Afficher graphiquement les points et les segments

Objectifs :

- Utiliser la documentation javadoc
- Utiliser des paquetages
- Comprendre l'intérêt des interfaces comme outil de spécification
- Manipuler les interfaces

Exercice 1 : Utiliser l'afficheur graphique Écran

Le paquetage `afficheur`[1] propose une classe `Ecran` qui permet de dessiner sur un écran graphique des points, des lignes, des cercles et des textes. Notons que l'`Ecran` est une réalisation de l'interface `Afficheur` qui spécifie les méthodes que l'on doit trouver sur tout afficheur.

Écrire un programme (classe `ExempleEcran`) qui définit un écran de dimension 400 pixels en largeur et 250 en hauteur avec une unité de 15 pixels. Le titre de la fenêtre contenant l'écran sera `"ExempleEcran"`. Les axes seront dessinés. Sur cet écran seront dessinés :

- un point de couleur verte en position (1,2) ;
- une ligne entre les points (6,2) et (11,9) et de couleur rouge ;
- un cercle jaune de centre (4,4) et de rayon 2,5 ;
- le texte « Premier dessin » en bleu à la position (1, -2).

Remarque : Dans cet exercice, nous n'utilisons pas les classes `Point` et `Segment`.
Conseil : Il faut, comme toujours, compiler et exécuter régulièrement.

Exercice 2 : Afficher graphiquement le schéma

Compléter la classe `ExempleSchema1` pour que le schéma soit dessiné sur un écran de dimension 600x400 avec l'unité fixée à 20 pixels. On ajoutera une méthode `dessiner` dans la classe `Point` et dans la classe `Segment`.
Conseil : Avant de modifier la classe, il faut commencer par la compiler et l'exécuter.

1. Pour que ce paquetage soit accessible depuis le projet Java, il faut télécharger afficheur.jar depuis la page du module (rubrique « ressources »), le sauver dans un dossier, faire un clic droit sur la racine du projet Java pour choisir *Build Path / Add External Archives...* et sélectionner afficheur.jar. On peut aussi faire *Build Path / Configure Build Path...*, sélectionner l'onglet *Librairies* et faire *Add External JARs*.
   En ligne de commande, il faut utiliser l'option `-classpath` ou la variable d'environnement `CLASSPATH`.

Exercice 3 : Translater le schéma

Dans le programme de l'exercice 2, après avoir affiché et dessiné le schéma, ajouter les instructions pour :

1. le translater de 4 suivant l'axe des X et -3 suivant l'axe des Y (translater le schéma, c'est translater les trois segments et le barycentre) ;
2. l'afficher et le dessiner de nouveau.

Expliquer les résultats de l'exécution. On ne cherchera pas à corriger le programme.

Exercice 4 : Afficher le schéma en SVG

Le paquetage `afficheur` définit une deuxième réalisation de l'interface `Afficheur` appelée `AfficheurSVG`. Elle dessine le schéma en SVG [2], spécification définie pour représenter les dessins vectoriels.

Compléter le programme de l'exercice 2 pour afficher le schéma en SVG en utilisant la classe `AfficheurSVG`. On affichera le schéma sur la sortie standard et dans un fichier, par exemple "schema.svg". Ce fichier peut alors être ouvert avec inkscape ou firefox.

Exercice 5 : Afficher le schéma sous forme d'un texte explicite

L'objectif de cet exercice est de réaliser un nouvel afficheur pour dessiner un schéma sous une forme textuelle qui permet de facilement comprendre la nature des objets du schéma et leurs caractéristiques. Nous allons donc définir une nouvelle réalisation [3] de l'interface `Afficheur` du paquetage `afficheur`. Nous l'appellerons `AfficheurTexte`. Elle sera définie dans le même paquetage que les classes `Point` et `Segment`. La manière dont doivent être affichés les points, segments, cercles et textes est donnée ci-dessous en reprenant l'exemple écrit à l'exercice 1.

```
Point {
      x = 1.0
      y = 2.0
      couleur = java.awt.Color[r=0,g=255,b=0]
}
Ligne {
      x1 = 6.0
      y1 = 2.0
      x2 = 11.0
      y2 = 9.0
      couleur = java.awt.Color[r=255,g=0,b=0]
}
Cercle {
      centre_x = 4.0
      centre_y = 4.0
      rayon = 2.5
      couleur = java.awt.Color[r=255,g=255,b=0]
}
Texte {
```

2. SVG, Scalable Vector Graphics, est une spécification définie par le W3C qui s'appuie sur XML. On peut trouver plus d'informations à l'URL http ://www.w3.org/TR/SVG/.

3. Avec Eclipse, on peut faire *New / Class*, renseigner son nom, puis sélectionner l'interface qu'elle réalise en faisant *Add....* Si l'option *Inherited abstract methods* est cochée (c'est le cas par défaut), le squelette des méthodes de l'interface est inclu dans le texte de la classe engendrée. Il ne reste plus qu'à compléter... Si on ne l'a pas fait à la création, on peut ensuite choisir dans le menu contextuel *Source / Override/implement Methods...* ou utiliser les propositions de correction des erreurs.

```
        x = 1.0
        y = -2.0
        valeur = "Premier dessin"
        couleur = java.awt.Color[r=0,g=0,b=255]
}
```

Écrire la classe `AfficheurTexte` correspondant à cet afficheur et compléter le programme de l'exercice 2 pour l'utiliser.

Indication : L'instruction Java « `System.out.println(java.awt.Color.GREEN)` » affiche :

```
java.awt.Color[r=0,g=255,b=0]
```