Laboratoire POO avec Java *TP5*

Grapheur élémentaire

Maxime Burri - Stéphane Malandain – Michael Polla – 2016/2017 – filière ITI

Objectifs

Ce laboratoire répond aux objectifs suivants.

1) Lecture d'un fichier texte
2) Analyse syntaxique avec un `StringTokenizer`
3) Production d'un graphique simple

Reddition

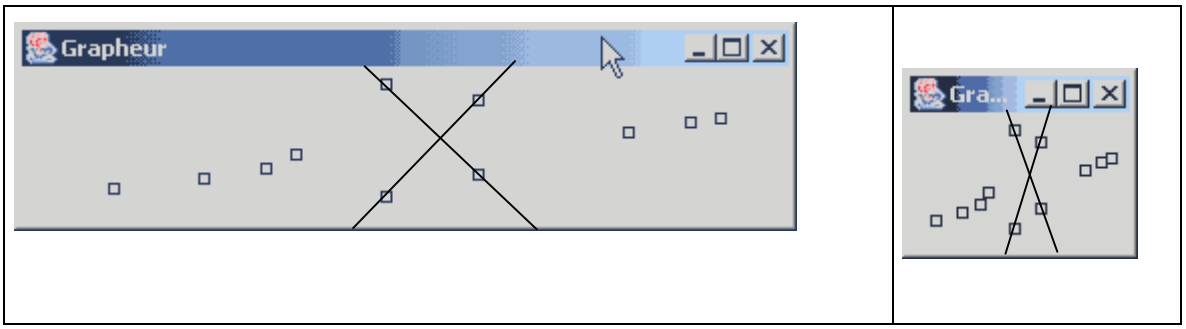
Vous devrez rendre le listing sur papier ET par courrier électronique à stephane.malandain@hesge.ch

Vous serez interrogé oralement et devrez faire une démonstration de votre programme.

I. Descriptif

Chaque ligne d'un fichier texte contient deux ou quatre nombres réels écrits sous forme littérale. Chaque ligne peut être analysée, d'abord en produisant des *tokens*, puis en faisant le *parsing* de ceux-ci pour voir s'ils se convertissent en nombres réels. Vous posséderez alors une collection de points affichables dans le plan en spécifiant une échelle. Si quatre valeurs figurent sur une ligne, on construit une droite qu'on dessine jusqu'au bord de la zone de dessin.

La figure ci-dessous vous montre deux aspects d'un même fichier de points et de droites, avant et après redimensionnement de la fenêtre.



II. Etapes de réalisation

A. Fichier de points

Un fichier rassemble des lignes de texte telles que :

`1.56 3.87` ou `1.36 6.87 10.23 8.3`

Prenez un éditeur de texte et créez quelques points selon la description ci-dessus.

B. Modèle pour un point

Réutiliser et adapter si nécessaire les classes `PointD2` et `DroiteD2` du TP4.

Les objets de ces classes devront être fabriqués à partir du contenu d'un fichier qui, sur chacune de ses lignes, possède des nombres sous forme littérale.

Vous devez lire ces chaînes de caractères et les transformer en valeur du type primitif `double` ou objet `Double` selon votre implémentation des classes `PointD2` et `DroiteD2`.

C. Lecture d'un fichier

Vous lirez une ligne du fichier avec un *stream* `LineNumberReader` et chaque ligne sera analysée avec un `StringTokenizer`. Une exception de « lecture » (syntaxe de ligne ou problème IO) mettra fin à l'exécution après avoir affiché le numéro de ligne erroné.

En fin de lecture, votre objet collection de points (instance de `ArrayList`) est prêt à être dessiné.

Java possède des objets pour lire les fichiers (paquetage `java.io`). Voici le principe :

```
. . .
LineNumberReader lecteurDeLignes = null;

try {
   lecteurDeLignes =
      new LineNumberReader(new FileReader(fname));

   String texteDeLigne = null;

   // lecture ligne après ligne
   while ((texteDeLigne = lecteurDeLignes.readLine()) != null) {

      // 'tokenisation' de la ligne
      StringTokenizer st = new StringTokenizer(texteDeLigne);
. . .
```

Une ligne syntaxiquement correcte contient 2 ou 4 *tokens* qui vous permettront d'instancier des objets de type `PointD2` ou `DroiteD2`.

Chacun des points instanciés à partir du fichier est ajouté à un vecteur : un objet `ArrayList()` du paquetage `java.util`. Cet objet sera itéré afin de placer les points dans un espace graphique.

D. Dessin

En Java, le dessin d'un objet ne peut avoir lieu qu'avec un objet qui comporte un `Graphics`. Ici, vous utiliserez un `JPanel`. Un `JPanel` permet de redéfinir la méthode

```
public void paintComponent(Graphics g);
```

qui est automatiquement appelée pour différents événements.

Voici une esquisse du travail à faire dans cette méthode :

```
. . .
itérer à travers l'ArrayList
   prendre un point UnPoint
   dessiner un drawRect() autour de ce point
fin itération
. . .
```

On dessine un rectangle avec `drawRect()` car un pixel est difficilement visible. On dessinera donc une « enveloppe » du point. Pour les droites, vous utiliserez `drawLine()`. On aimerait que les droites s'étendent jusqu'au bord de la zone de dessin. Consultez l'API à propos de la classe `Graphics`.

E. Dessin redimensionnable

La figure qui présente les captures d'écran de la fenêtre du grapheur, montre deux façons d'afficher une même collection de points. Vous pouvez donc remarquer que les points sont dessinés relativement à la dimension de la zone graphique (en ajustant la fenêtre, les points vont se redessiner automatiquement avec `paintComponent()`).

Vous allez donc

- calculer les extrêmes en X et les extrêmes en Y ;
- avec ces valeurs, vous calculerez l'étendue en X et l'étendue en Y.

Les données ci-dessus vous permettront de calculer un facteur d'échelle.

Pour avoir la `Dimension` du `Graphics` du `JPanel`, envoyez lui le message `getSize()`.

Les facteurs d'échelle en X et en Y peuvent être calculés selon

FactEchX = DimensionJPanelX / étendueX

FactEchY = DimensionJPanelY / étendueY

Alors chaque point sera dessiné avec son facteur d'échelle selon

NouveauX = (UnPoint.x – extrêmeX) * FactEchX

NouveauY = (UnPoint.y – extrêmeY) * FactEchY

Note 1 : introduisez un facteur de bord pour laisser une « bordure » entourant les points.

Note 2 : dans un premier temps limitez-vous en ne dessinant que les points.

Testez et consolidez votre programme afin que des cas douteux ne puissent pas conduire à une exécution erronée (signalez la cause, par exemple avec un `drawString()` pour le `JPanel`).