TP PPO
Rectangle

Polytech'Lille GIS2A3

1 Prise en main du JDK (Java Development Kit)

— Ajouter à votre `PATH` (dans votre fichier `.bashrc`) le chemin du jdk1.8 :
`export PATH=/usr/local/jdk1.8.0_25/bin:$PATH`
Remarque : les autres versions sont disponibles sur : `/usr/local/jdkXX.`

— La documentation (javadoc) est sur (conserver le lien dans vos bookmarks) :
`file:///usr/local/jdk1.8.0/docs/index.html`
Pour obtenir la documentation sur le langage et les classes disponibles aller sur "API & Language Documentation / Java 2 Platform API Specification". La documentation est entièrement navigable. A gauche, vous pouvez choisir une navigation alphabétique de "All Classes" ou d'un package particulier. A droite apparaissent les informations sur la sélection, notez les options :
- "Tree" : hiérarchie des packages et des classes de la sélection
- "Index" : index alphabétique de tous les symboles associés (classes, variables, méthodes, constructeurs, ...)

— Observer comment sont décrites quelques classes vues en cours : `Object, String, Applet, ....`

— Observer comment est décrite la classe `Point2D.Double` qui sera utilisée dans la suite du TP.

2 Rectangles

Creér un répertoire "`tp1`" puis travailler dans un répertoire "`rectangles`" que vous aurez bien évidemment mis dans `tp1`. Par ailleurs, veiller à bien respecter le nom des classes, méthodes, variables d'instance donnés dans l'énoncé.

2.1 La classe `Rectangle`

Programmer une classe `Rectangle` (dans un fichier `Rectangle.java`) telle que celle vue en cours :

— un rectangle est représenté par un couple de points "`origin`" (premier point) et "`corner`" (deuxième point). On utilisera la classe `Point2D.Double` du package `java.awt.geom` (voir la javadoc).
— programmer un constructeur paramétré par les coordonnées des 2 points origine et corner. (Pour que la section 3 fonctionne, il faut que la signature de votre constructeur soit `public` et donc
`public Rectangle (double x1, double y1, double x2, double y2).`
Il en sera de même pour toutes vos méthodes et vos variables d'instance.)
— programmer les méthodes : `largeur(), longueur(), surface(), perimetre()`.

Compiler régulièrement cette classe afin de vérifier que votre code compile en exécutant la commande suivante : `javac Rectangle.java`

2.2 La classe `ApplicationRectangle`

Programmer une classe principale `ApplicationRectangle` munie d'une méthode `main` qui :
— instancie un rectangle de coordonnées fixées, par exemple :`10.0 10.0 40.0 50.0`
— affiche ses caractéristiques : largeur, longueur, surface, périmètre.

Compiler cette classe (normalement, vous n'avez pas besoin de recompiler `Rectangle`) puis exécuter la en tapant respectivement les commandes suivantes :
```
javac ApplicationRectangle.java
java ApplicationRectangle
```

2.3 `toString()`

Programmer une méthode `toString()` dans la classe `Rectangle` qui renvoie sous forme de chaine de caractères son couple de points caractéristiques : "`(<origine> , <corner>)`". `<origine>` et `<corner>` correspondent à la représentation sous forme de chaine de caractères des points renvoyée par leur propre méthode `toString()` (voir la documentation), laissez les s'afficher comme ils veulent !

Ajouter dans la classe `ApplicationRectangle` l'affichage du rectangle par appel automatique à `toString()`.

Recompiler les classes et exécuter à nouveau `ApplicationRectangle` en exécutant les mêmes commandes que précédemment.

2.4 Paramètres du `main`

Dans la classe `ApplicationRectangle`, utiliser les paramètres du main pour récupérer un quadruplet de coordonnées, instancier le rectangle correspondant et le tester.

Comme toujours, les paramètres du `main` sont des chaînes de caractères, encodant ici des doubles (''`10.0`'' par exemple) qu'il faut transformer en valeurs (`10.0`). Pour cela utiliser la méthode static `Double.parseDouble(String s)` de la classe `Double`, wrapper de `double`.

2.5 Utilisation de `Scanner sur System.in`

Créer une nouvelle classe `ManipulateurRectangle` avec une méthode `creerRectangle()` qui demande à l'utilisateur les coordonnées de 2 points origine et corner, instancie le rectangle correspondant et le retourne en résultat.

2.6 Tableau de rectangles

Modifier la classe `ManipulateurRectangle` comme suit :

— ajouter un attribut `tabRect` de type tableau de rectangles

— Ajouter une méthode `creerTableauRectangle()` qui permet de créer un tableau de rectangles en demandant à l'utilisateur le nombre de rectangles à créer (taille du tableau) et en le remplissant par appel itéré à la méthode `creerRectangle()`.

— Ajouter une méthode `toString()` qui renvoie sous forme de chaîne de caractères le tableau de rectangles ainsi créés (par appel à leur méthode `toString()`).

— Ajouter une méthode : `Rectangle max()`
qui renvoie le rectangle de plus grande surface du tableau tabRect.

— Ajouter une méthode `decalerRectangles(double, double)` qui décale tous les rectangles de `x` en abscisse et `y` en ordonnée (déplacement de l'origine et du corner). On pourra éventuellement ajouter d'autres méthodes si nécessaire.

— Tester ces méthodes dans le main.

3 Tester votre travail

Afin de savoir si votre code est de qualité et correspond à ce qui vous est demandé, il vous est possible de le tester. Pour cela, copier sur votre compte dans le répertoire `tp1` les répertoires `test` et `bib` :

```
cp -r ~aetien/public/PPO/tp1/lib .
cp -r ~aetien/public/PPO/tp1/test .
```

Mettez vous dans votre répertoire `tp1/test` et exécutez la commande suivante :

```
./runTest.sh
```

Si votre code vérifie tous les critères de qualité évalués, vous aurez un message du genre `OK (22 tests)`. Sinon, vous aurez des messages vous indiquant les erreurs qui peuvent exister dans votre code.