Université de Nice - Sophia Antipolis
Faculté des Sciences

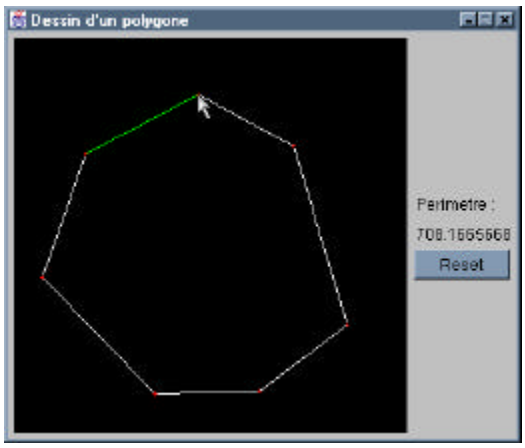
DEUG MIAS MP1

Programmation 2000-01

11. DESSINER UN POLYGONE

A. But de la séance

Dans ce TP nous allons rassembler les résultats des deux TP précédents : la création d'une classe polygone permettant de gérer la liste de ses sommets et la gestion des événements de la souris pour placer les points. Vous allez donc avoir besoin de la classe `Polygone.java`, faites en sorte d'en avoir un version à jour dans le même répertoire que celui où vous sauvegardez ce TP. L'interface finale ressemblera à ceci:



Préparez un fichier PolyDraw.java contenant :

```
import java.awt.*;
import java.awt.event.*;
import java.util.Vector;

class PolyDraw extends Frame
{

}

class PolyDrawCanvas extends Canvas
{

}

class PolygoneGraphique extends Polygone
{

}

class PointGraphique extends Point
{

}
```

La première classe sera notre fenêtre principale. La deuxième est notre canevas où l'on dessinera le polygone. La troisième étend la classe `Polygone` pour ajouter la méthode graphique permettant de l'afficher et la dernière fait de même avec la classe `Point` du package `awt`.

B. Mettre en place les composants graphiques

En vous inspirant des TP sur les interfaces graphiques et des exercices du contrôle continu, vous allez placer les composants graphiques pour obtenir une fenêtre ressemblant à la copie d'écran montrée précédemment.

Exercice 11.1 Modifiez le constructeur de la classe `PolyDraw` pour obtenir une fenêtre:
- ayant pour titre "Dessiner un polygone"
- avec un fond gris clair (`lightGray`)
- non redimensionnable
- s'affichant aux coordonnées 20, 30 de l'écran
- quittant l'application lorsque l'on appuie sur la petite case de fermeture
- et dont les composant suivent un `FlowLayout`

Exercice 11.2 Modifiez le constructeur de la classe `PolyDraw` pour ajouter à la fenêtre un Canvas `PolyDrawCanvas` et un panel. Le panel aura un fond gris clair et suivra un `GridLayout` en utilisant l'instruction:

```
setLayout(new GridLayout(3,1))
```

Ce layout dispose les composants suivant une grille de 3 cases sur une colonne. Vous ajouterez ensuite au panel:
- Un label fixe qui affichera la chaîne de caractères "`Perimetre :`"
- Un label pour qu'il faudra pouvoir modifier (ce sera donc une variable de la classe) et que l'on utilisera pour afficher la valeur du périmètre.
- Un bouton `Reset` dont on programmera le comportement plus tard

NB: N'oubliez pas d'ajouter l'instruction `this.pack();` pour que la fenêtre adapte sa taille à son contenu.

Exercice 11.3 Ajoutez à la classe `PolyDraw` une méthode `void changePerimeter(String p_text)` permettant de modifier la valeur du label affichant la longueur du périmètre ainsi que la méthode main :

```
public static void main(String[] args)
    {
      (new PolyDraw()).show();
    }
```

Exercice 11.4 Reprenez le code ci-dessous et complétez le constructeur du canevas pour :
- que l'on appelle le constructeur de la classe `Canvas`
- que l'on choisisse un fond noir
- que l'on copie la référence à la fenêtre `PolyDraw` dans la variable prévue à cet effet
- que l'on crée un nouveau `PolygoneGraphique` dont la référence sera placée dans la variable prévue à cet effet

```
class PolyDrawCanvas extends Canvas
{
    private PolyDraw myPolyDraw;
    PolygoneGraphique myPolygoneGraphique;

    PolyDrawCanvas(PolyDraw p_PolyDraw)
    {
       ...
    }

    public Dimension preferredSize() { return new Dimension(300,300); }
}
```

Compilez votre code, **CORRIGEZ VOS ERREURS TOUS SEULS COMME DES GRANDS** et vérifiez que l'interface ressemble bien à la copie d'écran en première page du sujet.

C. Préparer les classes Polygones et Points

Exercice 11.5 Nous allons compléter la classe Point:

```
class PointGraphique extends Point
{
  PointGraphique (int x, int y) {  ... // Appeler le constructeur de la classe Point   }


  public int getIntX() { ... }  // Renvoyer des abscisses entières en utilisant getX()
```

```
  public int getIntY() { ... }  // Renvoyer des ordonnées entières en utilisant getY()

  public void paint(Graphics p_Graphics)    // Ici on ajoute une méthode qui permet
    {                                       // d'afficher un point comme un disque rouge.
      p_Graphics.setColor(Color.red);       // On choisit la couleur rouge.
      p_Graphics.fillOval(getIntX()-1,getIntY()-1,3,3);   // On dessine un disque.
    }
}
```

Compilez pour vérifier que tout est syntaxiquement correcte (inutile d'exécuter le programme car rien n'a changé dans l'affichage)

Exercice 11.6 Nous allons compléter la classe Polygone:

Programmez les méthodes suivantes :

```
class PolygoneGraphique extends Polygone
{
       public PointGraphique getPointGraphiqueAt(int i)
       {      ... // renvoyer le i^eme PointGraphique en utilisant PointAt  }

       public void setLastPointGraphiqueAt(PointGraphique p)
       {      ... // remplacer le dernier point s'il existe }
}
```

Ajoutez la méthode d'affichage d'un polygone et commentez chaque ligne pour expliquer l'algorithme

```
public void paint(Graphics p_Graphics)
{
  int aNbPoints = size();
  if (aNbPoints>0)
    {
      PointGraphique aFirstPoint = getPointGraphiqueAt(0);
      PointGraphique aPreviousPoint = aFirstPoint;
      PointGraphique aCurrentPoint = aFirstPoint;
      for(int l_i=1; l_i<aNbPoints; l_i++)
        {
          aCurrentPoint = getPointGraphiqueAt(l_i);
          p_Graphics.setColor(Color.white);
          p_Graphics.drawLine(aPreviousPoint.getIntX(), aPreviousPoint.getIntY(),
                              aCurrentPoint.getIntX(), aCurrentPoint.getIntY());
          aPreviousPoint.paint(p_Graphics);
          aCurrentPoint.paint(p_Graphics);
          aPreviousPoint = aCurrentPoint;
        }
      p_Graphics.setColor(Color.green);
      p_Graphics.drawLine(aCurrentPoint.getIntX(),aCurrentPoint.getIntY(),
                          aFirstPoint.getIntX(),aFirstPoint.getIntY());
      aCurrentPoint.paint(p_Graphics);
      aFirstPoint.paint(p_Graphics);
    }
```

Compilez pour vérifier que tout est syntaxiquement correct (inutile d'exécuter le programme car rien n'a changé dans l'affichage)

D. Répondre aux événements

Tout comme au TP dernier nous allons détecter la pression du bouton de la souris et ses mouvements et pour cela nous avons besoin de nous créer un écouteur particulier puis de programmer la réponse aux événements qui nous intéressent : la pression d'un bouton et le mouvement de la souris lorsque le bouton est appuyé.

Exercice 11.7 Ajoutez la classe suivante :

```
class PolyDrawMouseListener extends MouseAdapter implements MouseMotionListener
{
       public void mouseMoved(MouseEvent e) {}
       public void mouseDragged(MouseEvent e) {}
}
```

Cette classe représente des écouteurs capables de répondre aux boutons (elle étend `MouseAdapter`) et aux mouvements de la souris (en implantant l'interface `MouseMotionListener`).

Exercice 11.8 Modifiez le constructeur du canevas pour ajouter le code suivant et complétez les morceaux manquants:

```
PolyDrawMouseListener aPolyDrawMouseListener = new PolyDrawMouseListener()
{ public void mousePressed(MouseEvent e)    // un bouton est appuyé
  {... // ajouter un GraphicPoint au polygone en récupérant les coordonnées de l'événement
       // mettre à jour l'affichage du canevas
  }

 public void mouseDragged(MouseEvent e)   // la souris a bougé avec un bouton enfoncé
  {... // modifier le dernier GraphicPoint du polygone pour suivre la souris
       // mettre à jour l'affichage du canevas
  }
};
this.addMouseListener(aPolyDrawMouseListener);
this.addMouseMotionListener(aPolyDrawMouseListener);
```

Exercice 11.9 Ajoutez au canevas la méthode affichant sont contenu:

```
public void paint(Graphics p_Graphics)
    {
      ...// afficher le polygone : une ligne suffit !!!
    }
```

Compilez et essayez le programme : vous devez pouvoir créer le polygone.

Exercice 11.10 Pour effacer le canevas vous allez programmer la réponse du bouton `Reset`. Pour cela il vous faut une méthode pour vider le polygone de ses points et rafraîchir l'affichage dans le canevas :

```
public void reset()  { ... }
```

Et dans le constructeur de la fenêtre (`PolyDraw`) ajoutez un écouteur qui appelle la méthode reset() du canevas.

```
ResetButton.addActionListener(new ActionListener()
       {  public void actionPerformed(ActionEvent e)
           {  ... }
       });
```

E. Calculs sur le polygone

Nous allons dans cette partie illustrer comment utiliser les objets polygones créés pour faire des calculs.

Exercice 11.11 En vous inspirant du code de la procédure d'affichage de la classe `GraphiquePolygone` (tout l'algorithme de parcours des cotés du polygone y est décrit) programmez une méthode de calcul du périmètre d'un polygone:

```
public double getPerimeter()
  { ... //
  }
```

Puis dans la méthode `paint()` du canevas mettez à jour la valeur du label prévu à cet effet en appelant la méthode de la fenêtre que vous avez préparée et en utilisant le résultat renvoyé par votre méthode.

Exercice 11.12 Calcul de l'enveloppe ?