TP 4 — Mini-projet

Introduction

On souhaite réaliser une application de gestion d'un annuaire téléphonique qui permet d'associer à une personne un ou plusieurs numéros de téléphone.

Cette application s'articulera autour des classes et interfaces suivantes :

- `Personne` classe représentant une personne définie par un nom, un prénom et sa civilité (homme, femme, homme trans, femme trans, non genré).
- `NumTel` : classe représentant un numéro de téléphone, défini par le numéro proprement dit et un code qui indique la nature du numéro (numéro de portable, numéro de poste fixe professionnel, numéro de poste fixe à domicile).
- `ListeNumTel` : interface définissant les fonctionnalités d'une liste de numéros de téléphone.
- `Annuaire` : interface définissant les fonctionnalité d'un annuaire téléphonique, c'est à dire d'un ensemble d'associations `<p,ln>` avec `p` une `Personne` et `ln` une `ListeNumTel`.

On vous fournit un certain nombre d'interfaces à implémenter, et des spécifications pour certaines classes à programmer. On vous fournit également une classe `LectureClavier.java`, déjà programmée, qui facilite la gestion des saisies clavier, notamment en vérifiant que les valeurs saisies sont cohérentes avec le type attendu.

Travail à rendre

Vous devez développer les classes `Personne` et `NumTel`, implémenter les interfaces `ListeNumTel` et `Annuaire`. Vous fournirez également des tests unitaires vérifiant la validité du code. Vous travaillerez avec un dépôt Git pour suivre les versions de votre code.

Le contenu précis du rendu attendu sur EPREL vous sera précisé par votre enseignant·e pendant les séances de TD/TP. Il faudra rendre *au moins* les codes sources et le dépôt Git utilisé.

1 Mise en place

1) Télécharger l'archive `annuaire.zip`, contenant les fichiers suivants :

`Annuaire.java` Code de l'interface définissant les services fournis par un annuaire.

`LectureClavier.java` Classe pour faciliter les lectures de valeurs au clavier.

`ListeNumTel.java` Interface définissant les services fournis par une liste de numéros de téléphone.

2) Intégrer ces fichiers sources dans un projet Java nommé `r201_tp4_miniProjet_<nomDeFamille>`.

2 Développement des classes `Personne` et `NumTel`

Visibilité des attributs et méthodes

On veillera à respecter scrupuleusement les principes de la programmation objet : les attributs doivent avoir une visibilité privée et les méthodes une visibilité publique. Tout autre choix de visibilité doit être techniquement justifié.

On veillera de même à faire bon usage des attributs et méthodes statiques si l'on en a besoin.

2.1 La classe `Personne`

On souhaite implémenter une classe pour représenter les informations relatives à une personne. Une personne est caractérisée par trois informations :

- som nom ;
- son prénom ;
- civilité (prenant une valeur parmi les suivantes : M., Mme, Femme trans, Homme trans, Non genré, Non renseignée) [1].

1) Créer (au moins) deux constructeurs, permettant de créer des personnes dont le nom et le prénom sont obligatoires, et la civilité est facultative. Si cette dernière est non renseignée, on lui affectera par défaut la valeur `"Non renseignée"`.

2) Ajouter les *getters* et *setters* nécessaires.*

3) Programmer la méthode `public String toString()`.

L'objectif est maintenant de préparer l'affichage par ordre alphabétique des personnes présentes dans l'annuaire. Pour ce faire, on commence par comparer le nom de famille, puis le prénom, puis la civilité, pour l'ordre lexicographique. On souhaite écrire une méthode qui, étant données deux personnes `p1` et `p2`, renvoie :

- un nombre strictement négatif si `p1` est avant `p2` dans l'ordre lexicographique ;
- `0` si les deux personnes ont le même prénom, le même nom, la même civilité ;
- un nombre strictement négatif si `p1` est après `p2` dans l'ordre lexicographique.

Pour ce faire, on pourra utiliser la méthode `compareTo`, fournie par l'interface `Comparable` de Java.

4) Adapter le code `public class Personne` pour qu'elle implémente l'interface `Comparable`.

5) Proposer un code pour la méthode `public int compareTo(Object p2)` respectant les spécifications ci-dessus.

6) Surcharger de même la méthode `equals`.

1. On fait ici le choix de l'*inclusion*. L'information renseignée peut donc ne pas être reconnue au sens de l'état civil.

Comparaison et égalité entre objets

Le test d'égalité == n'est pas adapté pour tester que deux instances d'objets sont égales. Il faut pour cela utiliser la méthode `equals` ci-dessus. Lorsque l'on souhaite redéfinir cette méthode, il faut alors faire attention à redéfinir aussi la méthode `hashCode()`, conformément à la documentation de la méthode `equals` de la classe `Object`.

Cette subtilité souligne, une fois supplémentaire, l'importance cruciale de *lire et comprendre la documentation* lorsque l'on utilise des outils fournis par le langage ou des bibliothèques externes.

2.2 La classe `NumTel`

Un numéro de téléphone est caractérisé par deux attributs :

- le numéro lui-même ;
- le type de numéro, prenant des valeurs parmi `'T'` (fixe professionnel), `'D'` (fixe domicile), `'P'`(portable personnel), `'F'` (portable professionnel) et `'?'` (type non renseigné).

1) Créer la classe avec ses attributs, ses constructeurs, ses *getters* et ses *setters*.

2) Redéfinir la méthode `public String toString()`.

3) Enfin, redéfinir les méthodes `public int hashCode()` et `public boolean equals(object o)`.

3 Implémentation de l'interface `ListeNumTel`

On dispose maintenant de classes pour représenter des personnes et des numéros de téléphone. Dans cette partie, intègre les fonctionnalités permettant de gérer une *liste de numéros de téléphone*.

Pour ce faire, nous allons créer une classe qui implémente l'interface `ListeNumTel` fournie avec le code initial du projet. Nous utiliserons une collection pour créer une liste d'objets ayant la classe `NumTel`. L'utilisation d'une `ArrayList` est recommandée.

1) Créer une classe `ListeNumTel_<nomDeFamille>`, qui implémente l'interface `ListeNumTel`.

2) Quels sont les attributs de cette classe ?

3) Implémenter toutes les méthodes demandées par l'interface `ListeNumTel`. Pour ce faire, lisez attentivement les commentaires fournis dans cette interface, car ils décrivent les spécifications de chaque méthode.

4) Créer un fichier `ListeNumTel_<nomDeFamille>_test`, pour gérer les tests unitaires associés à cette classe.

Rappel des bonnes pratiques pour la gestion de projet

On rappelle que *chaque code doit être soigneusement testé*. On créera, pour chaque classe, un fichier de tests associé.

On rappelle également que Git doit être utilisé à bon escient.

- Aucun développement sur la branche principale. Le développement doit se faire sur une branche annexe, par exemple nommée `dev`.
- On développe chaque nouvelle fonctionnalité dans une sous-branche de `dev`.
- Les commits doivent être *atomiques* et leur texte de description suffisamment explicite. Les tests unitaires *doivent être suivis* par Git.
- Seul les fichiers sources (et éventuellement certains fichiers de configuration du projet) doivent être suivis par Git. Les fichiers `.class` n'ont pas besoin d'être dans le dépôt (car ils ne font pas partie du code source).

4 Implémentation de l'interface `Annuaire`

Un annuaire est considéré comme un dictionnaire dont chaque clé est une personne et les valeurs sont les numéros de téléphone associés. Par conséquent, une classe qui implémente l'interface `Annuaire` doit disposer d'un attribut qui permet de gérer une telle structure.

On propose d'utiliser la structure `HashMap` fournie par `java.util`.

1) Créer une classe `Annuaire_<nomDeFamille>` qui implémente la classe `Annuaire`.

2) Ajouter l'attribut `HashMap entrees_annuaire`, avec les bons objets comme clés et comme valeurs.

3) Implémenter les méthodes suivantes, demandées par l'interface `Annuaire` :

Annuaire.java

```
1 public void ajouterEntree(Personne p, ListeNumTel nums);
2 public Iterator personnes();
3 public void afficher();
4 public ListeNumTel numeros(Personne p);
```

Implémentation incomplète de l'interface

Si vous implémentez uniquement ces méthodes, la compilation échoue car la classe `Annuaire_<nomDeFamille>` n'implémente pas toutes les méthode de l'interface. Il est donc impossible de tester le code produit.

Pour y remédier, l'astuce consiste à écrire toutes les méthodes demandées, en affichant simplement un message dans la console du type « `En développement` ». On pourra retourner `null` pour les méthodes qui ne sont pas de type `void`.

4) Tester ces méthodes avec le code ci-dessous (à adapter avec le nom que vous avez donné à vos classes et à leurs attributs).

Java

```java
public static void main(String args[]) {
    Annuaire an = new Annuaire_\<nomDeFamille>(); // Instancie un annuaire vide

    // Ajoute des personnes à l'annuaire
    Personne p1 = new Personne("DURAND","Sophie", "Femme");
    an.ajouterEntree(p1, new Ma_ListeNumTel(new NumTel(151171,'D')));
    an.ajouterEntree(new Personne("DUPONT", "Luc", "Homme trans"),
    new Ma_ListeNumTel(new NumTel(151170,'P')));
    an.ajouterEntree(new Personne("LESTIN", "Louis", "Homme"),
    new Ma_ListeNumTel(new NumTel(146761, 'P')));
    an.ajouterEntree(new Personne("CARUETTE", "Carla"),
    new Ma_ListeNumTel(new NumTel(140361, 'P')));
    // imprime l'annuaire
    System.out.println("-----------------------------");
    System.out.println(an);
    System.out.println("-----------------------------");
    // recherche les numéros de Sophie DURAND
    System.out.println("numéros de " + p1);
    System.out.println(an.numeros(p1));
    }
    // Recherche les numéros de Luc DUPONT
    Personne p2 = new Personne("DUPONT", "Luc", "Homme trans");
    System.out.println("numéros de " + p2);
    System.out.println(an.numeros(p2));
}
```

5) Observez que la recherche du numéro pour Luc DUPONT a échoué alors qu'elle a réussi pour Sophie DURAND. Quelle explication donnez-vous à cela ?

6) Modifiez la classe `Personne` pour corriger ce dysfonctionnement.

7) Poursuivre le développement et proposer un ensemble de tests unitaires.

On souhaite maintenant implémenter une nouvelle fonctionnalité :

Annuaire_<nomDeFamille>.java

```java
/** Donne l'ensemble de toutes les personnes de l'annuaire dont le nom
  * débute par une chaîne donnée.
  * @param s1 la chaine pour la recherche.
  * @return l'ensemble des personnes de l'annuaire dont le nom débute par s1
*/
public Set entreesPourChaine(String s1)
```

8) Implémenter cette nouvelle fonctionnalité et la tester.