TD1 : Bases de la programmation objet

Sébastien Jean

Représentation objet d'un point

On cherche à écrire une classe `Point2D`, représentant un point dans un repère xOy. On considère les *spécifications*[1] suivantes :

1. Un point est caractérisé par son abscisse et son ordonnée, nommées respectivement `x` et `y`. Les valeurs de ces coordonnées sont des nombres réels. Les coordonnées sont invisibles depuis l'extérieur de la classe, et ne sont plus modifiables une fois leurs valeurs fixées.
2. Il existe une valeur par défaut (égale à 0) pour chaque coordonnée
3. On peut obtenir un nouveau point de deux manières :
   — en ne donnant pas d'information particulière : on obtient dans ce cas le point à l'origine du repère dont les valeurs des coordonnées sont nulles
   — en donnant les valeurs des coordonnées
4. On peut obtenir les valeurs des deux coordonnées séparément
5. On peut obtenir une représentation texte d'un point sous la forme `(x, y)`

Application utilisant les points

On cherche à écrire, dans une classe `UneApplicationUtilisantDesPoints`, une application qui :

— Crée un nouveau point d'abscisse et ordonnée respectivement égales à -3.54 et 7.12
— Stocke la référence de l'objet correspondant dans une variable nommée `unPoint`
— Affiche sur la sortie standard les coordonnées de ce point
— Crée un nouveau point situé à l'origine du repère
— Stocke la référence de l'objet correspondant dans une variable nommée `origine`
— Affiche sur la sortie standard les coordonnées de ce point

Représentation objet d'un vecteur

On cherche à écrire une classe `Vecteur2D`, représentant un vecteur en 2D (au sens géométrique). On considère les *spécifications* suivantes :

1. Un vecteur est caractérisé par les deux points qui en forment les extrémité nommés respectivement `debut` et `fin`. Ces points sont invisibles depuis l'extérieur de la classe, et ne sont plus modifiables une fois leurs valeurs fixées.
2. On peut obtenir un nouveau vecteur de trois manières :
   — en ne donnant pas d'information particulière : on obtient dans ce cas le vecteur nul dont les extrémités sont confondues (et le point correspondant est à l'origine)

1. Le terme *spécifications* est ici un peu abusif, il s'agit plutôt d'indications

— en donnant ses extrémités

— en donnant son extrémité `debut` et le décalage `dx` et `dy` de l'extrémité `fin` par rapport à `debut`

3. On peut obtenir ses extrémités séparément

4. On peut obtenir la norme du vecteur, égale à $\sqrt{cx^2 + cy^2}$.
   cx est la composante du vecteur sur l'axe Ox et égale à $x_{fin} - x_{debut}$.
   `cy` est la composante du vecteur sur l'axe Oy et égale à $y_{fin} - y_{debut}$.

   On pourra utiliser la méthode permettant de calculer la racine carrée d'un nombre et définie dans la classe `Math` *avec la signature suivante :* `public static double sqrt(double n)`

5. On peut savoir si le vecteur (de composantes `cx` et `cy`) est colinéaire à un autre vecteur (de composantes `cx'` et `cy'`), c'est à dire si $cx * cy' = cx' * cy$.

6. On peut savoir si le vecteur (de composantes `cx` et `cy`) est orthogonal à un autre vecteur (de composantes `cx'` et `cy'`), c'est à dire si $cx * cx' + cy * cy' = 0$
7. On peut obtenir une représentation texte d'un vecteur sous la forme `(cx, cy) : norme`
8. On peut obtenir une représentation texte d'un vecteur sous la forme `[ debut -> fin ]`

Application utilisant les vecteurs

On cherche à écrire, dans une classe `UneApplicationUtilisantDesVecteurs`, une application qui :

— Crée un nouveau vecteur de composantes respectivement égales à -3.54 et 7.12 (de façon quelconque)

— Stocke la référence de l'objet correspondant dans une variable nommée `unVecteur`

— Affiche sur la sortie standard la norme de ce vecteur précédée du message `la norme du vecteur est égale à :`

— Crée un nouveau vecteur nul

— Stocke la référence de l'objet correspondant dans une variable nommée `vecteurNul`

— Affiche sur la sortie standard si les deux vecteurs précédents sont colinéaires et/ou orthogonaux

— Crée un nouveau vecteur à partir des points `(3.54, 0)` et `(0, 7.12)`

— Stocke la référence de l'objet correspondant dans une variable nommée `unAutreVecteur`

— Affiche sur la sortie standard si le vecteur précédent est colinéaire et/ou orthogonal au premier vecteur créé

Translation d'un point

On souhaite enrichir la classe `Point2D`, en ajoutant notamment la possibilité d'obtenir un nouveau point résultant de la translation d'un point `(x, y)` par un décalage `(dx, dy)`, par conséquent le point de coordonnées `(x+dx, y+dy)`.

Ajouter la méthode correspondante dans la classe `Point` et modifier l'application précédente en ajoutant l'affichage des coordonnées d'un point résultant de la translation du point `(-3.54, 7.12)` par un décalage de `(3.54, -7.12)`