TP 2 Conception OO avec UML et Java

Vous allez programmer le calcul des salaires mensuels des employés d'une entreprise. Cette entreprise est constituée de mentors `Mentor` qui gèrent une équipes de mentorés `Mentee`.

Un employé n'a pas besoin d'envoyer de message à l'entreprise, i.e. les messages iront toujours de l'entreprise vers les employés. Une entreprise peut avoir zéro ou plusieurs employés. Nous considérons qu'un mentor et un mentoré sont une spécialisation d'un employé. Nous supposons qu'il n'y aura jamais d'instance d'employé puisqu'un objet employé sera toujours un mentor ou un mentee mais jamais qu'un employé. Mentor et mentoré peuvent échanger des messages. Un mentor peut avoir zéro ou plusieurs mentoré.

Une entreprise a un nom. Un employé a un nom, une numéro de paiement et un salaire. On ne pourra pas modifier le nom d'un employé. De plus un employé est spécialiste d'un langage de programmation. Si il programme en Java il obtiendra un bonus de 10 % de son salaire. Un mentor, en plus du bonus sur le langage de programmation, obtiendra un bonus de 5 % par mentoré sous sa responsabilité.

1 Exercice 1

Concevoir le diagramme de classes permettant la réalisation de cette application en respectant les contraintes suivantes :

- Pour ne pas avoir trop de modificateurs vous simplifierez en ne donnant qu'un seul modificateur `setLanguage` pour entrer ou modifier le langage de programmation de l'employé. N'essayez pas de faire du polymorphisme avec cette méthode.
- Les classes comporteront au moins 2 constructeurs : un qui prend en paramètre le nom de l'employé, son numéro de paie et son salaire de base et l'autre qui est un constructeur par défaut et qui ne prend aucun paramètre.
- Le calcul des salaires se fera dans la méthode `getSalary()` qui sera utilisée pour faire du polymorphisme.
- La classe `Company` a deux méthodes `addEmployee`. Une pour ajouter un employé à la liste des employés, et une autre pour ajouter un mentoré à la liste des employés et associer ce mentoré à un mentor. Vous utiliserez les méthodes `addMentee` de la classe `Mentor` et `setMentor` de la classe `Mentee` pour respectivement ajouter un mentoré à la liste des mentorés d'un mentor et associer un mentor à un mentoré.
- On affichera le salaire mensuel de chacun des employés sur la console et dans un fichier `report.txt` en appelant respectivement les méthodes `displayEmployees()` et `displayReport(String theFile)`. Ces méthodes parcourrons la collection des employés et appellerons des méthodes d'affichage des informations d'un employée qui devront être utilisées pour faire du polymorphisme.

2 Exercice 2

Implémentez le diagramme de classe ci-dessus en Java en respectant les contraintes suivantes :

- Vous utiliserez des `HashSet` pour gérer les collections.
- Un mentor ne peut avoir que des mentorés qui programment le même langage que lui. Si ce n'est pas le cas, le mentoré est ajouté à la liste des employés mais il ne doit pas être associé à un mentor et vice versa. Un message s'affichera sur la console pour signaler que ce n'est pas possible.
- La méthode `main()` créera l'entreprise, les employés, affichera le salaire mensuel de chacun des employés sur la console et dans un fichier `report.txt` en appelant respectivement les méthodes `displayEmployees()` et `displayReport(String theFile)`. Ces méthodes parcourrons la collection des employés et seront utilisées pour faire du polymorphisme. Aussi il vous est imposé d'utiliser le mot clé `super`.
- L'affichage aura exactement la forme :

```
Calcul des salaires mensuels des employés de Nom_de_lentreprise
Nom : Jean  // Employé qui n'a pas de mentoré.
Numero de paie : 5200
Langage : C++
Salaire de base : 1000.0
```

```
Salaire du mois : 1000.0

Nom : Vincent  // Mentoré
Numero de paie : 5201
Langage : Java
Salaire de base : 1000.0
Son mentor est Marc avec le numero de paie 5203
Salaire du mois: 1100.0

Nom : Camille  // Mentoré
Numero de paie : 5202
Langage : Java
Salaire de base : 1000.0
Son mentor est Marc avec le numero de paie 5203
Salaire du mois: 1100.0

Nom : Marc   // Mentor
Numero de paie : 5203
Langage : Java
Salaire de base : 1210.0
Son (ses) mentoré(s) est(sont):
Vincent avec le numero de paie 5201
Camille avec le numero de paie 5202
Salaire du mois: 1210.0
```

– Vérifiez les calculs des salaires !

3 Exercice 3

Modifier le `main()` afin de récupérer les informations de l'entreprise à partir du fichier `init.txt` sous la forme exacte :

```
Marc,5203,1000,Java
Camille,5202,1000,Java,5203
Vincent,5201,1000,Java,5203
Jean,5200,1000,C++
```

4 Exercice 4

Modifier le `main()` afin de faire apparaître sur le console un menu permettant :
– d'afficher la liste des employés et leurs informations (numéro de paie, nom et langage) sur la console
– de générer le rapport
– de donner les détails d'un employé à partir de la saisie clavier de son numéro de paie
– de modifier le langage d'un employé
– d'ajouter des employés dynamiquement (durant l'exécution).

5 Documents à rendre

La livraison doit comprendre l'ensemble de votre programme ainsi qu'un document qui contiendra :
– Des diagrammes de classe qui modéliseront rigoureusement l'architecture de votre implémentation.
– Des diagrammes de séquence qui décrivent les interactions entre vos classes lors du déroulement des scénarios d'utilisation que vous aurez identifiés.
– L'explication et la justification de la mise en oeuvre des fonctionnalités de l'exercice 4.