Université Mohammed V
FACULTE DES SCIENCES
RABAT
Département informatique

Série d'exercices N° 2
Classes et Objets

Filière SMI –Semestre 5
Travaux dirigés / Travaux pratiques
Programmation orientée objet

Année universitaire
2015/2016

L'objectif de cette série de Travaux Dirigés/Pratiques et de pratiquer les notions de classes et d'objets. L'application est constituée de trois classes(Module, Note et Étudiant). La classe *Module* décrit un Module et a été traitée comme exemple à la séance de cours. Les deux classes *Note* et *Etudiant* dont les descriptions sont données ci-dessous restent à développer suivant les fonctionnalités décrites comme commentaires dans le corps des méthodes correspondantes.

```
public class Note {

     private Module mod;
     private double noteExF;
     private double noteCc;
     private double noteTp;

     public Note() {}
// Définir les constructeurs suivants
     public Note(Module mod, double noteExF, double noteCc, double noteTp)
     public Note(Module mod)
     public Note(String nomMod)
     public Note(String nomMod, double note)
     public Note(String nomModule, double noteExF, double noteCc, double noteTp)


     public double noteModule(){// Calcule et retourne la note du module
     //correspondant à la note de l'instance courante.}

     public boolean noteEquals (Note n){// Retourne true si les valeurs des notes
     //des modules correspondants à l'instance courante et à l'objet n ont sont
     //égales.}

     public boolean equals (Note n){// Retourne true les valeurs des notes
     //des modules correspondants à l'instance courante et à l'objet n ont sont
     //égales et les noms des modules de leurs attributs mod sont aussi égales
     //modules correspondants. Ne pas utiliser les getters/setters. }

     public String toString(){// Retourne une représentation textuelle d'une
     //instance de la classe Note.}

     public int compareTo(Note n){// Retourne 1 si la note du module de l'instance
     //courante est strictement supérieure à la note du module de l'objet n, -1 si
     //elle en est strictement inférieure ou 0 sinon.}

     private boolean estNoteValide(double n){// Retourne true si 0.0<=n<=20.0,
     //false sinon.}

     public boolean estNoteModule_Valide(){// Retourne true si
     //0.0<=note du module<=20.0, false sinon.}
}
```

```
public class Etudiant {
    String code;
    String nom;
    Note[] notes;
    int dim,der=-1;

    public Etudiant(int dim)
    public boolean vide(){ return der==-1;}
    public boolean plein(){ return der==dim-1;}
    public String toString()

    public int localiser(String nomMod){// Retourne une valeur positive ou nulle
    //représentant la position d'une note du module donné en argument si la note
    //de ce module existe dans le tableau notes. Retourne -1 si la note du module
    //donné en argument n'existe pas dans le tableau.}

    public void ajouter(Note n){// Ajoute l'entrée n de la méthode tableau notes.}

    public boolean aValidé( String nomMod){// Retourne true si la note du module
    //en argument est supérieure à 10.0, false sinon.}

    public boolean aToutValidé(){// Retourne true si tous les modules de l'instance
    //courante ont été validés, false sinon.}

    public Etudiant modulesValidés(double nt){// Retourne un objet Etudiant
    //contenant les notes des modules (de l'instance courante) qui sont supérieures
    //à la valeur de la varaible nt.}



}
```

La classe *Note* est associée à la classe *Module* par la présence d'une référence de la dernière dans la description de la classe Note. Cette association se lit "Une *note* correspond à un *module*".

Un étudiant de la classe *Etudiant* a plusieurs *notes* correspondants au *modules* auxquels il est inscrit. La relation d'inscription est implicite par la présence des notes des modules (auxquels l'étudiant est inscrit) dans la classe Etudiant. La relation (association) est explicitée par la référence *notes* au tableau des notes d'un étudiant.

L'attribut *dim* de la classe *Etudiant* représente le nombre maximum des modules auxquels un étudiant peut être inscrit. Par exemple un étudiant de SMI peut inscrire au maximum à 6 modules en S1, 12 en S2, 18 en S3, etc.
L'attribut der de la classe *Etudiant* quant à lui représente le nombre de modules auxquels un étudiant est effectivement inscrit. Il est mis à jour (incrémenté) à chaque nouvelle inscription à un module. Une inscription est réalisée par la méthode -***public void*** *ajouter(Note n)*- qui insère une nouvelle note donnée en argument.