Java et la programmation par composants

Licence Pro. Systèmes automatisés, réseaux et informatique industrielle

Durée 3h30 – Mars 2017

Important : **Lisez tout avant de commencer**, il est fortement recommandé de faire un diagramme de classes simple avant de programmer. Tout le code sera commenté brièvement, compilable et exécutable via un script maven,

1°/Des employés.

A partir d'un diagramme de classe UML que vous n'aurez pas à rendre écrire le code Java des classes nécessaires pour représenter le problème suivant :

Le système doit gérer des personnes et des employés. Les personnes ont un identifiant numérique et un nom. Un employé est une personne qui a un salaire horaire (variable d'un employé à l'autre). Un ingénieur est un employé qui touche en plus de son salaire une prime (variable d'un ingénieur à l'autre) et une indemnité (la même pour tout les ingénieurs). Les employés et les ingénieurs ont une méthode float calculerSalaire(int nbHeures). Chaque classe n'aura qu'un seul constructeur qui prend tous les paramètres utiles. Ecrire une classe Exécutable Test1 qui des instances dans des tableaux de Personnes, d'employé et d'iingénieurs et qui les affiche.

2°/Leurs identifiants.

Donner une solution pour calculer automatiquement l'identifiant des employés et être sur qu'ils sont uniques.

Modifier la classe Test1.

3°/Une mission.

On souhaite maintenant représenter une mission. Une mission est un travail qui dure un certain nombre d'heures et qui est réalisé par un ensemble d'employés (qui peuvent être aussi des ingénieurs) et qui nécessite du matériel (voiture, pioche, ordinateur). Chaque matériel à un identifiant numérique, un type (voiture, pioche, ordinateur, ...), un coût horaire (méthode float getCout(int nbHeures)).

Les personnes et les matériels n'appartiennent pas à la même hiérarchie d'héritage (c'est à dire que leur seul ancêtre comment est Object). Compléter le diagramme de classes UML pour représenter une mission et compléter le code Java pour représenter des missions. Vous utiliserez les notions d'aggrétation et/ou de composition et les Collections. Ecrire une classe Test2 qui crée des missions et les affichent.

4°/La facture d'une mission.

Une facture concerne une liste d'ItemDeFacture (qui sont des objets quelconques mais qui doivent tous avoir une méthode int getTarif(nbHeures)), une facture indique un certain nombre d'heures. Une facture possède une méthode getTotal() qui retourne la somme des tarifs des items pour le nombre d'heure indiqué dans la facture.

Sans modifier la hiérarchie d'héritage précédente comment faire pour que les employés et les matériels puissent être utilisés comme des ItemDeFacture. Le tarif d'un employé est son salaire plus une marge de 10% et celui d'un matériel est son coût plus une marge de 20% plus des frais fixes de 15€.

Compléter le diagramme UML et ajouter ce qui est nécessaire au code Java. Modifier Test2 pour qu'elle affiche le cout d'une mission.

5°/Délégation par aggrétation.

Il doit être possible d'invoquer la méthode :

```
int getTarifReduit(int pourcentage, int nbHeures) {return (int) getTarif(nbHeures)*pourcentage/100;}
```

sur tous les ItemDeFacture. **Sans recopier ce code dans plusieurs classes,** donner une solution pour factoriser ce code en utilisant la délégation par agrégation. Modifier votre code pour cela.