Exercices de préparation à l'examen final

N.B : Faites les exercices sans documentation et sans ordinateur.

Contexte :

- Un établissement peut avoir un maximum de 200 personnes comme membres (constante de l'interface **Config**) stockés dans un *ArrayList*;
- Certains des membres sont des étudiants;
- Lorsqu'une personne déménage, elle change de ville. Lorsqu'un étudiant déménage, il change d'établissement. On veut quand même pouvoir changer de ville à un étudiant;
- La méthode **equals()** a été redéfinie pour les objets de type **Personne** : 2 personnes sont égales lorsqu'elles ont le même **nom**, indépendamment de la casse des caractères;
- Un étudiant s'inscrit à des cours. Il ne peut pas dépasser 30 heures d'études par semaine (constante de l'interface **Config**);
- Un étudiant peut s'inscrire à un maximum de 8 cours (constante de l'interface **Config**);
- Un étudiant stocke dans un tableau les cours auxquels il est inscrit.
- On vous rappelle qu'un ArrayList d'éléments de type E fournit les méthodes suivantes :

> **boolean add(E e)** : ajoute l'objet **e** à la fin du *ArrayList*
>
> **int size()** : retourne le nombre d'éléments actuellement dans le *ArrayList*
>
> **boolean contains(E e)** : retourne *true* si le *ArrayList* contient un élément qui est égal à **e** (au sens de ***equals()***) et *false*, sinon
>
> **E get(int index)** : retourne l'élément à la position **index**
>
> **ListIterator<E> listIterator()** : retourne un itérateur pour parcourir le *ArrayList* (avec les méthodes **hasNext()** et **next()**)

Considérez le code suivant, ensuite, répondez aux questions :

```
public interface Config {
    int MAX_MEMBRES = 200;
    int MAX_HEURES_ETUDES = 30; //en heures/semaine
    int MAX_COURS = 8; //8 cours par semaine max
}
public class Cours {
    private String sigle;
    private int duree; //en heures/semaine

    public Cours(String sigle, int duree) {
        this.sigle = sigle;
        this.duree = duree;
    }
    public String getSigle() {
        return sigle;
    }
    public int getDuree() {
        return duree;
    }
}
```

```
public class Personne {
    private String nom, ville;

    public Personne(String nom, String ville) {
        this.nom = nom;
        this.ville = ville;
    }
    public void demenage(String nouvelleVille) {
        this.ville = nouvelleVille;
    }
    public String getVille() {
        return this.ville;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null)
            return false;
        if (o instanceof Personne) {
            Personne p = (Personne) o;
            return nom.equalsIgnoreCase(p.nom);
        }
        return false;
    }
}
public class Etudiant extends Personne {
    private String etablissement;
    private double note;
    private Cours[] coursSuivis;
    private int nbCours;

    public Etudiant(String nom, String ville, String etablis, double note) {
```

Question 1 : Votre réponse à la question vient ici

```
    }
    public double getNote() {
        return note;
    }
    @Override
    public void demenage(String nouvelleEtablissement) {
        this.etablissement = nouvelleEtablissement;
    }
    public void setVille(String nouvelleVille) {
```

Question 2 : Votre réponse à la question vient ici

```
    }
    public int getNbHeuresParSemaine() {
```

Question 3 : Votre réponse à la question vient ici

```
    }
    public boolean ajouter(Cours cours) {
```

```
            Question 4 : Votre réponse à la question vient ici
    }
}
public class Etablissement {
    private String nom;
    private ArrayList<Personne> membres;

    public Etablissement(String nom) {
        this.nom = nom;
        membres = new ArrayList<>();
    }
    public boolean ajouter(Personne personne) {
            Question 5 : Votre réponse à la question vient ici
    }
    public int getNbEtudiants() {
            Question 6 : Votre réponse à la question vient ici
    }
    public int getNbMembresParVille(String ville) {
            Question 7 : Votre réponse à la question vient ici
    }
    public double getMoyenneEtudiants() {
        if (membres.size()==0) { //Ou: if (membres.isEmpty()) {
            throw new IllegalStateException("Aucun étudiant");
        }
            Question 8 : Votre réponse à la question vient ici
    }
    public void toutLeMondeDemenageALaval() {
            Question 9 : Votre réponse à la question vient ici
    }
}
```

Question 1 :

Complétez le constructeur de la classe **Etudiant** pour qu'il appelle le constructeur de la classe mère en lui transmettant le nom et la ville. Ensuite, il initialise l'établissement et la note aux valeurs fournies en paramètres, il crée un tableau de 8 éléments pour stocker les cours suivis par l'étudiant et initialise le nombre de cours à 0.

Question 2 :

Complétez la méthode **setVille()** de la classe **Etudiant** pour mettre la ville de l'étudiant à la valeur reçue en paramètre (attention : l'attribut **ville** doit rester privé dans la classe **Personne**).

Question 3 :

Complétez la méthode **getNbHeuresParSemaine()** de la classe **Etudiant** pour calculer et retourner le nombre d'heures d'études de l'étudiant en une semaine (somme des heures qu'il consacre à tous les cours qu'il suit).

Question 4 :

Complétez la méthode **ajouter()** de la classe **Etudiant** pour ajouter le cours à la liste des cours suivis par l'étudiant, si les conditions suivantes sont réunies :

1. Il reste de la place dans le tableau des cours suivis par l'étudiant;
2. L'ajout du cours ne fera pas dépasser le nombre maximum d'heures d'études par semaine;
3. L'étudiant n'est pas déjà en train de suivre le cours (vérifiez par sigle sigle indépendamment de la casse des caractères).

Si l'ajout réussit, la méthode doit retourner *true*, sinon elle doit retourner *false*.

Question 5 :

Complétez la méthode **ajouter()** de la classe **Etablissement** pour ajouter la personne à la liste des membres de l'établissement, si les 2 conditions suivantes sont réunies :

1. La personne n'est pas déjà dans la liste. Pour vérifier si la personne est déjà dans la liste, on doit se baser sur le critère défini par la méthode **equals()** redéfinie dans la classe **Personne;**
2. Le nombre maximum de membres n'est pas atteint.

Si l'ajout est effectué, la méthode doit retourner *true*, sinon elle doit retourner *false*.

Question 6 :

Complétez la méthode **getNbEtudiants()** de la classe **Etablissement** pour calculer et retourner le nombre d'étudiants dans l'établissement.

Question 7 :

Complétez la méthode **getNbMembresParVille()** de la classe **Etablissement** qui reçoit le nom d'une ville et qui calcule puis et retourne le nombre de membres qui habitent cette ville.

Question 8 :

Complétez la méthode **getMoyenneEtudiants()** de la classe **Etablissement** pour calculer et retourner la moyenne des notes de tous les étudiants de l'établissement.

S'il n'y a aucun étudiant, la méthode doit déclencher une **IllegalStateException**.

Question 9 :

Complétez la méthode **toutLeMondeDemenageALaval()** de la classe **Etablissement** pour mettre à Laval la ville de tous les membres de l'établissement.