UNIVERSITÉ CAEN NORMANDIE

UFR des Sciences

Université de Caen Normandie
Département mathématiques-informatique
2022–2023
Bruno Zanuttini

L2 informatique

TP. Géométrie

Unité SMINFL3A : Programmation Java orientée objet

L'objectif de ce TP est d'écrire deux classes simples permettant de manipuler des positions (points) et des segments dans l'espace euclidien.

Note importante Compiler toutes les classes déjà écrites après l'écriture de chaque méthode.

1 Positions

Exercice 1. *Créer un* package *nommé* `geometry`*.*

Exercice 2. *Dans le* package `geometry`*, écrire une classe nommée* `Position` *avec pour attributs deux coordonnées entières, pour l'abscisse et l'ordonnée, et un constructeur permettant de les initialiser.*

Exercice 3. *Ajouter à la classe des méthodes nommées* `getX` *et* `getY`*, qui retournent l'abscisse et l'ordonnée de la position, respectivement.*

Exercice 4. *Toujours dans le* package `geometry`*, écrire une classe exécutable, dont la méthode* `main` *instancie une position de coordonnées* $(3, 4)$*, puis affiche les coordonnées de cette position.*

Exercice 5. *Compiler et exécuter.*

Exercice 6. *Ajouter à la classe* `Position` *une méthode nommée* `getRepresentation`*, qui retourne une représentation de la position sous forme de chaîne de caractères.*

Exercice 7. *Ajouter à la méthode* `main` *de la classe exécutable l'affichage de la représentation de la position.*

Exercice 8. *Compiler et exécuter.*

Exercice 9. *Ajouter à la classe* `Position` *une méthode nommée* `symmetricX`*, qui crée et retourne la position symétrique par rapport à l'axe des abscisses.*

Exercice 10. *Ajouter un appel de la méthode* `symmetricX` *à la méthode* `main` *de la classe exécutable, ainsi qu'un affichage du résultat.*

Exercice 11. *Compiler et exécuter.*

Exercice 12. *Ajouter à la classe* `Position` *une méthode nommée* `translate`*, qui prend en arguments deux entiers,* δ_x *et* δ_y*, et modifie la position en la translatant selon le vecteur* (δ_x, δ_y)*.*

Exercice 13. *Tester les méthodes écrites en utilisant la librairie de tests fournie sur la page du cours. Pour cela, toujours dans le* package `geometry`*, écrire une classe exécutable, en utilisant par exemple le code :*

```
import geometrytests.PositionTests;
[...]
boolean ok = true;
PositionTests positionTester = new PositionTests();
```

```
ok = ok && positionTester.testGetX();
ok = ok && positionTester.testGetY();
ok = ok && positionTester.testSymmetricX();
ok = ok && positionTester.testTranslate();
System.out.println(ok ? "All␣tests␣OK" : "At␣least␣one␣test␣KO");
```

Compiler et exécuter cette classe.

2 Segments

Pour cette partie, continuer à créer toutes les classes dans le *package* `geometry`.

Exercice 14. *Écrire une classe nommée* `Segment`*, avec pour attributs deux positions (les extrémités du segment). Ajouter à la classe un constructeur prenant deux instances de la classe* `Position` *en arguments, et qui initialise les attributs.*

Exercice 15. *Ajouter à la classe* `Segment` *une méthode nommée* `getRepresentation`*, qui retourne une représentation du segment sous forme de chaîne de caractères.*

Exercice 16. *Écrire une classe exécutable, dont la méthode* `main` *instancie un segment d'extrémités* $(3, 4)$ *et* $(7, 7)$*, et affiche sa représentation.*

Exercice 17. *Compiler et exécuter.*

Exercice 18. *Ajouter à la classe* `Segment` *une méthode nommée* `length`*, qui retourne la longueur du segment sous la forme d'une valeur de type* `double`*.*
Indication : *On pourra utiliser la méthode* `Math.sqrt`*, qui retourne une valeur de type* `double` *représentant la racine carrée de son argument.*

Exercice 19. *Pour tester, compléter la méthode* `main` *de la classe exécutable écrite à l'exercice 13 en s'inspirant du code :*

```
boolean ok = true;
SegmentTests segmentTester = new SegmentTests();
ok = ok && segmentTester.testLength();
System.out.println(ok ? "All␣tests␣OK" : "At␣least␣one␣test␣KO");
```

Compiler et exécuter.
Indication : *Ajouter également l'import :* `import geometrytests.SegmentTests;`.