**TP1 Points et segments**

**Exercice 1 : Comprendre la classe Point**

* 1. Pour obtenir le squelette d’une classe Java on suit ces étapes :
* Identifier le nom de la classe
* Identifier le constructeur ainsi que ses paramètres
* Identifier les attributs et leur portée (privé, public, protégé) ainsi que le type
* Identifier les méthodes, les paramètres et les types de retour
  1. Il existe des entorses par rapport aux conventions de nommage des variables et des classes. Java a une mauvaise gestion des exceptions

**Exercice 2 : Compiler et exécuter**

**2.1** l’évolution de l’état de la mémoire au cours de l’exécution du programme :

* Ligne 20-21 : Déclaration d’in objet de la classe Point et initialisation
* Ligne 25 : Affichage de l’objet p1 en utilisant la méthode Afficher définit de la ligne 11 à 15.
* Ligne 28-30 : Déclaration et affichage d’un objet p2 de la classe Point.
* Ligne 33-34 : Calcul et affichage de la distance entre les points p1 et p2 stocké dans une variable d, de type double.
* Ligne 37-39 : Translation du point p1 suivant 6 et -2 respectivement sur l’axe des abscisses et des ordonnées.

Ce qui est affiché est : « > p1.translater(6, -2) ; » puis p1 = (9,2)

* Ligne 42-44 : Changement de l’abscisse de p1 et affichage de p1.

Ce qui est affiché est : « > p1.setX(0) ; » puis p1 = (0,2)

* Ligne 47-49 : Changement de l’ordonnée de p1 et affichage de p1.

Ce qui est affiché est : « > p1.setY(10) ; » puis p1 = (0,10)

* Ligne 52-55 : Déclaration d’un point p3 à l’image de p1 et affichage de p1 et p3.

Ce qui est affiché est : « > Point p3 = p1 ; » puis p3 = (0,10) et p1 = (0,10)

* Ligne 58-63 : Translation de p3 et affichage de p3 et p1.

Ce qui est affiché est : « > p3.translater(100, 100) ; » puis p3 = (100,110) et p1 = (100,110)

* Ligne 67-70 : Re déclaration de p3 et affichage de p3 et p1

Ce qui est affiché est : « > p3 = new Point (123, 321) ; » puis p3 = (123,321)

et p1= (100,110)

* Ligne 72-77 : Affectation de l’objet p3 à p2 et p1 puis affichage de p1, p2 et p3.

Ce qui est affiché est : « > p1 = p2 = p3 ; » puis p1 = (123,321) et p2 = (123,321)

et p3 = (123,321)

* Ligne 79 : p1, p2 et p3 sont des points différents qui ont les mêmes coordonnées.

Il y aura donc trois points superposés sur les mêmes coordonnées.

* Ligne 80-84 : Translation de p1 et affichage de p1, p2 et p3.

Ce qui est affiché est : « > p1.translater(-123, -321); » puis p1 = (0,0) et p2 = (0,0)

et p3 = (0,0)

* Ligne 86-87 : Déclaration de deux objets Point de coordonnées respectives (5,5) et (8,1).

Calcul de la distance entre ces deux points stockés dans la variable d.

Affichage de cette distance.

* Ligne 89 : Il y a trois points accessibles qui sont p1, p2 et p3.

**2.3** voir dépôt git

**2.4**

**2.5**

**2.6** Le programme ne fonctionne pas parce qu’on appelle un constructeur sans paramètres alors qu’il n’en existe pas dans la classe Point.

L’intérêt d’un tel comportement est de pouvoir avoir plusieurs constructeurs pour une même classe et utilisé celui qui nous arrange en fonction de la situation.

**2.7**

**Exercice 3 : Produire la documentation**

**3.1**

**3.2**

**La méthode Translater**

Le paramètre dx permet de faire une translation du segment suivant : l’axe des X

Le paramètre dy permet de faire une translation du segment suivant : l’axe des Y

**La méthode afficher**

Cette méthode permet d’afficher le segment sous le format **[extrémité 1 – extrémité 2]**

**3.3**

**Exercice 4 : Comprendre et compléter la classe Segment**