

Java DUT 1 Feuille TD8

Université Paris-Est Marne-la-Vallée

On rappelle que l'on doit mettre une seule classe ou interface par fichier.

Exercice 1.—

On définit une classe `Employee` pour représenter les salariés d'une entreprise qui comprend les champs suivants :

```
public class Employee {  
    private final String firstName;  
    private final String lastName;  
    private int salary;  
}
```

Le champ `salary` représentera le salaire brut annuel.

- a) Écrire les accesseurs nécessaires pour cette classe.
- b) Écrire un constructeur qui permet de créer un `Employee`.

```
public Employee (String firstName, String lastName, int salary) {  
}
```

- c) Écrire un deuxième constructeur qui permet de créer un `Employee` dont le salaire n'est pas spécifié. Ce salaire sera provisoirement initialisé à 0. On utilisera l'appel `this()`.

```
public Employee(String firstName, String lastName) {  
}
```

- d) Écrire une méthode `toString` pour renvoyer une représentation d'un `Employee` sous forme de chaîne de caractères.
- e) Écrire une classe `Test` avec une méthode `main` qui teste les méthodes précédentes.

Exercice 2.—

On définit une interface `Company` pour représenter les salariés d'une entreprise qui comprend les méthodes suivantes :

```
interface Company {  
    // add the Employee e to the Company  
    void add(Employee e);  
    // remove the Employee e to the Company  
    void remove(Employee e);  
    // returns the number of Employee of the company  
    int size();  
}
```

- a) Recopiez l'interface `Company`.
- b) Écrire une classe implémentant l'interface `Company` appelé `SetCompany`. Cette classe contiendra un champ `set` qui permet de stocker les `Employee` de la compagnie dans un `Set`. Un `Set` (voir l'interface `Set` du paquetage `util`) permet de stocker des éléments dans un conteneur en évitant les doublons. Chaque `Employee` figurera donc exactement en un seul exemplaire dans le conteneur.

```
public class SetCompagny implements Company {
    private final Set<Employee> set;
}
```

- c) Écrire un constructeur pour la classe `SetCompagny`. Le `Set` créé sera une instance de la classe `HashSet<Employee>`.
- d) Quelles méthodes de la classe `Object` doit-on redéfinir dans la classe `Employee` pour que tout fonctionne correctement ? Pourquoi ? Redéfinissez ces méthodes.
- e) Implémenter toutes les méthodes de l'interface.
- f) Comment peut-on s'assurer qu'une référence `null` n'est pas ajoutée dans le `Set` d'une `Company` ? Faites les modifications nécessaires. Pensez à la méthode statique `Objects.requireNonNull(Employee e)` de la classe `Objects`.
- g) Ajouter dans l'interface une méthode qui renvoie la masse salariale totale.

```
interface Company {
    // ....
    int getTotalSalaryMass();
}
```

- g) Implémenter cette méthode. Vous pouvez utiliser une boucle `foreach` pour parcourir un `Set`.

Exercice 3.—

- a) Modifier la méthode `add` de l'interface `Company` et son implémentation dans `SetCompany` pour qu'elle renvoie un booléen `true` ou `false`, `true` si l'ajout a été fait, `false` si l'ajout n'a pas été fait car l'`Employee` `e` est déjà dans le `Set`.
- b) Modifier aussi la méthode `remove` pour qu'elle renvoie `null` si l'`Employee` n'est pas dans le `Set` et qu'elle renvoie l'`Employee` `e` qui vient d'être supprimé sinon.

```
interface Company {  
    // add the Employee e to the Company  
    boolean add(Employee e);  
    // remove the Employee e to the Compagny  
    Employee remove(Employee e);  
    // returns the number of Employee of the compagny  
    int size();  
}
```

Exercice 4.—

- a) Ajouter des commentaires pour créer une javadoc pour vos classes.
- b) Compiler la javadoc (commande `javadoc -help` sous linux, menu Projet, Générer la javadoc, sous eclipse).