Nom :
Prénom :
Numéro d'étudiant :
Formation :

Eléments de correction du Contrôle 2 - Classes, héritage, ArrayList

Question 1. Savoir utiliser une classe déjà définie. Etudiez le listing de la classe `Tache` (listing 1).

Listing 1 – Tache.java

```
package controleProjetTaches;

public class Tache {
    private String objectif;
    private int dateDebut, dateFin, // codees sous la forme AAAAMMJJ, ex. 20161209
                nbrPersonnesMois;
    private static double coutPersonneMois = 2000;

    public Tache() {
    }

    public Tache(String objectif, int dateDebut, int dateFin, int nbrPersonnesMois) {
        this.setObjectif(objectif);
        this.setDateDebut(dateDebut);
        this.setDateFin(dateFin);
        this.setNbrPersonnesMois(nbrPersonnesMois);
    }

    public String getObjectif() {return objectif;}

    public void setObjectif(String objectif) {this.objectif = objectif;}

    public int getDateDebut() {return dateDebut;}

    public void setDateDebut(int dateDebut) {
        if (dateDebut > 19700101)
            this.dateDebut = dateDebut;
        else this.dateDebut = 19700101;
    }

    public int getDateFin() {return dateFin;}

    public void setDateFin(int dateFin) {
        if (dateFin > this.dateDebut)
            this.dateFin = dateFin;
        else this.dateFin = this.dateDebut;
    }

    public int getNbrPersonnesMois() {return nbrPersonnesMois;}

    public void setNbrPersonnesMois(int nbrPersonnesMois) {this.nbrPersonnesMois = nbrPersonnesMois;}

    public static double getCoutPersonneMois() {return coutPersonneMois;}

    public static void setCoutPersonneMois(double coutPersonneMois)
        {Tache.coutPersonneMois = coutPersonneMois;}

    public String toString() {
        return "Tache [objectif=" + objectif + ", dateDebut=" + dateDebut
                + ", dateFin=" + dateFin + ", nbrPersonnesMois="
                + nbrPersonnesMois + "]";
    }
}
```

Indiquer ce que va afficher le main du listing 2 lors de son exécution :

Listing 2 – Programme.java

```
package controleProjetTaches;
public class ProgrammeTaches {
    public static void main(String[] args) {
        Tache t1 = new Tache("Analyse",20161101,20161130,4);
        System.out.println(t1);
        Tache t2 = new Tache("Conception",1101,1130,4);
        System.out.println(t2);
    }
}
```

Réponse à la question 1 :

Pour la construction du premier objet, les dates sont correctes et la date de fin est bien postérieure à la date de début, donc les attributs prennent les valeurs passées en paramètre au constructeur. Pour la construction du second objet, la date de début n'est pas postérieure à 19700101, donc l'attribut correspondant est affecté à 19700101 ; la date de fin n'est pas postérieure à 19700101, donc l'attribut correspondant reçoit aussi cette valeur. Les instructions `System.out.println(ti)` appelent la méthode `toString` sur l'objet `ti`, puis affichent son résultat.

```
Tache [objectif=Analyse, dateDebut=20161101, dateFin=20161130, nbrPersonnesMois=4]

Tache [objectif=Conception, dateDebut=19700101, dateFin=19700101, nbrPersonnesMois=4]
```

Question 2. Savoir compléter une classe
Ecrire une méthode `coutTache` calculant le coût d'une tâche (produit du nombre de personnes/mois par le coût d'une personne/mois).

Réponse à la question 2 :

```
    public double coutTache(){
        return this.nbrPersonnesMois * Tache.coutPersonneMois;
    }
```

Question 3. Savoir écrire le code d'une sous-classe
a- Ecrire l'entête et les attributs d'une classe représentant une *TacheHPC* (tâche nécessitant des ressources d'un centre de calcul intensif) avec les informations suivantes. Une telle tâche est décrite par le nombre d'heures de calcul utilisées (`nbH`) et le volume de stockage utilisé en giga-octets (`volGO`). La classe comprend également deux attributs de classe (statiques) indiquant le coût d'une heure de calcul (`coutHC`), par défaut égal à 0,05 (euros) et le coût d'un giga-octet (`coutGO`), par défaut égal à 0,02 (euros).
Réponse à la question 3.a :

```
public class TacheHPC extends Tache {
    private int nbH, volGO;
    private static double coutHC =0.05,
                          coutGO = 0.02;

    .......
```

b- Ecrire pour la classe *TacheHPC* un constructeur prenant tous les paramètres nécessaires pour initialiser les attributs d'instance (donc pas les attributs statiques).
Réponse à la question 3.b :

```
    public TacheHPC(String objectif, int dateDebut, int dateFin,
        int nbrPersonnesMois, int nbH, int volGO) {
        super(objectif, dateDebut, dateFin, nbrPersonnesMois);
        this.nbH = nbH;
        this.volGO = volGO;
    }
```

c- Redéfinir (en la spécialisant) dans la classe *TacheHPC* la méthode `coutTache`. Le coût d'une *TacheHPC* correspond au coût d'une tâche auquel on ajoute le coût dû au nombre d'heures de calcul utilisées et le coût dû au volume de stockage utilisé.
Réponse à la question 3.c :

```
    public double coutTache(){
        return super.coutTache()+
            this.nbH * TacheHPC.coutHC+
                this.volGO * TacheHPC.coutGO;
    }
```

Question 4. Savoir manipuler une liste d'objets
a- Ecrire l'entête et les attributs d'une classe `Projet` avec les informations suivantes. Un projet a un nom et se compose de plusieurs tâches. Initialisez les deux attributs au moment de leur déclaration.

Réponse à la question 4.a :

```
public class Projet {
    private String nom="inconnu";
    private ArrayList<Tache> listeTaches = new ArrayList<Tache>();
.........
```

b- Ecrire, pour la classe `Projet`, une méthode permettant d'ajouter, à la fin de la liste des tâches, une tâche qui n'y apparaît pas (si la tâche apparaît déjà dans la liste, la méthode l'indique par un message d'erreur et n'effectue pas l'ajout). De plus, si le projet comporte déjà des tâches, la date de début de la tâche ajoutée doit être supérieure à la date de fin de la dernière tâche du projet (sinon la méthode l'indique par un message d'erreur et n'effectue pas l'ajout).
Réponse à la question 4.b :

```
    public void ajoute(Tache t){
        if (this.listeTaches.contains(t))
            System.out.println("tache deja existante dans le projet");
        else
            if (! this.listeTaches.isEmpty()
                && t.getDateDebut()
                < this.listeTaches.get(this.listeTaches.size()-1).getDateFin())
                System.out.println("la nouvelle tache debute avant la fin
                de la derniere tache actuelle");
            else
                this.listeTaches.add(t);
}
```

c- Ecrire, pour la classe `Projet` , une méthode `coutTotal` calculant et retournant le coût total du projet (somme des coûts des différentes tâches).

Réponse à la question 4.c :

```
public double coutTotal(){
     double somme = 0;
     for (Tache t : this.listeTaches)
          somme += t.coutTache();
     return somme;
}
```

d- Ecrire, pour la classe `Projet`, une méthode `tachesInferieuresPM` prenant comme paramètre un nombre de personne/mois nommé `nbPM` et retournant la liste des tâches dont le nombre de personnes/mois est inférieur ou égal à `nbPM`.

Réponse à la question 4.d :

```
public ArrayList<Tache> tachesInferieuresPM(int pm){
     ArrayList<Tache> res = new ArrayList<Tache>();
     for (Tache t : this.listeTaches)
          if (t.getNbrPersonnesMois() <= pm)
               res.add(t);
     return res;
}
```

e- Ecrire un programme où l'on crée deux tâches (dont une tâche nécessitant du calcul HPC), un projet, puis on ajoute les tâches au projet et enfin on affiche le résultat des deux méthodes définies dans la classe `Projet` (coût total du projet et tâches réclamant moins de 5 personnes/mois).

Réponse à la question 4.e :

```
public static void main(String[] args) {
     Tache t3 = new Tache("Analyse",20161101,20161130,4);
     System.out.println(t3);
     Tache t4 = new TacheHPC("CalculInteret",20161201,20161220,6,10,10);
     System.out.println(t4);
     Tache t5 = new Tache("Conception",20160701,20161240,6);
     System.out.println(t5);
     Projet p = new Projet("SI bancaire");
     p.ajoute(t3);
     p.ajoute(t4);
     p.ajoute(t4);
     p.ajoute(t5);
     System.out.println(p);
     System.out.println(p.coutTotal());
     System.out.println(p.tachesInferieuresPM(5));
}
```