ITC315 - Informatique 2

TP 2 - JAVA : Héritage et polymorphisme

Wahabou Abdou

wahabou.abdou@u-bourgogne.fr

2022 - 2023

La notion d'héritage permet de réutiliser des classes. Une classe dérivée (fille) hérite des attributs et méthodes d'une classe de base (mère).

Ce TP a pour objectif d'étudier les droits d'accès aux attributs et méthodes d'une classe après un héritage, la construction et l'initialisation d'objets, la redéfinition de méthodes et la notion de polymorphisme. Rappelons que le polymorphisme permet de manipuler des objets sans en connaître précisément le type. Par exemple, dans notre cas, en nous appuyant sur l'illustration de la figure 1, nous pourrons manipuler un tableau contenant des objets de type `Personnel`. Certains éléments de ce tableau pourront être de type `Administratif` et d'autres de type `Enseignant`. Nous pourrons lire et afficher les éléments de ce tableau sans en connaître le type précis.

Au cours de ce TP, nous utiliserons la représentation hiérarchique des classes présentée à la figure 1.

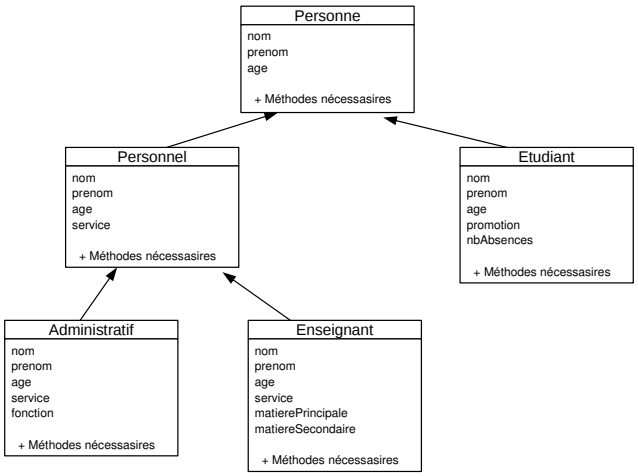

Figure 1 – Représentation hiérarchique des classes

Exercice 1 : Droits d'accès

Soit la classe *Personne* définie comme suit :

```
public class Personne {

        private String nom;
        public String prenom;
        protected int age;

        public Personne(){
                nom = "Anonyme";
                prenom = "Anonyme";
                age = -1;
        }

        public Personne (String nom, String prenom, int age){
                this.nom = nom;
                this.prenom = prenom;
                this.age = age;
        }
}
```

1. Dans la classe *Personnel* les attributs *nom, prenom, age* doivent-ils être explicitement définis ?
2. Écrire les classes *Personne* et *Personnel*.
3. Dans la définition des classes filles, à quoi sert le mot-clé *super* ?
4. Dans la classe Personnel, pourrait-on afficher l'*age* si l'on dispose de la méthode *afficherAge()* définie ci-après ?

```
public void afficherAge() {
        System.out.println( "Ce personnel a " + age + " an(s)" );
}
```

> ***Note :***
> *Si vous avez bien recopié le code proposé au début de l'exercice et bien implémenté la classe* `Personnel`*, l'affichage de l'âge devrait être possible. Si tel n'est pas le cas chez vous, faites appel à l'enseignant.*

5. Serait-il possible d'écrire, de la même façon une méthode *afficherNom()* ? Et pour *afficherPrenom()* ? Que peut-on déduire au sujet des droits d'accès (`private, public et protected`) aux membres et méthodes de la classe de base par les classes dérivées (classes filles) ?

> ***Indications :***
> *Le tableau 1 indique la visibilité des attributs et méthodes en fonction des droits choisis.*

	Dans la classe	Dans le package	Dans les classes dérivées	Ailleurs
`private`	Oui	Non	Non	Non
`public`	Oui	Oui	Oui	Oui
`protected`	Oui	Oui	Oui	Non
Si rien n'est indiqué	Oui	Oui	Non	Non

Tableau 1 – Règle de visibilité des attributs et méthodes

6. Une classe fille peut-elle être également une classe de base ? Si oui donner un exemple. Sinon expliquer pourquoi cela serait impossible.

> ***Indications :***
> *Des éléments de réponse se trouvent sur la figure 1. Observez bien la hiérarchie qui est définie.*

7. Écrire les classes *Etudiant*, *Administratif* et *Enseignant*.
8. Écrire des accesseurs et mutateurs pour chacune des classes (avoir à l'esprit la notion d'héritage facilitera la tâche).

> ***Indications :***
> *A-t-on besoin de réécrire les accesseurs et mutateurs dans un classe fille s'ils sont présents dans la classe mère ?*

Exercice 2 : Construction d'objets et l'initialisation d'objets

1. Écrire une classe *Execution* qui contiendra une méthode *main()*.
2. Dans la classe *Execution*, créer (en complétant les informations manquantes par des valeurs de votre choix) :
   - Etudiants :
     - Dupont Alice 21 ans, promotion "Pascal"
     - Dujardin Benjamin 22 ans, promotion "Ohm"
     - 2 étudiants "anonymes" promotion "Promotion inconnue"
   - Administratif :
     - Dupont Béatrice 19 ans, secrétaire
   - Enseignant :
     - Machin Boris 36 ans, qui enseigne l'informatique et l'électronique
3. Dans la classes *Personne*, ajouter une méthode *ouMeTrouver()* qui affiche, lorsqu'elle est appelée, le texte suivant "*Je ne sais pas où je suis.*"

Exercice 3 : Redéfinition de méthodes et polymorphisme

1. En utilisant des objets *Etudiant*, *Enseignant* et *Administratif*, appeler la méthode *ouMeTrouver()*.
2. Redéfinir la méthode *ouMeTrouver()* afin que le texte affiché dépende du l'objet utilisé :
   — Etudiants : "*Je suis en cours tous les jours, sauf les jeudis après-midis (je fais du sport, mais c'est comme un cours car j'aime le sport).*"
   — Administratif : "*Le plus facile c'est de passer à mon bureau.*"
   — Enseignant : "*Ce n'est pas la peine de me chercher, je saurai vous trouver !*"
3. Dans la méthode *main()* vérifier que l'utilisation de *ouMeTrouver* permet l'affichage du bon message en fonction du type d'objet.
4. Dans la méthode *main()*, créer un tableau unique (**pas une ArrayList**) dans lequel seront stockés tous les objets créés à l'exercice 2.

> ***Indications :***
> *Pensez à utiliser un type assez générique pour pouvoir englober ceux des différents objets que vous manipulez. Remontrez suffisamment haut dans la hiérarchie.*

5. Quel texte serait affiché si l'on appelle la méthode *ouMeTrouver()* en parcourant le tableau précédemment créé (sans vérifier le type de l'objet lu) ?
6. Afficher le contenu du tableau grâce à des méthodes *toString* à définir dans les différentes classes (rappel : utiliser la notion d'héritage).

> ***Indications :***
> *Vous pouvez afficher le résultat de* `toString` *du parent et le compléter avec des informations propres au type dérivé (dans la classe fille).*

7. Affecter des noms, prénoms, ages, promotions aux étudiants "anonymes" précédemment créés.
8. Afficher à nouveau le contenu du tableau.