Suite TP Exceptions+Enregistrement

Enregistrement == article == données avec plusieurs champs...

Exercice 1 : On cherche a représenter des points dans un espace euclidien (et cartésien) a deux dimensions et d'avoir des méthodes « intéressantes » sur ces points. Chaque point comporte donc une abscisse et une ordonnée (coordonnées) qui seront chacun un « float ». Voici les spécifications :

a) Écrivez une classe **Point** qui contiendra un constructeur qui initialise les abscisses et ordonnées à 0.

b) Toutes les méthodes doivent être documentées (javadoc)

c) On souhaite les méthodes (de base) suivantes :

1) **public void set(float abs, float ord)** qui met de modifier les coordonnées d'un point

2) **public void translate(float u, float v)** qui permet de déplacer un point (abs+u,ord+v)

3) **public void toString()** qui permet l'affichage (à votre guise)

4) **public boolean origine()** qui test si les coordonnées d'un point sont nulles

5) **public boolean equals(Point pt)** qui test si le point est égale a un autre point (les coordonnées sont identiques)

6) **public float distance(Point pt)** qui calcul la distance du point avec un autre point

d) On souhaite rajouter une méthode permettant de calculer la distance d'un point par rapport à une droite (https://fr.wikipedia.org/wiki/Distance_d%27un_point_%C3%A0_une_droite pour avoir la formule); Pour ce faire on suppose que la droite est représentée par le triplets de floats a,b,c (pour ax+by+c=0, x,y sont les coordonnées de tout les points d'une droite) ; Il nous faut donc la méthode

public float distance(float a, float b, float c)

Testez pour a et b qui valent respectivement 0. Créer une exception appelée **DistanceImpossible** pour gérer ce problème...

e) Écrivez un petit **main** pour tester vos points ; Rattraper l'exception si besoin

ps : Il vous faudra utiliser les méthodes statiques Math.sqrt(float a) et Math.abs(float a)

Exercice 2 : On veut maintenant représenter des polygones (https://fr.wikipedia.org/wiki/Polygone), des figures géométriques avec N sommets. Voici les spécifications :

a) Les sommets (points) seront un tableau de points. Le constructeur définira juste la taille du tableau et on supposera des points avec des coordonnées à 0 partout.

b) Toutes les méthodes doivent être documentées (javadoc)

c) On souhaite le méthodes de base suivantes :

1) **public void set(Point pt, int n)** initialise le **n**ième sommet avec le point **pt**

2) **public void toString()** qui permet l'affichage (à votre guise)

3) **public boolean equals(Poligone pg)** qui test si le polygone est égale a un autre polygone (égalité de la suite de points, dans un sens ou dans l'autre !!!) ; Attention si le nombre de sommets différent !

4) **public float perimetre()** calcul le périmètre du polygone. On peut s'aider de la formule suivante sur la distance entre 2 points :

https://fr.wikipedia.org/wiki/Distance_entre_deux_points_sur_le_plan_cart%C3%A9sien

d) Gérer le cas où le tableau est initialisé avec un N<=0

e) Écrivez un petit main pour tester vos polygones ; Rattraper l'exception pour ne lire que des polygones correctes

Pour les courageux, on peut écrire des méthodes permettant de déterminer si un polygone est (ou non) croisé, convexe *etc.*