

	<b>TP N°7</b> <b>Dérivations successives + classe abstraite+ interface</b>	
---	---	---

Le but de ce problème est de traiter le calcul des impôts pour chaque type de propriété personnelle ou professionnelle.

Une propriété personnelle peut être soit une villa ou un appartement.

### I.La classe Personne

Définissez une classe *Personne* caractérisée par les attributs et les méthodes suivantes :

- *cin* : (int)
- *nom* : (String)
- *prenom* : (String)

- 1) Définissez un **constructeur** prenant en paramètre les trois attributs.
- 2) Définissez les **getters** et les **setters** des attributs.
- 3) Définissez une méthode publique ***toString()*** qui retourne l'état de l'instance, c'est-à-dire la valeur de ses attributs.

### II.La classe Propriété

Définissez une classe *Propriété* qui a pour attributs des informations valables pour tout type d'habitation :

- *Id* : entier
- *responsable* : (Personne)
- *adresse* : (String)
- *surface en m<sup>2</sup>* : (double)

- 4) Définissez un **constructeur** permettant l'initialisation explicite de l'ensemble des attributs.
- 5) Définissez une méthode publique ***toString()*** qui retourne l'état de l'instance, c'est-à-dire la valeur de ses attributs.
- 6) Les impôts à payer dépendent du type de la propriété (individuelle ou professionnelle) : Ajoutez donc une méthode ***double calculImpot()*** dans la classe *Propriété*, cette **méthode n'a pas d'implémentation** (elle sera implémentée dans les classes filles).

### III.Les classes dérivées

Définissez deux classes *PropriétéPrivée* et *PropriétéProfessionnelle*, héritant de la classe *Propriété* et ayant les attributs supplémentaires suivants :

- pour la classe *PropriétéPrivée*:
  - *nbPièces* : (int) ;
- pour la classe *PropriétéProfessionnelle*:
  - *nbEmployés* : (int);
  - *estEtatique* : (boolean) (vaut true s'il s'agit d'une propriété appartenant à l'État, false sinon)

Remarque : le responsable d'une propriété privée est le propriétaire.

Définissez deux classes **Villa** et **Appartement**, héritant de la classe *PropriétéPrivée* et ayant les attributs supplémentaires suivants :

- pour la classe *Villa*:
  - *avecPiscine* : (boolean).
- pour la classe *Appartement*:
  - *numEtage* : (int).

- 7) Définissez, pour chacune de ces classes, un **constructeur** permettant l'initialisation explicite de l'ensemble des attributs.
- 8) Redéfinir la méthode ***toString()*** pour les deux classes afin qu'elle retourne la valeur de tous les attributs.

Les constructeurs et méthodes ***toString()*** devront utiliser les méthodes appropriées de la classe parente.

- 9) Redéfinissez la méthode ***double calculeImpot()*** dans les sous-classes de sorte à calculer et retourner la valeur des impôts à payer en fonction de certains critères :
  - Pour une Propriété privée, les impôts sont calculés de la manière suivante :
    - $50DT/100 m^2 + 10DT/pièce$
  - Pour une villa :
    - Si l'habitation comporte une piscine, le propriétaire devra ajouter 200 dinars aux impôts d'une propriété privée.
  - Pour une Propriété professionnelle, les impôts reviennent à :
    - Si la propriété n'appartient pas à l'État : la méthode retourne  $100 DT/100 m^2 + 30 DT/employé$ .
    - Sinon (appartenant à l'État) alors la méthode retourne 0.

#### IV.Interface

Ecrire une interface **GestionPropriete** ayant les éléments suivants :

- **MAX\_PROPRIETES** : une constante présentant le nombre max de propriétés autorisés

- **void afficherPropriétés()** : méthode qui affiche les détails d'une propriété
- **boolean ajouter(Propriété p)** : méthode qui ajoute une nouvelle propriété et renvoie **true** (**false** en cas d'échec)
- **boolean supprimer(Propriété p)** : méthode qui supprime la propriété indiquée et renvoie **true**(**false** en cas d'échec)

## V.Classe Lotissement

Un lotissement sera représenté par un tableau de *Propriétés* :

```
protected Propriété [] tabProp;
protected int nombre;
```

Cette classe implémente l'interface GestionPropriete Et possédant les méthodes suivantes :

- **Lotissement(int capacité)** - constructeur qui crée un lotissement ayant la capacité (nombre maximum de propriétés) indiquée
- **Propriété getpropriétéByIndex(int i)** - renvoie la ième propriété d'indice i
- **int getnbPièces()** - retourne le nombre de pièces de toutes les propriétés privées.

**NB** : deux propriétés sont égales s'ils ont le même id

## VI.Classe Fiscalité

Dans une classe *Fiscalité* contenant la méthode **main()**, effectuer les opérations suivantes :

- 10) Créer 3 personnes (propriétaires) **p1, p2, p3**
- 11) Créer un lotissement de 10 propriétés.
- 12) Ajouter les propriétés suivantes au lotissement :

```
ProprietePrivee(1,p1,"Corniche",350,4);
```

```
Villa(2,p2,"Dar Chaabane", 400,6,true);
```

```
Appartement(3,p2,"Hammamet",1200,8, 3);
```

```
ProprieteProfessionnelle(4,p3,"Korba", 1000, 50,true);
```

```
ProprieteProfessionnelle(5,p1,"Bir Bouragba",2500, 400, false);
```

Afficher **toutes les informations** concernant les propriétés du lotissement **y compris la valeur des impôts à payer**.

- 13) Afficher le nombre global des pièces
- 14) Afficher, pour **la propriété privée** qui paye le **moins** d'impôts, le propriétaire ainsi que le montant des impôts qu'il va payer.
- 15) Supprimer l'appartement du Hammamet puis réafficher la liste des propriétés

## VII.Classe LotissementPrivée

Soit à définir une classe LotissementPrivée dont les instances ont les mêmes fonctionnalités que les Lotissements mais sont entièrement constituées de propriétés Privées.

Cette classe hérite des attributs et des méthodes de la classe Lotissement.

Elle possède un constructeur qui prend comme argument la capacité et redéfinit les méthodes suivantes :

- public boolean **ajouter**(Propriété p)
- public ProprietePrivee **getpropriétéByIndex** (int i)
- public int **getnbPièces()**

16) Dans la classe Fiscalité, créer un objet LotissementPrivée comme suit :

```
Lotissement lt= new LotissementPrivee(10);
```

Puis tester les différentes méthodes définies précédemment.