TD 5 JAVA : Héritage et Abstraction
Tris

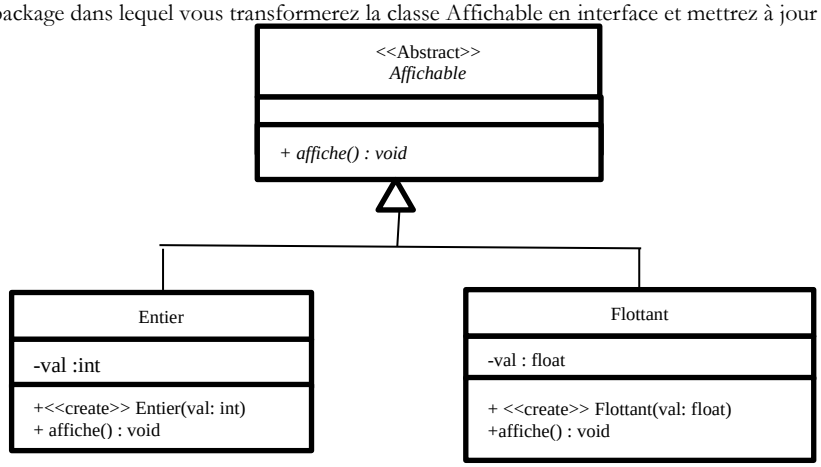
Remarque : une classe abstraite et une méthode abstraite sont représentées en italique dans un diagramme de classe.

Exercice 1 :

Partie1 « Affichable »

Le but de cette exercice est de créer une classe abstraite Affichable, dotée d'une seule méthode abstraite void affiche(). Deux classes, Entier et Flottant, dérivent de cette classe. Chacune de ces deux classes possèdent un attribut entier et flottant respectivement et un constructeur complet qui initialise son attribut.
Vous différencierez les méthodes affiche des classes Entier et Flottant en présentant ces objets de la manière suivante :
Je suis un entier de valeur 28
Je suis un flottant de valeur 3.27

1. Réaliser les classes Affichable avec une seule méthode, Entier et Flottant, et TestAffiche (avec la méthode main qui utilise un tableau hétérogène d'objets de type Affichable qu'elle remplit en instanciant des objets de type Entier et Flottant). Ces classes permettent de mettre en œuvre la notion de classe abstraite et le polymorphisme.
2. Créer une nouveau package dans lequel vous transformerez la classe Affichable en interface et mettrez à jour les classes Entier et Flottant.



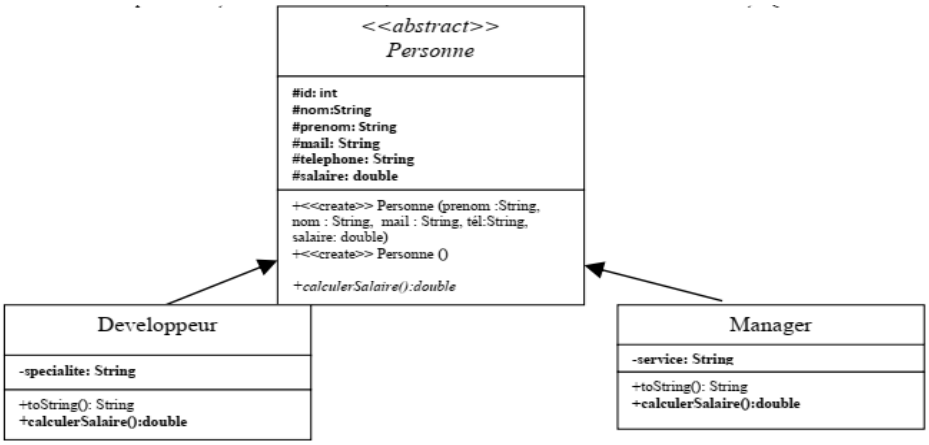
Partie2 « Société Clavier »

Le directeur des systèmes d'information de la société CLAVIER souhaite développer un module pour la gestion des utilisateurs de son service, pour cela il vous a fait appel pour réaliser cette tâche. Le diagramme de classe a été établi par un analyste afin de mettre en place une base de données sous ORACLE ou MySQL :

1. Créer la classe **abstraite «Personne»** avec un constructeur complet et un constructeur par défaut, où l'id est un compteur automatique.
2. Créer les classes **«Developpeur» et «Manager»** avec chacun un constructeur complet en redéfinissant la méthode calculerSalaire(), sachant que:

- Le développeur aura une augmentation de 20% par rapport à son salaire normal.
- Le manager aura une augmentation de 35% par rapport à son salaire normal.

3. Dans une autre classe test, créer deux développeurs et deux managers.
4. Afficher les informations des objets crées sous la forme :
Le salaire du manager …….. est: 7000 dt, son service: Informatique
Le salaire du développeur …… est: 4000 dt, sa spécialité : PHP



5. Modifiez les classes Developpeur et Manger pour qu'elles implémentent aussi l'interface Affichable et modifiez la classe TestAffiche et la méthode main qui utilise un tableau hétérogène d'objets de type Affichable qu'elle remplit en instanciant des objets de type Entier et Flottant mais aussi des objets Manager et Developpeur.

Exercice 2 :

Le tri à bulles est un algorithme classique permettant de trier un tableau d'entiers. Il peut s'écrire de la façon suivante en Java :

```
Static void triBulles(int tab[])
{       Boolean change = false;
        do {    change = false;
                for (inti=0; i<tab.length - 1; i++) {
                        if (tab[i] >tab [i+1]) {
                        inttmp = tab[i+1];
                        tab[i+1] = tab[i];
                        tab[i] = tmp;
                        change = true;   }}
          } while (change);
}
```

Cette implémentation du tri à bulles permet de trier un tableau d'entiers. Maintenant on veut pouvoir utiliser tout autre type de données (muni d'une relation d'ordre) sans avoir à réécrire l'algorithme à chaque fois. Pour cela on va supposer définie comme suit, l'interface Triable :

```
public interface Triable {
// échange les éléments en positions i et j
void echange(int i, int j);
// retourne vrai si l'élément de position i est plus grand que l'élément de position j
boolean plusGrand(int i, int j);
// nombre d'éléments à trier
int taille();}
}
```

Les objets des classes implémentant cette interface devront représenter ainsi des tableaux d'éléments, comparables entre eux, que l'on souhaite trier.

1- Ecrivez dans une classe Test la méthode static void triBulles(Triable t) qui met en œuvre le tri a bulles pour les objets Triables. Elle est similaire à static void triBulles(int tab[]), à quelques différences prés.

2- Ecrivez une classe EntierTriable qui implémente l'interface Triable et permet à triBulles(Triable t) de trier des entiers (selon leur ordre naturel). Ecrire un constructeur qui prend en paramètre un tableau d'entier plein. Ecrire la redéfinition de la méthode toString() et de la méthode equals (qui compare deux EntierTriable et retourne true quand les contenus sont les même, i.e. ils ont la même taille et des cases aux mêmes contenus).

3- Ecrivez une classe Dictionnaire qui implémente l'interface Triable et permet à triBulles(Triable t) de trier des chaînes de caractères (en ordre alphabétique). Ecrire un constructeur qui prend en paramètre un tableau de String plein. Ecrire la redéfinition de la méthode toString() et de la méthode equals (qui compare deux Dictionnaires et retourne true quand les contenus sont les même, i.e. ils ont la même taille et des cases aux mêmes contenus).

4- Tester votre code avec cette méthode main() :

```
Public static void main(String[] args) {
                    EntierTriable et= new EntierTriable(newint[] {56,3,4,23,1,9,99});
                    EntierTriable et1= new EntierTriable(newint[] {3,4,56,23,9,1,99});
                    System.out.println(et);
                    System.out.println(et1);
                    //icinousdevrionsavoirun false car lescontenusdes 2
tableauxsontdifférents(ilsontlesmêmesélémentsmaisordonnésdifférement)
                    System.out.println("les deux tableaux sont egaux: "+ et.equals(et1));
                    triBulles(et);
                    triBulles(et1);
                    System.out.println("Après tri");
                    System.out.println(et);
                    System.out.println(et1);
                    //icinousdevrionsavoirun true car lescontenusdes 2 tableauxsontlesmêmesaprèstri!
                    System.out.println("les deux tableaux sont egaux: "+ et.equals(et1));
                    Dictionnaire dt= new Dictionnaire(new String[] {"allo","salut","byebye"});
                    Dictionnaire dt1= new Dictionnaire(new String[] {"coucou","allo","salut","byebye"});
                    System.out.println(dt);
                    System.out.println(dt1);
//icinousdevrionsavoirun false car lescontenusdes 2 tableauxsontdifférents(ils n'ont pas les mêmes éléments)
                    System.out.println("les deux tableaux sont egaux: "+ dt.equals(dt1));
                    triBulles(dt);
                    triBulles(dt1);
                    System.out.println("Après tri");
                    System.out.println(dt);
                    System.out.println(dt1);
                    //icinousdevrionsavoirun false car lescontenusdes 2 tableauxsontdifférentsmêmetriés
                    System.out.println("les deux tableaux sont egaux: "+ dt.equals(dt1));
                    }
```

5- Sur le même modèle, inventez et écrivez une autre classe qui implémente Triable