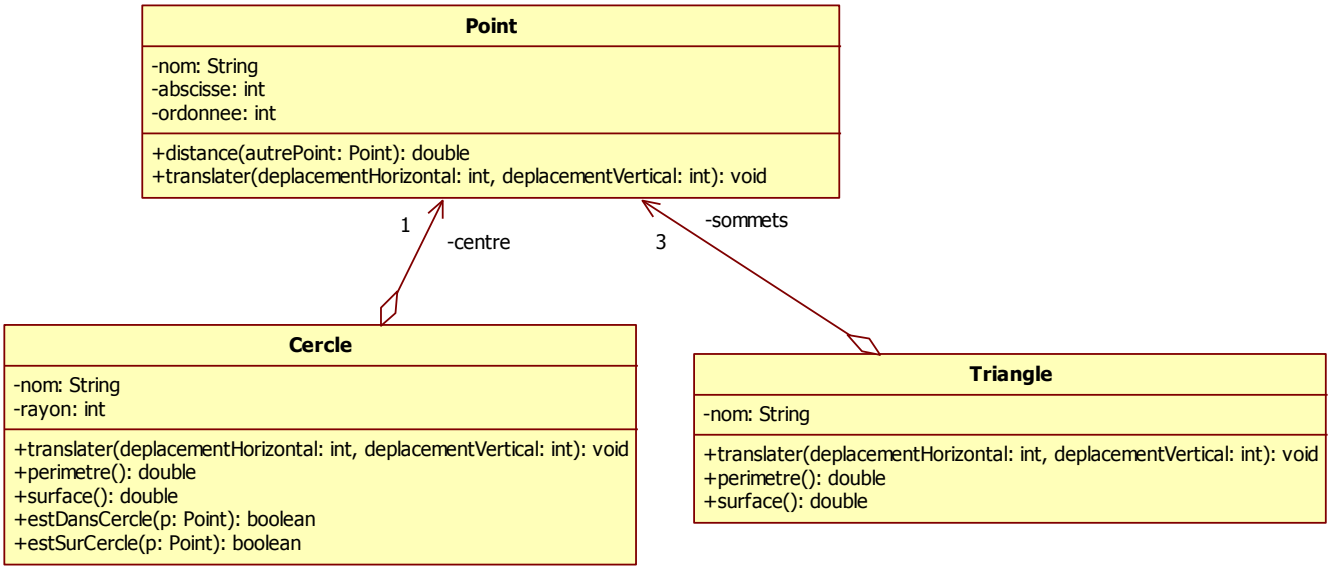
TD3 – Association de classes Java : Point, Cercle, Triangle, Polygone

Introduction.

Le but de ce TD est d'écrire de traduire d'UML vers Java la notion d'association.

On complète le diagramme UML du TD2 avec les classes Cercle et Triangle :



Exercice 1. Un Cercle - Mapping UML d'une agrégation simple.

a) Ecrire la classe **Cercle** qui permet de représenter un cercle avec son nom, son centre et son rayon en respectant le diagramme UML ci-dessus.
b) Fournir un **constructeur** complet prenant en paramètres le nom, le centre et le rayon du cercle.
c) Fournir des **accesseurs** en lecture pour les 3 propriétés.
d) Implémenter le code des 5 méthodes : **translater**, **perimetre**, **surface**, **estDansCercle** et **estSurCercle**.
e) Ajouter une méthode **toString()** pour représenter le cercle avec le format : **nom[centre ; rayon]**

Exercice 2. Une figure avec des points et des cercles.

a) Ajouter à l'application **AppliFigures** du TD2 un cercle C de centre B et de rayon 10. Afficher le cercle, son périmètre et sa surface.
b) Translater le cercle C d'un vecteur (-1,-1). Afficher le cercle C et le point B. Que constatez-vous ?
c) Translater le point B d'un vecteur (-1,-1). Afficher le cercle C et le point B. Que constatez-vous ?
d) Tester vos méthodes estDansCercle et estSurCercle avec quelques points significatifs. Que constatez-vous ?

Exercice 3. Un Triangle - Mapping UML d'une agrégation multiple.

a) Ecrire la classe **Triangle** qui permet de représenter un cercle avec son nom et ses 3 sommets. Traduire correctement la relation d'agrégation entre les classes Triangle et Point en suivant les règles du chapitre précédent. On tiendra compte du fait que le nombre de sommets est constant.
b) Fournir un **constructeur** complet prenant en paramètres le nom et les 3 sommets du triangle.

c) Fournir des **accesseurs** en lecture pour le nom et les sommets du triangle. Pour ces derniers, on aura la possibilité d'accéder à l'ensemble des sommets ou à un seul sommet à la fois par son numéro (1 à 3).

d) Implémenter le code des 3 méthodes : **translater**, **perimetre** et **surface**.
Pour le calcul de la surface d'un triangle, on pourra utiliser la formule de Héron :

Aire = $\sqrt{s(s-a)(s-b)(s-c)}$ avec s=½(a + b + c)

e) Ajouter une méthode **toString()** pour représenter le cercle avec le format :

nom/sommet1 - sommet2 - sommet3

Exercice 4. Une figure avec des points, des cercles et des triangles.

a) Ajouter à l'application **AppliFigures** un point D de coordonnée (10,8) puis un triangle T de sommets A, B et D. Afficher T, son périmètre et sa surface.

b) Translater le triangle T d'un vecteur (-1,-1). Afficher le triangle T.

c) Translater le point B d'un vecteur (-1,-1). Afficher le triangle T.

Exercice Bonus. La classe Polygone.

a) Définir la classe **Polygone** qui aggrège n points, ses sommets (dans la pratique, un vrai polygone est défini par au moins 3 points). Le constructeur prendra en paramètre la liste des sommets séparés par des virgules, quelque soit leur nombre. Par exemple :

```
1. Point pointA, pointB, pointC, pointD, pointE ; // sommets du pentagone ABCDE
2. pointA = new Point("A", 0, 0) ;
3. pointB = new Point("B", 5, 0) ;
4. pointC = new Point("C", 10, 10) ;
5. pointD = new Point("D", 5, 20) ;
6. pointE = new Point("E", 0, 10) ;
7. Polygone pentagone = new Polygone(pointA, pointB, pointC, pointD, pointE) ;
```

b) Ajouter à la classe un **accesseur en lecture** pour accéder au **nième sommet** du polygone. La méthode renverra une référence nulle si l'index est trop élevé.

c) Ajouter à la classe les mêmes méthodes que les autres figures : **translater, périmètre** et **toString**.

d) Ecrire une **application de test** permettant de manipuler des objets Polygone et leurs différentes méthodes.

e) Bonus : écrire la méthode **surface**