# Examen Java SMI S6

## Mai 2019

## Prof. Abdessamad Belangour

**Problème** :

Nous souhaitons écrire une petite application en Java qui gère l'association des parents pour un lycée. Un parent peut être un père, une mère ou un tuteur. Tout parent dont l'enfant est inscrit dans ce lycée fait partie automatiquement de l'association. Ces parents sont représentés par un petit groupe de parents qui sont élus et qui sont appelés délégués. Ainsi, on ne peut être délégué si on n'est pas parent. Les différentes classes de l'application sont comme suit :

```
public enum TypeParent {  pere, mere, tuteur }
```

```
public class Eleve implements Comparable{
   private String id;
   private String nom;
   private String prenom;
   private Parent parent;
  public Eleve(String id, String nom, String prenom, Parent parent) { }
  public Eleve(String id, String nom, String prenom) { }
  // on suppose que les getters & setters sont fournis
   @Override
   public int compareTo(Object o) {……}
   }
```

```
public class Parent {

   private String cin;
   private String nom;
   private String prénom;
   private TypeParent type;
   private Set<Eleve> enfants;

   public Parent(String cin, String nom, String prénom, TypeParent type) {
      this.cin = cin;
      this.nom = nom;
      this.prénom = prénom;
      this.type = type;
      this.enfants = ……;
   }
  // on suppose que les getters & setters sont fournis
```

```
public void ajouterEnfant(Eleve eleve) { }
public void ajouterEnfant(String id, String nom, String prenom) { ……. }
public void ajouterEnfants(Set<Eleve> enfants) {}
public Eleve chercherEnfant(String id) {……..}
public boolean supprimerEnfant(String id) {……..}
@Override
   public String toString() {}
```

```
public class CompNbEnfants implements Comparator {
   @Override
   public int compare(Object o1, Object o2) { …….. }
}
```

```
public class ParentInexistantException extends Exception {  }
```

```
public class Lycee {

   private String nom;
   private Set<Parent> parents;
   private Map<String, Parent> délégués;

   public Lycee(String nom) {
      this.nom = nom;
      this.parents =……..;
      this.délégués = ……..;
   }

 // on suppose que les getters & setters sont fournis

   public void ajouterParent(Parent parent) { }
   public Parent chercherParent(String CIN) { ……  }
   public void ajouterDélégué(String CIN) throws ParentInexistantException { …….}
   public void ajouterDélégué(Parent p) throws ParentInexistantException {…… }
   public void ajouterDélégués(String[] tabCIN) throws ParentInexistantException { }
   public void afficherPourcentageTypes() { …. }
   @Override
   public String toString() { }

}
```

```
public class Main {

   public static void main(String[] args) {
      Lycee lycee=new Lycee("Annajah");
      //-------------------------- Parent 1 -------------------------//
      Parent p1=new Parent("X1001", "Alaoui", "Ali", TypeParent.pere);
      lycee.ajouterParent(p1);
      p1.ajouterEnfant("2019/1", "Alaoui", "Adil");
      //-------------------------- Parent 2 -------------------------//
      Parent p2=new Parent("B5003", "Omari", "Salwa", TypeParent.mere);
      lycee.ajouterParent(p2);
      p2.ajouterEnfant("2019/2", "Yousfi", "Hassan");
      p2.ajouterEnfant("2019/3", "Yousfi", "Houssine");
      //-------------------------- Parent 3 -------------------------//
      Parent p3=new Parent("ZR1003", "Zakani", "Yasser", TypeParent.pere);
      lycee.ajouterParent(p3);
      p3.ajouterEnfant("2019/4", "Tahiri", "Samira");
      //-------------------------- Parent 4 -------------------------//
      Parent p4=new Parent("AB1509", "Othmani", "Othman", TypeParent.tuteur);
      lycee.ajouterParent(p4);
      p4.ajouterEnfant("2019/5", "Nassiri", "Youssef");
      p4.ajouterEnfant("2019/6", "Nassiri", "Sara");
      p4.ajouterEnfant("2019/7", "Nassiri", "Ayoub");

         ……..
         ……..
         ……..
         ……..
         ……..
}
}
```

## Questions :

1) Fournir le code de la méthode **compareTo** de la classe Eleve permettant de comparer les objets Eleve par ordre alphabétique des noms.
2) Fournir le code d'initialisation de l'attribut **enfants** dans le constructeur de la classe Parent.
3) Fournir le code de la méthode **ajouterEnfant** de la classe Parent.
4) Fournir le code de la méthode **chercherEnfant** de la classe Parent.
5) Fournir le code de la méthode **supprimerEnfant** de la classe Parent.
6) Fournir le code de la méthode **compare** de la classe CompNbEnfants permettant de comparer des objets Parents par nombre d'enfants.
7) Fournir le code de l'initialisation des attributs **parents** et **délégués** dans le constructeur de la classe Lycee, sachant que les parents doivent être ordonnés par nombre d'enfants.
8) Fournir le code de la méthode **chercherParent** de la classe Lycee.
9) Fournir le code de la méthode **ajouterDélégué(String CIN)** de la classe Lycee.
10) Fournir le code de la méthode **ajouterDélégué(Parent p)** de la classe Lycee.
11) Fournir le code de la méthode **afficherPourcentageTypes** de la classe Lycee qui affiche le pourcentage des pères, le pourcentage des mères et le pourcentage des tuteurs parmi les délégués.
12) Terminer la méthode **main()** de la classe Main par ajout des objets p1, p2, p4 comme délégués et afficher l'objet lycee et le pourcetage des types de parents.

**N. B. :**

- N'écrivez que la partie demandée sur la feuille de l'examen.
- Respectez l'ordre des questions dans la feuille.
- N'oubliez pas de mettre le numéro de la question devant vos réponses.
- Ecrivez avec une écriture claire et lisible.
- Toute tentative de triche découverte dans la correction sera sanctionnée par un zéro.

Bon courage !