TP3

Exercice – coordonnées cartésiennes

En coordonnées cartésiennes, la position d'un point dans le plan est représentée par son abscisse et son ordonnée.

1. Créez une classe `Point` permettant de modéliser un tel point. Pensez à bien documenter votre code et à utiliser l'auto-complétion [Ctrl] + [Espace] pour vous aider et gagner du temps.
2. Écrivez un constructeur qui initialise un point à l'abscisse et l'ordonnée passées en paramètre.
3. Écrivez un constructeur par défaut (*i.e.*, sans paramètre) qui initialise un point aux coordonnées (0,0), en utilisant le chaînage de constructeurs.
4. Ajoutez les accesseurs en lecture et en écriture. Faîtes un clic-droit dans la fenêtre de code, sélectionnez *Source*, puis *Generate Getters and Setters...*. Dans la fenêtre qui s'ouvre, sélectionnez les attributs concernés et cliquez sur *OK*. Les constructeurs peuvent être générés de la même manière.
5. Redéfinissez la méthode `toString` afin que la chaîne de caractères retournée soit de la forme `(`*`abscisse`*`, `*`ordonnée`*`)`.
6. Redéfinissez la méthode `equals` afin qu'elle retourne `true` si l'instance d'`Object` passée en paramètre est de type `Point` et que son abscisse et son ordonnée sont égales à celles de l'objet receveur, sinon retourne `false`.
7. Écrivez une méthode d'instance `distance` qui retourne la distance entre l'objet receveur et un autre point passé en paramètre. Consultez la documentation de la classe `Math` pour l'utilisation des fonctions mathématiques.
8. Écrivez une méthode d'instance `translater` qui déplace l'objet receveur suivant un vecteur passé en paramètre.
9. Créez une classe `TestPoint` avec une méthode `main` qui :
   (a) Crée deux points distincts,
   (b) Affiche les deux points dans la console,
   (c) Affiche la distance entre les deux points,
   (d) Déplace un des points vers l'autre,
   (e) Compare les deux points en utilisant `equals`, puis `==`,
   (f) Crée une référence à un des points,
   (g) Fait appel à un accesseur en écriture sur cette référence,
   (h) Affiche de nouveau les deux points et la référence.

Exercice – coordonnées polaires

Nous proposons de changer la représentation mémoire de la classe `Point` en utilisant un système de coordonnées polaires au lieu de coordonnées cartésiennes. En coordonnées polaires, la position d'un point dans le plan est représentée par une distance et un angle (compris entre $]-\pi;\pi]$).

1. Donnez la liste des méthodes de la classe `Point` dont le code doit obligatoirement être mis à jour suite au changement de représentation mémoire.
2. Donnez les nouvelles implémentations des méthodes sans changer leur entête (*i.e.*, paramètres et type de retour) ni leur sémantique.
3. Exécutez de nouveau la classe `TestPoint`.
4. Ajoutez une méthode d'instance `rotation` qui fait tourner l'objet receveur autour de l'origine d'un angle passé en paramètre.
5. Complétez la méthode `main` de la classe `TestPoint` pour tester le comportement de cette méthode et vérifier qu'elle n'altère pas la cohérence de l'objet receveur.

Exercice – segment

1. Créez une classe `Segment` permettant de modéliser un segment dans le plan en utilisant la classe `Point`.
2. Ajoutez des constructeurs et des accesseurs pour cette classe.
3. Écrivez un constructeur par recopie superficielle pour cette classe.
4. Écrivez une méthode `main` qui :
   (a) Crée deux points,
   (b) Crée un segment dont les extrémités sont ces points,
   (c) Affiche les extrémités du segment dans la console,
   (d) Déplace les deux points,
   (e) Affiche les deux points,
   (f) Affiche de nouveau les extrémités du segment.
5. Exécutez la classe.
6. Écrivez un constructeur par recopie profonde pour cette classe.
7. Exécutez de nouveau la classe.