TP 01 :
Points et segments

Table des matières

Remarque : Ceci est le document réponse, il ne reprend pas l'intégralité du sujet.

1. Exercice 1 : Comprendre la classe Point

1.1. Squelette d'une classe Java

Expliquer comment on obtient le squelette d'une classe Java à partir de son diagramme d'analyse UML.
Les commandes deviennent des méthodes, les requêtes deviennent soit des attributs privés avec leur accesseur, soit des méthodes.

1.2. Convention en Java

Indiquer, s'il y en a, les entorses par rapport aux conventions de programmation Java.
Je remarque que la classe `Point` a ses arguments définis à la fin de la classe, alors que par convention, ils devraient l'être au tout début, avant le(s) constructeur(s).

2. Exercice 2 : Compiler et exécuter

Dans cet exercice, nous allons utiliser les outils du JDK (Java Development Kit) pour compiler et exécuter une application.

2.1. Lire et comprendre le programme ExempleComprendre

Lire le programme `ExempleComprendre` et dessiner l'évolution de l'état de la mémoire au cours de l'exécution du programme. Ceci doit être fait sur le listing, avant toute exécution du programme.

</> Code 1 : ExempleComprendre simplifié

```
1  Point p1;
2  p1 = new Point(3, 4);
3  Point p2 = new Point(0, 0);
4  double d = p1.distance(p2);
5  p1.translater(6, -2);
6  p1.setX(0);
7  p1.setY(10);
8  Point p3 = p1;
9  p3.translater(100, 100);
10 p3 = new Point(123, 321);
11 p1 = p2 = p3;
12 p1.translater(-123, -321);
13 d  = new Point(5, 5).distance(new Point(8, 1));
```

Je me base sur les lignes du Code1 pour le tableau suivant (les valeurs en rouge sont celles qui changent) :

ligne n°	p1	p2	p3	d
2	(3,4)	...	...	...
3	(3,4)	(0,0)	...	...
4	(3,4)	(0,0)	...	5
5	(9, 2)	(0,0)	...	5
6	(0, 2)	(0,0)	...	5
7	(0, 10)	(0,0)	...	5
8	(0, 10)	(0,0)	(0, 10)	5
9	(100, 110)	(0,0)	(100, 110)	5
10	(100, 110)	(0,0)	(123, 321)	5
11	(123, 321)	(123, 321)	(123, 321)	5
12	(0, 0)	(0, 0)	(0, 0)	5
12	(0, 0)	(0, 0)	(0, 0)	5

2.2. Compiler ExempleComprendre

Compiler le programme `ExempleComprendre` ***fourni.***

```
> javac ./ExempleComprendre.java
```

2.3. Exécution de ExempleComprendre

Exécuter le programme `ExempleComprendre` **.**

```
> java ./ExempleComprendre
```

2.4. Vérifier les résultats

Vérifier que l'exécution donne des résultats compatibles avec l'exécution à la main réalisée à la question 2.1

Résultats :

</> Résultats de compilation de ExempleComprendre

```
CONSTRUCTEUR Point(3.0, 4.0)
p1 = (3.0, 4.0)
CONSTRUCTEUR Point(0.0, 0.0)
p2 = (0.0, 0.0)
Distance de p1 a  p2 : 5.0
> p1.translater(6, -2);
p1 = (9.0, 2.0)
> p1.setX(0);
p1 = (0.0, 2.0)
> p1.setY(10);
p1 = (0.0, 10.0)
> Point p3 = p1;
p3 = (0.0, 10.0)
p1 = (0.0, 10.0)
> p3.translater(100, 100);
p3 = (100.0, 110.0)
p1 = (100.0, 110.0)
> p3 = new Point(123, 321);
CONSTRUCTEUR Point(123.0, 321.0)
p3 = (123.0, 321.0)
p1 = (100.0, 110.0)
> p1 = p2 = p3;
p1 = (123.0, 321.0)
p2 = (123.0, 321.0)
p3 = (123.0, 321.0)
> p1.translater(-123, -321);
p1 = (0.0, 0.0)
p2 = (0.0, 0.0)
p3 = (0.0, 0.0)
CONSTRUCTEUR Point(5.0, 5.0)
CONSTRUCTEUR Point(8.0, 1.0)
d = 5.0
```

Cela correspond au tableau donnée en réponse de la question 2.1.

2.5. Exécuter avec Java Tutor

2.6. Corriger le programme ExempleErreur

Compiler le programme `ExempleErreur` ***. Le compilateur refuse de créer le point à attacher à p1. Est-ce justifié ? Expliquer pourquoi.***

En effet, cela est justifié, car à la ligne 10 : `Point p1 = new Point();`, le programme `ExempleErreur`

essaie de créer un point sans arguments par défaut, alors que la classe `Point` a défini un nouveau constructeur qui remplace celui par défaut. Il faut donc obligatoirement donner en paramètre des valeurs de `x` et `y`.

2.7. Durée de vie des objets

Supprimer dans le fichier `Point.java` ***les commentaires devant l'affichage dans le constructeur et les commentaires à la C (/* */) autour de la méthode*** `finalize`***.***
Compiler et exécuter le programme `CycleVie`***. Est-ce que les points sont « détruits » ?***
Non, les points ne semblent pas être détruits. ***Augmenter le nombre d'itérations (5000, 50000, 500000, etc) jusqu'à ce que les messages de « destruction » apparaissent lors de l'exécution du programme.***
Les points ont commencé à se détruire après un grand nombre de points générés (plus de 100 000).

3. Exercice 3 : Produire la documentation

3.1. Produire la documentation des classes

Utiliser la commande `> javadoc` ***pour produire la documentation des classes de notre application. Consulter la documentation ainsi engendrée.***

3.2. Consulter la documentation de la classe Segment

En consultant la documentation engendrée pour la classe `Segment`***, indiquer le sens des paramètres de la méthode*** `translater` ***et le but de la méthode*** `afficher`***.***

3.3. Consulter la documentation en ligne

4. Exercice 4 : Comprendre et compléter la classe Segment

5. Exercice 5 : Définir un schéma particulier