USTHB **TP POO**
Faculté d'Informatique **ING 2 (A)**

TP 0 : Rappels sur l'OO

Exercice 01 :

Définir la classe CompteBancaire comme suit :

1) **Variables d'instance:** Les comptes bancaires CompteBancaire sont représentés le plus simplement possible, i.e., par deux variables d'instance de noms respectifs solde et id :
   • solde : un nombre en double précision conservant le solde du compte.
   • id : une chaîne de caractère mémorisant l'identité du propriétaire du compte.
   Constructeurs : On dotera cette classe des 2 constructeurs.
   • CompteBancaire(double soldeInitial, String idInitial) constructeur avec valeurs initiales explicites pour le solde et le nom id du propriétaire du compte.
   • CompteBancaire() constructeur sans arguments, donnant la valeur 0 au solde. et la valeur "anonyme" au nom du propriétaire.
2) **Accesseurs et modifieurs :**
   •String getId() accesseur en lecture retournant une chaîne de caractère désignant le propriétaire du compte.
   • void setId(String id) accesseur en écriture permettant de changer le propriétaire du compte, du moins la chaîne de caractère le représentant.
   • double getSolde() accesseur en lecture retournant le montant du solde.
   • void setSolde(double solde) accesseur en écriture permettant de changer le montant du solde.
3) **Méthodes d'instance :**
   • void retirer(double montant) opération débitant le compte d'un montant strictement positif si le solde du compte est au moins égal à montant (ne fait rien sinon).
   • void deposer(double montant) opération créditant le compte d'un montant strictement positif (ne fait rien si montant n'est pas strictement positif).
   • void transferer(double montant, CompteBancaire compteDepot) opération transférant le montant strictement positif du compte courant au compte

compteDepot si le solde du compte courant est au moins égal à montant (ne fait rien sinon).

• void afficher() affiche à l'écran la chaîne de caractère caractérisant l'état du CompteBancaire.

4) Utilisation de la classe CompteBancaire :

Déclarer une référence compte sur un CompteBancaire; Créer/construire le compte bancaire correspondant ; Affecter Mohamed Kamel comme propriétaire du compte, déposer 100 unités, puis encore 50 unités, en retirer 20 pour acheter des livres, calculer le solde ; En supposant qu'il existe un autre compte, référencé par la variable compteEpargne, transférer 100 unités du compte au compte compteEpargne.

Exercice 02 :

On considère une entité Adresse décrite par un numéro, un nom de rue, un code postale et une ville.

1) Donnez une implémentation de l'entité Adresse, en donnant la possibilité de retourner ou de modifier n'importe quel champ de l'entité par des méthodes définies dans la classe, prévoir une méthode d'affichage.
2) Créer une entité Citoyen décrite par un nom, prénom et adresse. Pour les méthodes, on demande de définir uniquement le constructeur et une méthode d'affichage.
3) Ecrire un programme qui crée un tableau C de 10 citoyens et affiche ensuite tous les citoyens (nom, prénom, adresse) habitant dans une même ville V donnée.

Exercice 03 :

Soit les classes suivantes :

La classe Personne qui comporte deux attributs privés, nom, prénom et anneeNaissance. Cette classe comporte un constructeur pour permettre d'initialiser les attributs. Elle comporte également une méthode polymorphe afficher() pour afficher les données de chaque personne.

La classe Employé qui hérite de la classe Personne, avec en plus un attribut privé salaire, un constructeur et la redéfinition de la méthode afficher() qui permet d'afficher le salaire.

La classe Directeur qui hérite de la classe Employe, avec en plus un attribut (instance d'objet) société, un constructeur et la redéfinition de la méthode afficher() qui permet d'afficher le nom de la société.

La classe Societe qui comporte le nom de la société nomSociete et le nombre d'employés nbEmploye.

Questions :

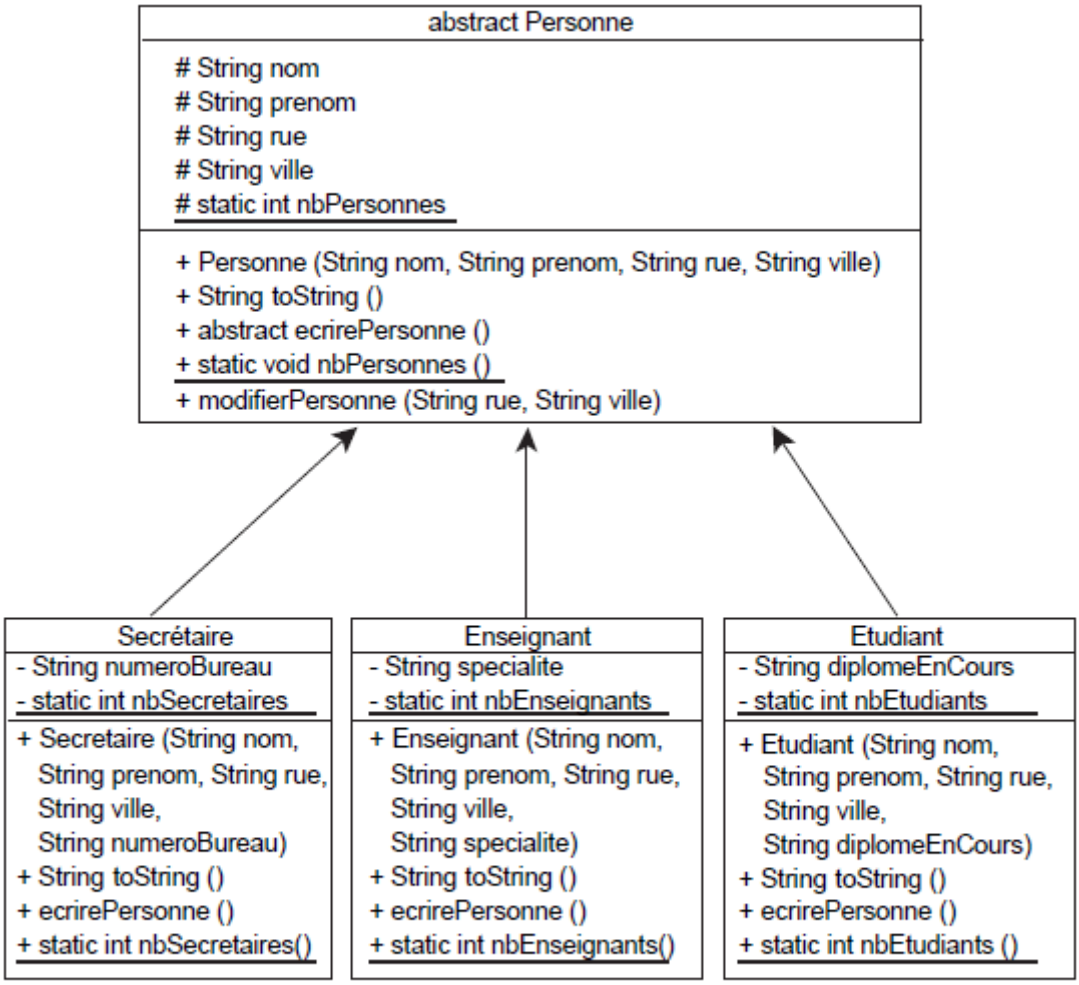
1. Ecrire les classes Personne, Employe, Directeur et Societe. On vous demande d'écrire les accesseurs uniquement pour la classe Société.

2. Ecrire un programme java qui crée un tableau de 5 personnes comportant 4 employés, et un directeur (5 références de la classe Personne dans lesquelles ranger 4 instances de la classe Employé, et 1 de la classe Directeur).

3. Affichez l'ensemble des éléments du tableau à l'aide de l'instruction for.

Exercice 04 :

Dans un établissement d'enseignement, on trouve trois sortes de personnes : des administratifs représentés par la catégorie secrétaire, des enseignants et des étudiants. Chaque personne est caractérisée par son nom et prénom, son adresse (rue et ville) qui sont des attributs protégés et communs à toutes les personnes. On peut représenter une personne suivant un schéma UML comme indiqué sur la figure.



Arbre d'héritage.

Les classes Secretaire, Enseignant et Etudiant sont des classes dérivées de la classe Personne. La classe Personne est abstraite et ne peut être instanciée.

indique un attribut protected, - indique un attribut private, + indique un attribut public.

Les variables d'instances (attributs) sont le nom, le prénom, la rue et la ville. *nbPersonnes* est une variable de classe (donc static) qui comptabilise le nombre de Personne dans l'établissement.

On définit les méthodes public suivantes de la classe **Personne** :

• le constructeur **Personne** (String nom, String prenom, String rue, String ville): crée et initialise un objet de type Personne.

• String **toString** () : fournit une chaîne de caractères correspondant aux caractéristiques (attributs) d'une personne.

• **ecrirePersonne** () : pourrait écrire les caractéristiques d'une personne. Sur l'exemple ci-dessous, elle ne fait rien. Elle est déclarée **abstraite**.

• static **nbPersonnes** () : écrit le nombre total de personnes et le nombre de personnes par catégorie. C'est une méthode de classe.

• **modifierPersonne** (String rue, String ville) : modifie l'adresse d'une personne et appelle **ecrirePersonne** () pour vérifier que la modification a bien été faite.

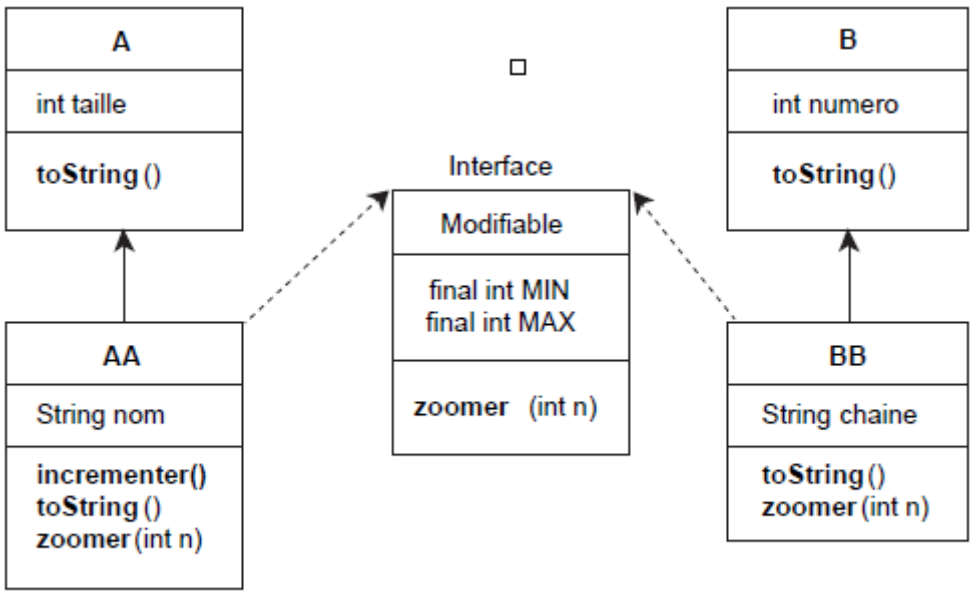
Questions :

1) Implémentez les classes ci-dessus.
2) Compléter la classe exécutable PPPersonne1 suivante :

```
public class PPPersonne1 { // classe exécutable
public static void main(String[] args) {
Secretaire Sec=new Secretaire("Seba","Zohra","rue des rosiers", "Alger", "A326");
Enseignant Ens=new Enseignant ("Fridi","Boualem","rue des mimosas", "Alger",
"Mathematique");
Etudiant Etu=new Etudiant ("Smati","Ryad","rue des lilas", "Alger", "Informatique") ;
// à compléter … affichez le nombre de personnes.
System.out.println("\nAprès modification:\n");
/* à compléter … modifiez l'adresse de « Seba Zohra » par : "rue des orangers, ville de Blida.
*/
/* à compléter … modifiez l'adresse de « Fridi Boualem » par : "rue des
marguerites", "Kouba". */
// à compléter … affichez les informations de toutes les personnes.
}
```

Exercice 05 :

On définit, une classe AA qui hérite de la classe A et une classe BB qui hérite de la classe B. L'interface Modifiable permet de voir les classes AA et BB comme ayant une propriété commune, celle d'être modifiable. Une interface définit uniquement des constantes et des prototypes (signatures) de méthodes.



Héritage et interface.

Questions :

1) Implémentez les classes et l'interface ci-dessus.
2) Compléter la classe exécutable PPInterface suivante :

```
public class PPInterface { // classe exécutable

public static void main (String[] args) {

// à compléter ...  créer un tableau de Modifiable contenant de 2 objets AA et 2 objets BB.

// à compléter ...  afficher le tableau.


// à compléter ...  déclencher la méthode zoomer() pour les éléments du tableau.
// les AA sont divisés par 2.
// les BB sont divisés par 4.

// à compléter ...  afficher le tableau.

}
}
```