Programmation orientée objet
ING 1 TD/TP n°1

1. Lecture d'une classe : Présentation de la Classe PointCartesien

Nous allons voir dans cette série d'exercices la manipulation d'objets en Java. Nous allons utiliser une classe appelée PointCartesien qui permet de définir un point dans un plan à deux dimensions à partir de son abscisse et de son ordonnée.

(le source `PointCartesien.java`)

Ouvrez un éditeur comme **IntellijIdea** par exemple pour visualiser le code source de cette classe. Il y a 3 constructeurs différents pour cette classe, donnez ces 3 constructeurs avec leurs éventuels paramètres. Comment fonctionnent-ils ?

Donnez les méthodes dites « accesseurs » avec leurs différents paramètres ? Donnez les autres méthodes dites « de classe » ? Combien y en a-t-il ?

Quelle est l'utilité de la méthode toString() de la classe PointCartesien ?

2. Utilisation d'une classe : Utilisation de PointCartesien.

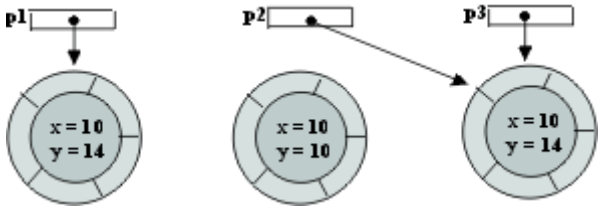
Soit la suite d'instructions java suivante :

```
PointCartesien     p1;
p1     = new PointCartesien();
PointCartesien     p2= new PointCartesien(10,10);
PointCartesien     p3= new PointCartesien(p2);
PointCartesien     p4     =      p3;
p4.positionner(14,14);
System.out.println ("p3="+ p3.toString());
p2.translation(4,4);
System.out.println("p1=" + p1 + ",p2=" + p2 + ",p3=" + p3 + ", p4=" + p4);
System.out.println ("\"p2==p3\"="+(p2==p3));
System.out.println ("\"p3==p4\"="+(p3==p4));
System.out.println("\"p2.egaleA(p4)\"=" +p2.egaleA(p4));
System.out.println ("\"p1=p3\"=" + (p1=p3));
```

a) Précisez quel est l'état du système après chaque étape en terme de références et d'objets existants et quels sont les affichages produits par les différentes instructions `System.out.println`.

b) Que se passe-t-il si on insère après l'étape 1 l'instruction : `p1.positionner(10,10);`

c) Quelles séquences d'instructions java permettent-elles d'atteindre l'état suivant, sachant que les variables p1, p2 et p3 sont affectées dans cet ordre et que les objets ont été créés dans l'ordre de gauche à droite ?

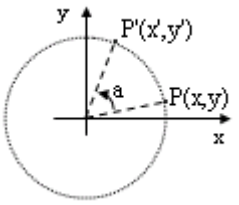


3.Modification de la classe PointCartesien

a) Écrire le corps de la méthode `rotation(double a)` qui fait effectuer au point une rotation d'un angle a par rapport à l'origine.

Rappel : la rotation d'un point P(x,y) d'un angle a par rapport à l'origine est un point P'(x',y') tel que:

```
x' = x . cos(a) – y . sin(a)
y' = x . sin(a) + y . cos(a)
```

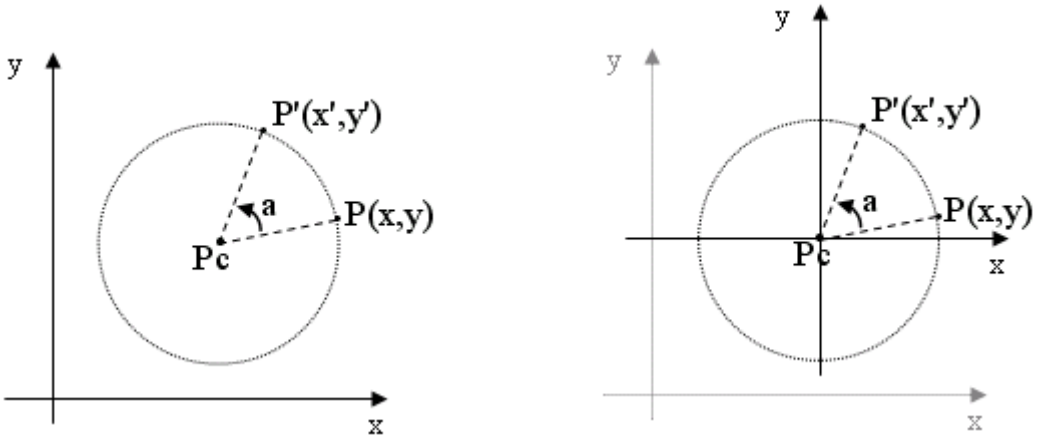


Écrire un programme créant un point p1 de coordonnée (10.0,13.0), afficher ce point, lui faire subir une rotation de +0.5 radians et afficher le point résultant.

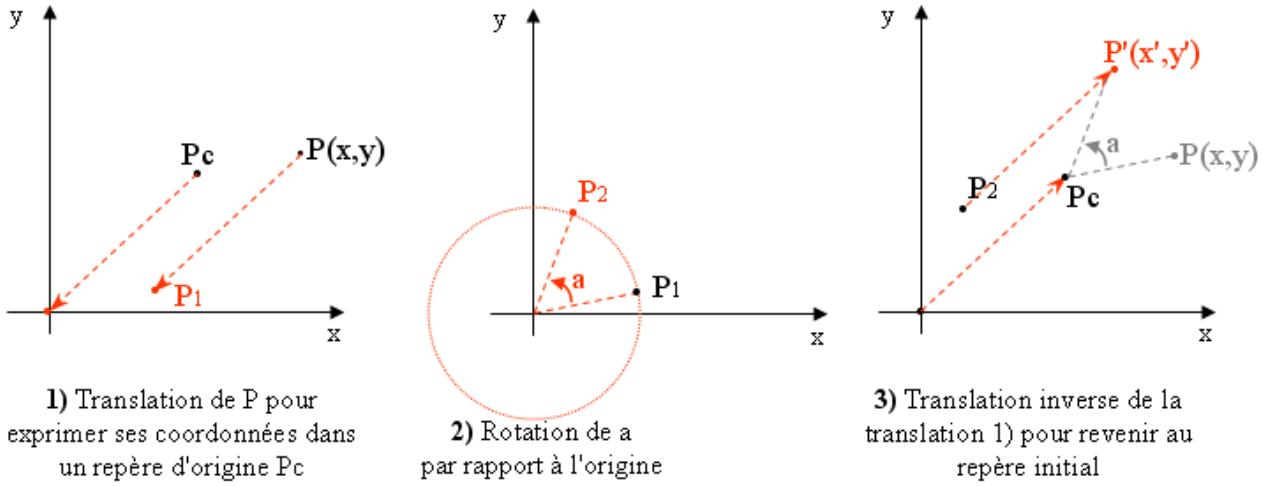
b) Écrire le corps de la méthode `rotation(PointCartesien pc, double a)` qui fait effectuer au point une rotation d'un angle a dont le centre est le point référencé par pc

Pour calculer les coordonnées (x',y') d'un point P de cordonnée (x,y) affecté par une telle rotation, on peut décomposer cette opération en trois étapes :

1. Changement de repère pour placer l'origine en Pc
2. Rotation par rapport à l'origine
3. Changement de repère pour replacer l'origine à sa position initiale.



Cette opération peut être réalisée en utilisant les opérations de translation et de rotation par rapport à l'origine.



1) Translation de P pour exprimer ses coordonnées dans un repère d'origine Pc

2) Rotation de a par rapport à l'origine

3) Translation inverse de la translation 1) pour revenir au repère initial

c) Ajouter à la classe `PointCartesien` une méthode copie qui renvoie une copie d'un point cartésien.

Tester cette méthode en créant un point p1 de coordonnée (10.0,13.0), afficher ce point, et créer une copie de p1 dans un point p2.

d) Ajouter à la classe `PointCartesien` une méthode `distanceAB(PointCartesien p)` qui renvoie la distance entre un point et un autre point.

Tester cette méthode en créant deux points p1 et p2 de coordonnées (10,14) et (5,5). Afficher ces deux points et la distance entre ces deux points.

```
p1 = PointCartesien{10.0,14.0}
p2 = PointCartesien{5.0,5.0}
10,29
```

e) Écrire un programme de test qui effectue les traitements suivants :

•crée trois points en tirant leurs coordonnées au hasard dans le rectangle de coin inférieur gauche (-10.0,-10.0) et de coin supérieur droit (10.0,10.0). Vous utiliserez la méthode `random`() pour générer un nombre aléatoire dans l'intervalle [0,1], puis vous le modifierez de manière adéquate pour lui permettre de se trouver dans l'intervalle [-10,+10] ;
•affiche sous forme de chaîne de caractères (méthode `toString`) ces trois points ;
•et indique les deux points les plus proches.

Pour tirer un nombre au hasard on utilisera la méthode `random` de la classe `Math` du package `java.lang` . (*cf* https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html)

4. Écriture d'une nouvelle classe

Sur le modèle de la classe `PointCartesien`, écrire une nouvelle classe qui permet de représenter des cercles. Les opérations possibles sur un cercle sont :

- l'obtention de son rayon ;
- le changement de son rayon ;
- l'obtention de son centre ;
- la translation de son centre ;
- le calcul de sa surface ;
- le calcul de son périmètre ;
- la restitution d'une représentation textuelle du cercle (méthode `toString()`).

1) Définir les attributs caractérisant un cercle. Vous utiliserez deux attributs un point et un rayon.

2) Spécifier l'interface publique (signature des constructeurs et méthodes publiques de la classe).

3) Écrire le code java de la classe (on prendra soin pour chaque méthode de définir le commentaire documentant correspondant de façon à générer une documentation automatiquement avec l'outil javadoc).

Pour tester votre classe, dotez là d'un programme principal effectuant les opérations suivantes :

- création d'un cercle de rayon 1 et de centre 10,10 ;
- affichage de ses caractéristiques ;
- calcul et affichage de son périmètre et de sa surface ;
- doublement de son rayon ;
- à nouveau calcul et affichage de son périmètre et de sa surface ;
- déplacement du cercle pour ramener son centre à l'origine ;
- test de l'égalité du cercle avec un autre cercle de centre (0,0) et de rayon 2 ;
- test de l'égalité du cercle avec un autre cercle de centre (0,0) et de rayon1.