TD 1/TP1 - Employés

Programmation Java : Restrictions d'accès, héritage et polymorphisme.

On va représenter à l'aide de classes en Java des employés d'une entreprise. Chaque employé aura un nom et un genre et peut-être un supérieur hiérarchique. Les cadres seront des employés qui auront éventuellement des subordonnés (au plus 10 employés par cadre dans un premier temps). On va utiliser pour cela deux classes, les classes **`Employe`** et **`Cadre`**.

La première partie est constituée de questions "théoriques", la seconde de questions "pratiques". Vous pouvez traiter les deux parties en parallèle.

PARTIE A : TRAVAUX DIRIGÉES

Q.1) - Quel sera le rapport entre les classes **`Employe`** et **`Cadre`** ?

Q.2) - Décrire les attributs de la classe **`Employe`** : on représentera le genre par un booléen et on utilisera deux constantes de cette classe pour le genre `_HOMME` et `_FEMME`.

Q.3) - Quels en-têtes choisir pour restreindre l'accès à certains attributs ?

Q.4) - Quel type d'attribut doit-on ajouter à la classe **`Cadre`** ?

Q.5) - Décrire les constructeurs de ces deux classes.

Q.6) - Quelles sont les possibilités pour que la classe **`Cadre`** puisse accéder aux valeurs des attributs `genre` et `nom` de la classe **`Employe`** ?

Q.7) - On va redéfinir la méthode `public` **`String`** `toString()` de la classe **`Object`** afin qu'elle retourne le nom de l'employé. Pour un cadre, le nom sera précédé la formule de politesse *Mr* ou *Mme* suivant le genre (mais pas pour un simple employé).
Décrire rapidement ce que l'on doit ajouter aux classes **`Employe`** et **`Cadre`**. Quel est le concept de la programmation objet que l'on utilise ?

Q.8) - Le salaire de chacun des employés sera défini à partir d'un salaire de base, égal au smic mensuel brut. Est-il préférable d'utiliser une variable d'instances ou de classe pour représenter ce salaire de base ? Quel est le mot clé que l'on utilisera pour cela ?

Q.9) - On définira une méthode qui retournera le salaire d'un employé avec les règles suivantes :
- pour les simples employés (employés non cadre), il sera égal au salaire de base ;
- pour les cadres il sera égal au salaire de base plus 40% du salaire de chacun de ses subordonnés directs.

Décrire rapidement ce que l'on doit ajouter aux classes **`Employe`** et **`Cadre`**.

PARTIE B : TRAVAUX PRATIQUES

L'objectif de cette partie est d'implémenter les classes **Employe** et **Cadre**. Pour cette partie, créez un nouveau projet **tpemploye** à partir du logiciel **NetBeans** et dans ce projet une classe **Principale** contenant la fonction `main` dans laquelle vous testerez vos différentes méthodes au fur et à mesure. Assurez-vous à la fin du TP d'avoir enregistré vos fichiers sources **.java** dans un répertoire non temporaire (c.à.d. personnel) et/ou sur une clé USB personnelle.

Les classes **Employe** et **Cadre** auront les structures de bases suivantes :

```
public class Employe{
    public static final booelan _HOMME = false;
    public static final boolean _FEMME = true;

    private boolean genre;
    private String nom;
    private Cadre superieur;

    ...
}
```

```
public class Cadre extends Employe {
    private  Employe[] subordonnes;

    ...
}
```

Tous les employés doivent avoir les fonctionnalités suivantes :

Q.10) - Deux constructeurs publiques suivants s'il y a ou non un supérieur.

Q.11) - Des accesseurs publiques aux attributs `genre`, `nom` et `superieur`.

Q.12) - Une redéfinition de la méthode `public` **`String`** `toString()` pour qu'elle retourne le nom de l'employé. Pour un cadre, le nom sera précédé la formule de politesse *Mr* ou *Mme* suivant le genre (mais pas pour un simple employé).

Q.13) - Une méthode **`boolean`** `equals(`**`Employe`** `e)` pour tester l'égalité de deux objets **Employe**. Ces derniers sont considérés égaux s'ils ont même nom et même genre.

Q.14) - Dans la classe cadre, une méthode **`void`** `ajouteSubordonne(`**`Employe`** `e)` qui permet d'ajouter un nouvel employé au service de ce cadre. Attention, elle doit vérifier qu'il n'est pas déjà présent. Cette méthode doit être utilisée pour compléter le constructeur de la classe **Employe** ayant pour argument un supérieur.

Q.15) - Dans la classe cadre, une méthode `public` **`void`** `afficheSubordonnes()` qui affiche sur la sortie standard la liste des subordonnés directs de ce cadre.

Q.16) - Une variable qui représente le salaire de base des employés. Celui-ci sera initialisé à 1487,27€ (smic mensuel brut au 1/01/2017).

Q.17) - Une méthode pour revaloriser ce salaire de base.

Q.18) - Une méthode `public` **`double`** `getSalaire()` retournant le salaire mensuel brut d'un employé :

— pour les simples employés, il sera égal au salaire de base ;

— pour les cadres il sera égal au salaire d'un simple employé plus 40% du salaire de chacun de ses subordonnés directs.

N'oubliez pas de tester au fur et à mesure les méthodes précédentes. Puis,

Q.19) - dans la fonction `main` de la classe **`Principale`** :
- Créer un cadre A qui a pour subordonnés directs un cadre B et deux simples employés.
- Ajouter quatre simples employés au service du cadre B.
- Tester vos méthodes avec ces deux cadres et six simples employés.

Questions facultatives.

Q.20) - Ajouter à la classe **`Employe`** :
- une méthode `public` **`boolean`** `aPourSuperieur(`**`Employe`** `e)` qui teste si l'employé a pour supérieur direct ou indirect l'employé `e`.
- une méthode `public` **`void`** `changementAffectation(`**`Cadre`** `c)` qui change l'affectation de l'employé vers le service du cadre `c`. Ce changement est interdit si l'employé est un supérieur hiérarchique (direct ou indirect) du cadre `c`. Si l'employé était précédemment affecté a un autre cadre, il faut l'enlever des subordonnés de ce cadre.

Q.21) - Ajouter à la classe **`Cadre`** une méthode qui décrit complètement le service de celui-ci sur la sortie standard. La description comprendra la liste complète de tous les subordonnés directs et indirects de ce cadre avec le salaire de chacun.

Q.22) - Enlever la limite de 10 subordonnés directs par cadre.