */

/**
 *
 * @author luxformel
 */
public class DrawPanel extends javax.swing.JPanel {
    private Lines lines;

    public void setLines(Lines pLines){
        lines = pLines;
    }
    /**
     * Creates new form DrawPanel
     */
    public DrawPanel() {
        initComponents();
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g); // Generated from nbfs:
//nbhost/SystemFileSystem/Templates/Classes/Code/OverriddenMethodBody
        g.setColor(Color.white);
        g.fillRect(0, 0, getWidth(), getHeight());
        if (lines != null)
            lines.draw(g);
    }

    /**
     * This method is called from within the constructor to initialize the
form.
     * WARNING: Do NOT modify this code. The content of this method is
always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")



    // Variables declaration - do not modify//GEN-BEGIN:variables
    // End of variables declaration//GEN-END:variables
}
```

```java

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Point;

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author luxformel
 */
public class Line {
    //Attributs
    //from - point de départ de la ligne
    //to - point de fin de la ligne
    //color - couleur de la ligne
    private Point from;
    private Point to;
    private Color color;

    public Line(Point from, Point to, Color pColor) {
        this.from = from;
        this.to = to;
        color = pColor;
    }

    public Line(int pX1, int pY1, int pX2, int pY2, Color pColor) {
        from = new Point();
        from.x = pX1;
        from.y = pY1;

        //Soit on instancie directement l'objet avec les coordonnés
        to = new Point(pX2, pY2);

        color = pColor;
    }

    public Point getFrom() {
        return from;
    }

    public void setFrom(Point from) {
        this.from = from;
    }

    public Point getTo() {
        return to;
```

```
52      }
53
54      public void setTo(Point to) {
55          this.to = to;
56      }
57
58      public void draw(Graphics g){
59          g.setColor(color);
60          g.drawLine(from.x, from.y, to.x, to.y);
61      }
62
63      public String toString(){
63          return "(" + from.x + "," + from.y + ") -> (" + to.x + "," + to.y +
64 ")";
65      }
66 }
67
```

```java

import java.awt.Graphics;
import java.util.ArrayList;

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author luxformel
 */
public class Lines {
    //Attribut
    //Initialiser / Instancier l'attribut
    private ArrayList<Line> alLines = new ArrayList<>();

    //Ajouter une ligne à la liste
    public boolean add(Line e) {
        return alLines.add(e);
    }

    //Supprimer toutes les lignes de la liste
    public void clear() {
        alLines.clear();
    }

    //Transformer l'ArrayList en tableau pour pouvoir l'afficher plus tard
dans
    //la JList dans le MainFrame
    public Object[] toArray() {
        return alLines.toArray();
    }

    //Dessiner toutes les lignes contenues dans la liste
    public void draw(Graphics g){
        //Parcourir une ligne après l'autre de la liste
        for (int i = 0; i < alLines.size(); i++){
            //Récupérer la ligne actuelle dans une variable
            Line line = alLines.get(i);
            //Dessiner la ligne
            line.draw(g);
        }
    }
}

```

```
1
2   import java.awt.Color;
3   import java.awt.Point;
4
5   /*
5    * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.
6   txt to change this license
6    * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to
7   edit this template
8    */
9
10  /**
11   *
12   * @author luxformel
13   */
14  public class MainFrame extends javax.swing.JFrame {
14      //L'attribut lines est l'objet qui stockera toutes les lignes qui
15  seront
16      //à dessiner
17      private Lines lines;// = new Lines();
18
19      //L'attribut line sert à stocker une seule ligne
19      //C'est la ligne actuelle qui est en train d'être dessinée avec la
20  souris
21      private Line line;
22
23      /**
24       * Creates new form MainFrame
25       */
26      public MainFrame() {
27          initComponents();
28          //Instancier/créer l'objet lines
29          lines = new Lines();
30
31          //informer le drawPanel du nouvel objet lines
32          //Il faut faire cela, sinon rien ne sera dessiné
33          drawPanel.setLines(lines);
34      }
35
36      /**
36       * This method is called from within the constructor to initialize the
37  form.
37       * WARNING: Do NOT modify this code. The content of this method is
38  always
39       * regenerated by the Form Editor.
40       */
43      @SuppressWarnings("unchecked")
45
47
61
72
89
```

```
92
93
94      private void drawPanelMousePressed(java.awt.event.MouseEvent evt) {
95          //Lorsque le bouton de la souris est appuyé
96          line = new Line(evt.getPoint(), evt.getPoint(), Color.black);
97          lines.add(line);
98      }
99
100     private void drawPanelMouseReleased(java.awt.event.MouseEvent evt) {
101         line.setTo(evt.getPoint());
102         drawPanel.repaint();
103     }
104
105     private void drawPanelMouseDragged(java.awt.event.MouseEvent evt) {
106         line.setTo(evt.getPoint());
107         drawPanel.repaint();
108     }
109
110     /**
111      * @param args the command line arguments
112      */
113     public static void main(String args[]) {
135         /* Set the Nimbus look and feel */
136
137         /* Create and display the form */
138         java.awt.EventQueue.invokeLater(new Runnable() {
139             public void run() {
140                 new MainFrame().setVisible(true);
141             }
142         });
143     }
144
145     // Variables declaration - do not modify//GEN-BEGIN:variables
146     private DrawPanel drawPanel;
147     // End of variables declaration//GEN-END:variables
148 }
149
```