1 Présentation

Dans ce sujet, nous présentons quelques éléments d'un logiciel pour gérer des réunions. Les noms des classes, attributs et méthodes vous sont indiqués en police de caractères `TrueType`. Les signatures des méthodes ne vous sont pas toujours données, dans ce cas, vous devez les déterminer.

Une `SalleReunion` accueillera des `Reunions`. Nous considérons ici deux catégories de réunions (qui seront des sous-classes de la classe `Reunion`) : la `ReunionPresentiel` dont tous les participants sont sur place et la `VideoConference` dont une partie seulement des participants sont en présentiel, et les autres participants sont sur des sites distants.

2 Représentation des réunions

On ne créera jamais d'instance directe de la classe `Reunion`, seulement des instances de ses sous-classes.
Toutes les réunions (`Reunion`) sont décrites par :
— un sujet (`sujet`),
— une date au format "AAAAMMJJ" (`date`). Par exemple "20191218" représente le 18 décembre 2019,
— une heure de début et une heure de fin, qui sont des nombres entiers entre 8 et 20 (pour représenter des réunions se tenant entre 8h et 20h).
Une réunion en présentiel (`ReunionPresentiel`) est une sorte de réunion avec la particularité suivante :
— un nombre de participants (`nbParticipants`).
Une vidéo-conférence (`VideoConference`) est une sorte de réunion avec les particularités suivantes :
— un nombre de participants présents (`nbParticipantsPresents`),
— un nombre de participants sur les sites distants (`nbParticipantsDistants`),
— un nombre de sites distants connectés (`nbSitesDistants`).

Question 1. Ecrivez en Java les entêtes et attributs des classes représentant les réunions. Pour la suite, vous supposerez qu'un constructeur sans paramètre et les accesseurs de type `get` et `set` existent pour tous les attributs de toutes ces classes. Vous ne les écrivez pas et vous supposez que les accesseurs en modification (`setXXX`) contrôlent les valeurs.

Réponse question 1 (3 PTS)

```
public abstract class Reunion { //0.75
        private String sujet = "sujet inconnu"; //0.25
        private String date = "19700101"; //0.25
        private int heureDebut=8, heureFin=20; //0.25
....
}

public class ReunionPresentiel extends Reunion {        //0.5
        private int nbParticipants; //0.25
....
}

public class VideoConference extends Reunion{ /0.5
        private int nbParticipantsPresents; //0.25
        private int nbParticipantsDistants; //0.25
        private int nbSitesDistants; //0.25
....
}
```

Question 2. Ecrivez en Java, dans les classes `Reunion` et `VideoConference` uniquement, un constructeur avec paramètres permettant d'initialiser les attributs propres aux instances. Indiquez bien dans quelle classe vous écrivez quel constructeur.

Réponse question 2 (1,5 PTS)

```
// Dans Reunion
public Reunion(String sujet, String date, int heureDebut, int heureFin) { //0.25
  this.setSujet(sujet); //0.25 pour les 4 lignes
  this.setDate(date);
  this.setHeureDebut(heureDebut);
  this.setHeureFin(heureFin);
}
//Dans VideoConference
public VideoConference(String sujet, String date, //0.25 pour la signature
                       int heureDebut, int heureFin,
                       int nbParticipantsPresents,
                       int nbParticipantsDistants,
                       int nbSitesDistants) {
  super(sujet, date, heureDebut, heureFin); //0.5
  this.setNbParticipantsPresents(nbParticipantsPresents); //0.25 pour les 3 lignes
  this.setNbParticipantsDistants(nbParticipantsDistants);
  this.setNbSitesDistants(nbSitesDistants);
}
```

Question 3. Ecrivez en Java, dans la classe `Reunion`, une méthode `duree`, qui retourne la durée d'une réunion. Vous l'obtiendrez en soustrayant l'heure de début à l'heure de fin. Par exemple, si la réunion commence à 10h et termine à 13h, la durée est 3h.

Réponse question 3 (1 PT)

```
public int duree() { // 0.25
        return this.heureFin-this.heureDebut; // 0.75
}
```

Question 4. Ecrivez en Java, dans les classes `Reunion` et `VideoConference` uniquement, une méthode `nbPlacesNecessaires` qui retourne le nombre de places nécessaires. Ce nombre de places nécessaires n'est pas défini pour les réunions en général. Pour les vidéoconférences, il s'agit du nombre de participants présents.

Réponse question 4 (2 PTS)

```
// Dans Reunion
public abstract int nbPlacesNecessaire(); // 1
//Dans VideoConference
public int nbPlacesNecessaire() { // 1 pour les 2 lignes
        return this.nbParticipantsPresents;
}
```

Question 5. Ecrivez en Java, pour les classes `Reunion` et `VideoConference` uniquement, une méthode `toString`, retournant une chaîne de caractères représentant l'objet. Indiquez bien dans quelle classe vous écrivez quelle méthode.

- Pour toutes les réunions, cette chaîne contient : la date (gardez le format AAAAMMJJ), l'heure de début, l'heure de fin, la durée, le sujet, et le nombre de places nécessaires.
- Pour les vidéo-conférences, cette chaîne contient comme ci-dessus la date (gardez le format AAAAMMJJ), l'heure de début, l'heure de fin, la durée, le sujet, le nombre de places nécessaires, et on ajoute en plus le nombre de sites distants.

Réponse question 5 (2 PTS)

```
// Dans Reunion
public String toString() {
        return "Reunion du "+this.date+ // 0.25 pour les 4 lignes
                "a "+this.heureDebut+"h"+
                "\nduree de "+this.duree()+"h"+
                "\nsujet :"+ this.sujet+
                "\nnb. de places necessaires :"+ this.nbPlacesNecessaire();      // 0.5
}
//Dans VideoConference
        public String toString() {
                return super.toString()+ // 1
                        "\nnb sites distants : "+this.nbSitesDistants; // 0.25
        }
```

3 Représentation des salles de réunion

Une salle de réunion (`SalleReunion`) a une adresse, un numéro, une capacité (nombre de personnes qu'elle peut accueillir), et une liste de réunions déjà accueillies, en cours ou à venir (réunions du passé, du présent, du futur) .

Question 6. Ecrivez en Java l'entête et les attributs de la classe `SalleReunion` en les initialisant.
Réponse question 6 (1,5 PTS)

```
public class SalleReunion {
        private String adresse= "adresse inconnue"; // 0.5 pour les 3 attributs
        private String numeroSalle = "numero de salle inconnu";
        private int capacite;
        private ArrayList<Reunion> listeReunion = new ArrayList<>(); //1
```

Question 7. Ecrivez en Java, dans la classe `SalleReunion`, une méthode `ajout` permettant de lui ajouter une réunion sans effectuer de vérification.
Réponse question 7 (0,5 PT)

```
        public void ajout(Reunion nr) { // 0.25
                this.listeReunion.add(nr); // 0.25
        }
```

Question 8. Ecrivez en Java, dans la classe `SalleReunion`, une méthode `dureeOccupation` retournant la somme des durées d'occupation des réunions organisées (passé, présent, futur) dans la salle.
Réponse question 8 (2 PTS à distribuer sur le code)

```
public int dureeOccupation() {
        int res = 0;
                for (Reunion r : this.listeReunion) {
                        res = res + r.duree();
                }
        return res;
}
```

Question 9. Ecrivez en Java, dans la classe `SalleReunion`, une méthode `recherche` retournant la liste des réunions organisées (passé, présent, futur) dans cette salle et traitant d'un sujet passé en paramètre à la méthode.
Réponse question 9 (2,5 PTS à distribuer sur le code)

```
public ArrayList<Reunion> recherche(String sujet) {
        ArrayList<Reunion> res = new ArrayList<>();
                for (Reunion r : this.listeReunion) {
                        if (r.getSujet().equals(sujet))
                                res.add(r);
                }
        return res;
}
```

4 Approfondissement (sur 4 points maximum du barême final)

Question 10. Ecrivez en Java, dans la classe `Reunion`, une méthode `static boolean intervalleCorrect(int hd, int hf)` qui retourne vrai si l'heure de fin est bien postérieure à l'heure de début, et si `hd` et `hf` sont bien compris entre 8 et 20.
Réponse question 10 (1 PT)

```
public static boolean intervalleCorrect(int hd, int hf) {
        return hd>=8 && hf <=20 && hf > hd;
}
```

Question 11. Ecrivez en Java, dans la classe `Reunion`, une méthode `static boolean NeSeChevauchentPas(int hd1, int hf1, int hd2, int hf2)` qui retourne vrai si les deux intervalles ne se chevauchent pas. Pour cela, vous vérifierez que `[hd1,hf1]` est entièrement avant `[hd2,hf2]` (ce qui se produit lorsque `hf1 <= hd2`) ou l'inverse (ce qui se produit lorsque `hf2 <= hd1`).
Réponse question 11 (1 PT)

```
public static boolean NeSeChevauchentPas(int hd1, int hf1, int hd2, int hf2)
{
    // vrai si [hd1,hf1] est entierement avant [hd2,hf2]
    // ou [hd2,hf2] est entierement avant [hd1,hf1]
    // on peut commencer par verifier que les intervalles sont corrects
    return (Reunion.intervalleCorrect(hd1, hf1) &&
            Reunion.intervalleCorrect(hd2, hf2) &&
            (hf1 <= hd2 || hf2 <= hd1));
}
```

Question 12. Ecrivez en Java, dans la classe `SalleReunion`, une méthode `boolean salleLibre(String date, int hd, int hf)` retournant vrai si la salle est (ou était) libre à la `date`, entre `hd` et `hf`. Il en existe une si aucune des réunions organisées dans la salle ce jour-là n'a des horaires qui chevauchent l'intervalle `[hd, hf]`.

Réponse question 12 (1 PT)

```
public boolean salleLibre(String date, int hd, int hf)
{
        if (Reunion.intervalleCorrect(hd, hf)){
                for (Reunion r : this.listeReunion) {
                        // la salle n'est pas libre si
                        // les deux creneaux sont le meme jour
                        // et qu'ils se chevauchent
                        if (r.getDate().equals(date) &&
                                ! Reunion.NeSeChevauchentPas(hd,hf,
                                                r.getHeureDebut(),
                                                r.getHeureFin()))
                                return false;
                }
        }
        return true;
}
```

Question 13. Ré-écrivez en Java, dans la classe `SalleReunion`, la méthode `ajout` permettant de lui ajouter une réunion, en ajoutant les conditions selon lesquelles : (1) la capacité de la salle est suffisante, (2) la salle est libre à ce moment-là.

Réponse question 13 (1 PT)

```
public void ajout(Reunion nr) {
        if (this.capacite >= nr.nbPlacesNecessaire() &&
             salleLibre(nr.getDate(),nr.getHeureDebut(),nr.getHeureFin()))
        {
                this.listeReunion.add(nr);
        }
        else {
                System.out.println("la salle est trop petite ou occupee");
        }
}
```