Programmation Orientée Objet (Java)
*****

TD 4 JAVA : Tableaux d'Objets et Héritage

Exercice 1: Forme géométrique et groupe de forme

1-Définissez une classe Polygone représentant un polygone à n côtés dans le plan comportant les variables membres:

- **sommets –** tableau contenant les sommets(Point2D) du polygone
- déclaration d'une **variable statique comptpoly initialisée à 0, qui s'incrémentera à chaque construction d'un nouveau polygone.**
- **public Polygone(Point2D[] sommets) – unique constructeur** d'un polygone à partir du tableau de ses sommets. Les points peuvent exister sans le Polygone (agrégation), donc ils ne sont pas copiés à la construction du Polygone mais partagés avec d'autres classes.
- **public Point2D getSommet(int i) –** obtention du ième sommet du polygone
- **static double longueur(Point2D p, Point2D q)** - Méthode de classe longueur d'un côté (déterminé par 2 points)
- **double perimetre ( ) -** Méthode d'instance calculant le périmètre du polygone
- **public String nom()** - La fonction nom rendra le type du polygone dont il s'agit ("polygone" dans le cas présent)
- **public String toString()** – expression textuelle du polygone ; par exemple "[(1.0,1.0),(5.0,1.0),(3.0,2.0)]"

2-Définissez une classe Triangle, sous-classe de Polygone, avec un constructeur

- **public Triangle(Point2D a, Point2D b, Point2D c)** – construction du triangle ayant les trois sommets indiqués
- **boolean estTriangleRectangle() –** vérifie si le triangle est rectangle**.**

3-De la même façon, écrire une classe Quadrilatere qui hérite de la classe Polygone avec un constructeur

- **public Quadrilatere(Point2D a, Point2D b, Point2D c, Point2D d)**
- **boolean estParallelogramme** – vérifie si le quadrilatère est un parallélogramme.

- Pour ces deux classes redéfinir la méthode **nom(),** la fonction rendra le type du polygone dont il s'agit ("Triangle" pour la classe Triangle ou "Quadrilatère" pour la classe Quanrilatere)

.

Redéfinir, dans toutes les classes que vous venez de créer tout au long de ce TD, la méthode equals(Object that) - qui permet de comparer les objets - afin de fournir une égalité basée sur le contenu au lieu d'une égalité identitaire .Vous devez faire bien attention aux points suivants :
— Le cas où l'objet that passé en paramètre est le même que l'objet appelant.
— Le cas où l'objet that est égal à null.
— Dans les cas autres que les deux cas susmentionnés, vous devez vous assurer que l'objet that est une instance de la même classe que votre objet appelant (vous pouvez utiliser l'opérateur instanceof), dans ce cas, il faut "caster" that dans cette classe pour pouvoir accéder à ses attributs convenablement. La signature de la méthode equals est la suivante : public boolean equals(Object that). Vous devez impérativement adhérer à cette signature. Notamment, l'objet passé en paramètre doit obligatoirement avoir Object comme type.

4-Écrivez une classe TestPolygone avec une méthode main construisant et affichant des polygones de diverses sortes et permettant de vérifier les différentes méthodes.

public static void main(String args[]) {
// construction d'un premier point a à l'origine,
// construction d'un deuxième point b d'abscisse 0 et d'ordonnée 3
//construction d'un troisième point c d'abscisse 4 et d'ordonnée 0

//construction d'un triangle avec les 3 points construits précédemment
// affichage du nom du triangle, du triangle et de son périmètre du triangle
//vérifier et afficher si le triangle est rectangle
// construction d'un quatrième point d d'abscisse 4 et d'ordonnée 3
//construction d'un quadrilatère avec les 4 points a, b, c et d.
//affichage du nom du quadrilatère de ses caractéristiques et de son périmètre
//vérifier et afficher si le quadrilatère est un parallélogramme
//affichage du nombre de polygones construits } }
Si vous définissez une référence Polygone poly que vous initialisez à un triangle, que va afficher poly.nom() ?la méthode estRectangle() est-elle définie pour l'objet poly ?

5-Ecrire une classe GroupeP composée de tableau privé de polygone hétérogènes (Triangle et Quadrilatère) et un compteur indiquant le nombre d'éléments exacts dans le tableau.

- Ecrire un constructeur avec un paramètre x qui permet de créer le tableau de taille x Polygones maximum.
- Ecrire une méthode ajouterPolygone(Polygone p) qui permet d'ajouter un polygone au tableau. Cette méthode retourne vrai si le polygone p est ajouté faux sinon.
- Ecrire une redéfinition de la méthode toString() qui permet d'avoir une représentation textuelle de tous les polygones présents dans la tableau.
- Ecrire une méthode perimetre() qui permet de calculer le périmètre de tous les polygones du groupe.

Exercice 2: Classe Personne

Ecrivez une classe Personne avec les attributs d'instances privés suivants :
nom: Le nom de famille de la personne, String
prenom: Le prénom de la personne, String
age: L'âge de la personne, int (**L'âge est compris entre 0 et 130 ans)**
genre: **Masculin** ou **Féminin**. String
La classe Personne doit disposer des constructeurs suivants :
Personne(): constructeur par défaut, initialisant le nom et prénom et le genre à une ***chaine vide***.
Personne (nom, prenom, age, genre), le constructeur complet
Personne (Personne), le constructeur complet.
Remarque : ***Respectez les consignes déjà mentionnées dans le cours pour l'écriture des constructeurs ; i.e. écrire le constructeur complet avec des setters et tous les autres constructeurs feront appel à ce dernier.***
La classe Personne doit contenir des accesseurs et mutateurs pour les différents attributs.
— Une méthode sameLastName (Personne p) qui prend en paramètre un deuxième objet de type Personne et qui permet de savoir si les deux personnes ont le même nom de famille **en affichant un message**.
— Une méthode oldest (Personne p) qui compare la personne appelante avec la personne fournie en paramètre **et retourne la personne** qui est la plus âgée.
—Elle doit aussi contenir une méthode **toString()** donnant une représentation de la classe Personne comme suit :
La personne s'appelle ... .
Il est [un homme | une femme]
Il a.... ans
—Redéfinir la méthode equals
Ecrivez aussi une classe testPersonne afin de tester la classe Personne effectuant les opérations suivantes :

- Création d'une première personne p1 par le constructeur par défaut
- Modifier le nom, le prénom, l'âge de la personne p1 avec les votre.
- Afficher les informations de la personne p1.
- Création d'une seconde personne p2 par recopie de p1
- Modifier le nom et l'âge de p2

- Afficher les informations de l'étudiant p2
- Création d'une troisième personne p3 avec les trois arguments : "Dupond", "Jean", 131
- Afficher juste le nom et l'âge de la personne p3

Exercice 3 : classe Employe

---

un employé est une personne particulière qui possède des attributs et méthodes particuliers. Définir une classe
Employe en la faisant hériter de **Personne** avec les champs particuliers suivants :
constante *SOCIETE* de valeur "MICRORDI"
attributs d'instance privés : *salaire* de type **float,** son numéro de type **int**
attribut statique privé : *nombre* pour noter le nombre total d'employés "créés"

constructeurs : Ecrire les constructeurs en faisant appel aux constructeurs correspondants de Personne. Les adapter en leur ajoutant l'argument Salaire et numero

```
public Employe()
public Employe(float salaire, int numero) et
public Employe(Personne personne, float salaire, int numero)
```

méthodes :getSalaire(), getNumero(), getNombre(), setSalaire()et Redéfinir la méthode toString() : en utilisant la méthode toString() de la classe parent et faire afficher le salaire et le numéro:

```
la personne s'appelle ….
 il est un/une …..
 il a ….. ans
 Employe [SOCIETE=MICRORDI, salaire=…., numero=….]
```

La classe TestEmploye
Créer des employés en utilisant des constructeurs variés et les faire afficher

Exercice 4: classe Entreprise

Proposer une classe Entreprise dont chaque instance contient une liste d'employés. Un constructeur qui initialise la liste avec un nombre max d'élément. On programmera les méthodes suivantes :
(a) ajout d'un employé ; attention ce tableau d'objet ne peut pas contenir des trous (valeurs null)
(b) suppression d'un employé désigné par son numéro ; faites un décalage pour éviter des trous.
(c) calcul du nombre des employés ayant un nom donné ;
(d) calcul du salaire moyen des employés.

Avancé : Pour ceux qui veulent avancer plus, essayez de modifier cette classe en définissant la liste d'employés avec la classe ArrayList voir : https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
Un objet ArrayList est un objet qui n'a pas de taille limite et qui, en plus, accepte n'importe quel type de données, y compris null ! Nous pouvons mettre tout ce que nous voulons dans un ArrayList et sont rapides en lecture.
Sachez aussi qu'il existe tout un panel de méthodes fournies avec cet objet :

- add() permet d'ajouter un élément ;
- get(int index) retourne l'élément à l'indice demandé ;
- remove(int index) efface l'entrée à l'indice demandé ;
- isEmpty() renvoie « vrai » si l'objet est vide ;
- removeAll() efface tout le contenu de l'objet ;
- contains(Object element) retourne « vrai » si l'élément passé en paramètre est dans l'ArrayList.