CS312 – Programmation Objet

TP2 : Gestion des emplois du temps

L'objectif de ce TP est de vous familiariser avec l'utilisation de la notion d'héritage dans le cadre du développement d'une application (modeste) destinée à gérer les emplois du temps d'établissements d'enseignement (telle que ADE pour les emplois du temps de l'Esisar) en nous limitant à la gestion des salles et des activités pédagogiques. Toutes les classes de l'application (éventuellement incomplètes) sont données en annexe.

Nous représentons les **salles d'enseignement** à l'aide des classes `Salle`, `SalleCTD`, `SalleTP` et de l'énumération `Discipline`.

Les **groupes d'étudiants** sont modélisés par la classe `Groupe`. Un groupe est caractérisé par son nom et son effectif (par exemple, "3ATP1" et 16 pour les élèves du premier groupe de TP de 3A).

Les **activités pédagogiques** sont, quant à elles, représentées par les classes `Activité` et ses sous-classes `CM`, `TD`, `TP`. Ainsi, une instance de la classe `Activité` correspond à une activité pédagogique (CM, TD ou TP) caractérisée par :

(i) son nom (par exemple, "CS312 TP1")
(ii) les groupes d'étudiants qui lui sont associés (par exemple, le groupe TP1 pour "CS312 TP1"),
(iii) la liste des salles qui techniquement peuvent l'accueillir (salles de cours pour un CM ou un TD, salles de TP pour les TP), indépendamment de leur nombre de places.

1. Questions

(i) Est-il possible de créer une activité pédagogique de type TD et lui associer des salles de TP ?
(ii) Dessiner les objets créés par la séquence de code suivante:

```
SalleCTD a042 = new SalleCTD(100, "A042");
SalleCTD d030 = new SalleCTD(180, "D030");
SalleTP b141 = new SalleTP(16, "B141", Discipline.Informatique);
SalleCTD a048 = new SalleCTD(25, "A048");

CM cs310cm = new CM("CM CS310");
cs310cm.addSalle(a042);
cs310cm.addSalle(d030);
cs310cm.addSalle(a048);

TP cs330tp1 = new TP("TP1 CS330");
cs330tp1.addSalle(b141);

Groupe a3tp1 = new Groupe("3ATP1", 16);
Groupe a3tp2 = new Groupe("3ATP2", 16);
Groupe a3tp3 = new Groupe("3ATP3", 16);
Groupe a3tp4 = new Groupe("3ATP4", 16);
Groupe a3tp5 = new Groupe("3ATP5", 16);

cs310cm.addGroupe(a3tp1);
cs310cm.addGroupe(a3tp2);
cs310cm.addGroupe(a3tp3);
cs310cm.addGroupe(a3tp4);
cs310cm.addGroupe(a3tp5);
cs330tp1.addGroupe(a3tp1);
```

La classe `Créneau` **permet d'associer une activité pédagogique à une salle effective dans une tranche horaire donnée**. Ainsi, son constructeur reçoit en paramètre un horaire (année, mois, jour, heure et minute de début, durée), une activité et une salle. Il procède à trois vérifications réalisées par les trois méthodes privées `vérifSalle`, `vérifDurée`, `vérifCapacité`.

2. Questions

(iii) Réaliser la méthode `vérifDurée` qui retourne vrai si et seulement si le créneau horaire est contenu entre 8h et 19h.
(iv) Réaliser la méthode `vérifSalle` qui retourne vrai si et seulement si la salle du créneau est bien une des salles appropriées pour l'activité.
(v) Réaliser la méthode `vérifCapacité` qui retourne vrai si la salle du créneau a une capacité au moins égale à la somme des effectifs des groupes associés à l'activité.

La classe principale de notre application est la classe `Planning`. Elle gère une liste de créneaux en effectuant quelques opérations de vérification et de consultation. La méthode `addCréneau`, notamment, permet d'insérer dans l'emploi du temps un créneau préalablement créé. Avant l'insertion du créneau, il faut s'assurer qu'il est compatible avec ceux déjà insérés ou, autrement dit, il ne présente pas d'intersection avec eux. Deux créneaux ont une intersection si leurs tranches horaires se chevauchent et, soit ils occupent la même salle, soit ils sont associés à des activités ayant des groupes d'élèves en commun. Par exemple, à la suite du code proposé à la question (ii), les instructions

```
c1 = new Créneau(2014, 1, 17, 13, 15, 105, a042, cs310cm);
c2 = new Créneau(2014, 1, 17, 13, 15, 210, b141, cs330tp1);
```

créent deux créneaux qui ne pourront pas coexister dans l'emploi du temps étant donné qu'ils concernent tous les deux le même groupe `a3tp1`. Ainsi, la méthode `addCréneau` n'ajoute à l'emploi du temps un créneau qu'après avoir appelé la méthode `vérifCréneau` qui effectue cette vérification, en faisant elle-même appel à la méthode `intersection` de la classe `Créneau`.

3. Questions

(vi) Réaliser la méthode `intersection` de la classe `Créneau`.
(vii) Réaliser la méthode `planningGroupe` de la classe `Planning` qui, pour un groupe donné, retourne la liste des créneaux qui lui sont associés dans l'emploi du temps.
(viii) Réaliser la méthode `totalHeuresGroupe` de la classe `Planning` qui, pour un groupe donné, retourne le nombre total d'heures d'enseignement le concernant dans l'emploi du temps.
(ix) Quel est le résultat de l'exécution de la méthode main de la classe `Planning` (ne donner que ce qui est affiché à l'écran)?

Cette première version de l'application peut être améliorée de plusieurs manières, dont deux exemples suivent.

(x) Actuellement, rien n'empêche qu'un TP d'automatique se déroule dans une salle de TP d'informatique. Quelles modifications doit-on apporter à l'application pour que les TP d'une discipline donnée ne puissent se dérouler que dans des salles de TP de cette discipline?
(xi) Dans la réalité, la notion de groupe est plus riche que sa représentation par la classe `Groupe`. En effet, la totalité de la promotion p2018 constitue un groupe décomposé en groupes de TD et groupes de TP. On peut donc naturellement souhaiter pouvoir introduire cette notion de dépendance (ou appartenance) entre groupes. Par exemple, si a3promo est le groupe associé à l'ensemble de la promotion de 3A et a3tp1 est le premier groupe de TP, la séquence de code suivante ne devrait pas ajouter le créneau `c2` dans l'emploi du temps.

```
cs310cm.addGroupe(a3promo);
cs330tp1.addGroupe(a3tp1);

Créneau c1 = null, c2 = null;

c1 = new Créneau(2014, 1, 17, 13, 15, 105, a042, cs310cm);
c2 = new Créneau(2014, 1, 17, 13, 15, 210, b141, cs330tp1);


p.addCréneau(c1);
p.addCréneau(c2);
```

Comment peut-on modifier la définition des classes pour atteindre ce résultat?

ANNEXE

```
public class Salle {
    private int capacité;
    private String nom;

    public Salle(int c, String n){
        capacité = c;
        nom = new String(n);
    }

    public String toString(){
        return nom + " (" + capacité + " places)";
    }

    public int getCapacité(){
        return capacité;
    }

    public String getNom(){
        return nom;
    }
}

public class SalleCTD extends Salle{
    public SalleCTD(int capacité, String nom){
        super(capacité, nom);
    }

    public String toString(){
        return "Salle cours-TD "+ super.toString();
    }
}

public class SalleTP extends Salle{

    private Discipline type;

    public SalleTP(int capacité, String nom, Discipline d){
        super(capacité, nom);
        type = d;
    }

    public String toString(){
        return "Salle TP "+ type + " " + super.toString();
    }
}

public enum Discipline {
    Automatique,
    Informatique,
    Electronique,
    Réseau;
}
```

```java
import java.util.*;


 public class Activité {
    private String nom;
    private ArrayList<Groupe> groupes;
    private ArrayList<Salle> sallesAppropriées;

    public Activité(String nom){
        this.nom = new String(nom);
        groupes = new ArrayList<Groupe>();
        sallesAppropriées = new ArrayList<Salle>();
    }

    public void addGroupe(Groupe groupe){
        groupes.add(groupe);
    }

    protected void addSalle(Salle s){
        sallesAppropriées.add(s);
    }

    public ArrayList<Salle>  getSalles(){
        return sallesAppropriées;
    }

    public ArrayList<Groupe> getGroupes(){
        return groupes;
    }

    public String toString(){
        return "Activité " + nom + " (" + groupes +")";
    }
}

public class CM extends Activité{
    public CM(String nom){
        super(nom);
    }

    public void addSalle(SalleCTD s){
        super.addSalle(s);
    }
}

public class TD extends Activité{
    public TD(String nom){
        super(nom);
    }

    public void addSalle(SalleCTD s){
        super.addSalle(s);
    }
}

public class TP extends Activité{
    public TP(String nom){
        super(nom);
    }

    public void addSalle(SalleTP s){
        super.addSalle(s);
    }
}
```

```java
public class Créneau {
    private int année;
    private int mois; // 1 à 12
    private int jour; // 1 à 31
    private int heure; // 0 à 23
    private int minute; // 0 à 59
    private int durée; // en minutes, maximum 210

    private Salle salle;
    private Activité activité;

    public Créneau(int a, int m, int j, int h, int min, int d, Salle s, Activité ac)
    {
        année = a; mois = m; jour = j; heure = h; minute = min; durée = d;
        salle = s;
        activité = ac;

        if(!vérifCapacité()){
            System.exit(1);
        }
        if(!vérifDurée()){
            System.exit(1);
        }
        if(!vérifSalle()){
            System.exit(1);
        }
    }

    private boolean vérifSalle(){
        // COMPLETER
    }

    private boolean vérifCapacité(){
        // COMPLETER
    }

    private boolean vérifDurée(){
        // COMPLETER
    }

    public Salle getSalle(){ return salle;}

    public Activité  getActivité(){return activité;}

    public int  getDurée(){ return durée;}

    public String toString(){
        return jour + "/" + mois + "/" + année + " " + heure + ":" + minute +" (" +
               durée +") : " + activité + " " + salle;
    }

    public boolean intersection(Créneau c){
        // COMPLETER


}
```

```java
public class Groupe{
    private String nom;
    private int effectif;

    public Groupe(String n, int e){
        nom = new String(n);
        effectif = e;
    }

    public int getEffectif(){
        return effectif;
    }

    public String getNom(){
        return nom;
    }

    public String toString(){
        return nom + " (" + effectif + " étudiants)";
    }
}
```

```java
import java.util.*;

public class Planning {
    private ArrayList<Créneau> edt;

    public Planning(){
        edt = new ArrayList<Créneau>();
    }

    public String toString(){
        String res = new String();
        for (Créneau c : edt){
            res = res + c + "\n";
        }
        return res;
    }

    public boolean vérifCréneau(Créneau c){
        boolean ok = true;
        int i = 0;
        while (ok && i < edt.size()){
            Créneau cour = edt.get(i++);
            ok = !c.intersection(cour);
        }
        return ok;
    }

    public void addCréneau(Créneau c){
        if (this.vérifCréneau(c)) edt.add(c);
        else System.out.println("Créneau " + c + " non inséré");
    }

    public ArrayList<Créneau> planningGroupe(Groupe groupe){
        // COMPLETER
    }

    public float totalHeuresGroupe(Groupe groupe){
        // COMPLETER
    }

    public static void main (String [] args){
        Planning p = new Planning();

        SalleCTD a042 = new SalleCTD(100, "A042");
```

```
        SalleCTD d030 = new SalleCTD(180, "D030");
        SalleTP b141 = new SalleTP(16, "B141", Discipline.Informatique);
        SalleCTD a048 = new SalleCTD(25, "A048");

        CM cs310cm = new CM("CM CS310");
        cs310cm.addSalle(a042);
        cs310cm.addSalle(d030);
        cs310cm.addSalle(a048);

        TP cs330tp1 = new TP("TP1 CS330");
        cs330tp1.addSalle(b141);

        CM cs410cm = new CM("CM CS410");
        cs410cm.addSalle(a042);
        cs410cm.addSalle(d030);
        cs410cm.addSalle(a048);

        Groupe a3tp1 = new Groupe("3ATP1", 16);
        Groupe a3tp2 = new Groupe("3ATP2", 16);
        Groupe a3tp3 = new Groupe("3ATP3", 16);
        Groupe a3tp4 = new Groupe("3ATP4", 16);
        Groupe a3tp5 = new Groupe("3ATP5", 16);

        Groupe a4ir = new Groupe("4AIR", 29);

        cs310cm.addGroupe(a3tp1);
        cs310cm.addGroupe(a3tp2);
        cs310cm.addGroupe(a3tp3);
        cs310cm.addGroupe(a3tp4);
        cs310cm.addGroupe(a3tp5);

        cs330tp1.addGroupe(a3tp1);
        cs410cm.addGroupe(a4ir);

        Créneau c1 = null, c2 = null, c3 = null, c4 = null;

        c1 = new Créneau(2014, 1, 17, 13, 15, 105, a042, cs310cm);
        c2 = new Créneau(2014, 1, 17, 8, 00, 210, b141, cs330tp1);
        c3 = new Créneau(2014, 1, 17, 15, 15, 105, d030, cs410cm);
        c4 = new Créneau(2014, 1, 17, 10, 00, 105, a042, cs310cm);

        p.addCréneau(c1);
        p.addCréneau(c2);
        p.addCréneau(c3);
        p.addCréneau(c4);

        System.out.println(p.planningGroupe(a3tp1) + " (" +
                           p.totalHeuresGroupe(a3tp1) + ")");
        System.out.println(p.planningGroupe(a4ir));
    }

}
```