Correction examen Java 2015-2016/ Session principale

TODO 1 : *(1 point) (0,5 point) (0,5 point)*

```java
@Override
public int hashCode() {
    int hash = 3;
    hash = 67 * hash + Objects.hashCode(this.nom);
    hash = 67 * hash + this.cin;
    return hash;
}

@Override
public boolean equals(Object obj) {
    if (obj == null) {
        return false;
    }
    if (obj instanceof Etudiant) {
        final Etudiant other = (Etudiant) obj;
        return nom.equals(other.nom);
    }
    return false;
}

@Override
public String toString() {
    return "Student{" + "nom=" + nom + ", cin=" + cin
            + ", age=" + age + ", moyenne=" + moyenne + '}';
}
```

TODO 2 : *(0,5 point)*

```java
public class SetEtudiants implements IEtudiantService<Etudiant> {
```

NB : 0 si implémente l'interface sans mettre la classe Etudiant entre l'expression diamant.

TODO 3 : ***(0,5 point)***

```
public SetEtudiants() {
    etudiants = new HashSet<>();
}
```

TODO 4 : ***(0,5 point)***

```
@Override
public void ajouterEtudiant(Etudiant e) {
    etudiants.add(e);
}
```

TODO 5 : ***(0,5 point)***

```
@Override
public void supprimerEtudiant(Etudiant e) {
    etudiants.remove(e);
}
```

TODO 6 : ***(0,5 point)***

```
@Override
public boolean chercherEtudiant(Etudiant e) {
    return etudiants.contains(e);
}
```

Ou

```
@Override
public boolean chercherEtudiant(Etudiant et) {
    return etudiants.stream().anyMatch(e -> e.equals(et));
}
```

TODO 7 : *(1 point)*

```java
@Override
public boolean chercherEtudiant(int cin) {
    return etudiants.stream().anyMatch(e -> e.getCin() == cin);
}
```

TODO 8 : *(1 point)*

Ou

```java
@Override
public void afficher() {
    etudiants.stream().forEach(System.out::println);
}
```

TODO 9 : *(1,5 point)*

```java
@Override
public TreeSet<Etudiant> trierEtudiants() {
    return etudiants.stream().collect(Collectors
            .toCollection(() -> new TreeSet<>((e1, e2) -> e1.getNom().compareTo(e2.getNom()))));
}
```

Ou

```java
@Override
public TreeSet<Etudiant> trierEtudiants() {
    TreeSet<Etudiant> ets = new TreeSet<>((e1, e2) -> e1.getNom().compareTo(e2.getNom()));
    ets.addAll(etudiants);
    return ets;
}
```

Ou

L'étudiant peut implémenter l'interface comparable ou compartor et travailler avec l'ancienne procédure.

TODO 10 : ***(2 points)***

```
public double sommeDesMoyenne() {
    return etudiants.stream().mapToDouble(e -> e.getMoyenne()).sum();
}
```

TODO 11 : ***(0,5 point)***

```
public University() {
    university = new HashMap<>();
}
```

TODO 12 : ***(0,5 point)***

```
public void ajouterClasse(Classe classe) {
    university.put(classe, new SetEtudiants());
}
```

TODO 13 : ***(1,5 point)***

```
public void ajouterEtudiant(Etudiant e, Classe c) {
    if (university.containsKey(c)) {
        university.get(c).ajouterEtudiant(e);
    } else {
        SetEtudiants etudiants = new SetEtudiants();
        etudiants.ajouterEtudiant(e);
        university.put(c, etudiants);
    }
}
```

NB : 1.5 si l'étudiant traite la condition de la Classe existante ou non

0.5 Si non

TODO 14 : *(1,5 point)*

```
public void deplacerEtudiant(Etudiant e, Classe destination) {
    for (SetEtudiants se : university.values()) {
        if (se.chercherEtudiant(e)) {
            se.supprimerEtudiant(e);
        }
    }
    university.get(destination).ajouterEtudiant(e);
}
```

TODO 15 : *(1 point)*

```
public void permuterEtudiant(Etudiant e1, Etudiant e2) {
    Classe c1 = null, c2 = null;
    for (Classe c : university.keySet()) {
        if (university.get(c).chercherEtudiant(e1)) {
            c1 = c;
        }
        if (university.get(c).chercherEtudiant(e2)) {
            c2 = c;
        }
        if (c1 != null && c2 != null) {
            break;
        }

    }
    university.get(c1).ajouterEtudiant(e2);
    university.get(c1).supprimerEtudiant(e1);
    university.get(c2).ajouterEtudiant(e1);
    university.get(c1).supprimerEtudiant(e2);
}
```

TODO 16 : *(1,5 point)*

```
public void afficherUniversity() {
    university.forEach((k, v) -> {
        System.out.println(k);
        v.afficher();
    });
}
```

Ou

```
public void afficherUniversity() {
    university.entrySet().stream().forEach((e) -> {
        System.out.println(e.getKey());
        e.getValue().afficher();
    });
}
```

NB : 1.5 si l'étudiant parcours les valeurs de la Map et ne les affiche pas directement. 0.75 si il a travaillé avec la solutions si dessous.

```
public void afficherUniversity() {
    university.entrySet().stream().forEach((e) -> {
        System.out.println(e.getKey()+"  "+e.getValue());
    });
}
```

TODO 17 : *(1,5 point)*

```
public void afficherEtudiantAge(Classe c) {
    university.get(c).etudiants.stream()
            .filter(e -> e.getAge() > 20 && e.getAge() < 23)
            .forEach(System.out::println);
}
```

TODO 18 : ***(2 points)***

```java
public List<Etudiant> meilleuresEtudiants() {
    return university.values().stream()
            .map(se -> se.etudiants)
            .reduce((s1, s2) -> {
                s1.addAll(s2);
                return s1;
            }).get().stream()
            .sorted((e1, e2) -> (int) (e1.getMoyenne() - e2.getMoyenne()))
            .limit(10)
            .collect(Collectors.toList());
}
```

Bonne Correction