Modélisation et programmation par objets 1, examen

Documents autorisés : une feuille A4 recto-verso. Durée : 2h.
Réponses à donner sur les feuilles fournies. Placer votre numéro d'anonymat sur chacune (en haut).
Le barème est donné à titre indicatif.

1 Spécialisation, généralisation, exceptions, associations, collections

On se place dans le cadre d'une université fictive. Chaque étudiant y a des résultats semestriels pour son semestre en cours. Une première modélisation très simplifiée des résultats semestriels (à compléter) est donnée à la figure 3. Les résultats semestriels sont destinés à contenir tous les résultats de l'étudiant pour un semestre (un résultat par unité d'enseignement (UE) est prévu). Si les résultats sont tous saisis, le booléen saisieTerminée est à vrai, sinon il est à faux. Un résultat pour une UE est ici caractérisé par la note obtenue (sur 20), le nombre de crédits ECTS [1] de l'UE et le nom de l'UE (c'est ici une modélisation très simplifiée). Selon le diplôme auquel l'étudiant est inscrit, les résultats semestriels peuvent être avec compensation ou sans compensation.

On prévoit pour les résultats semestriels quelques méthodes, seule `nbTotalECTS` est déjà présente sur le diagramme :

- `nbTotalECTS` qui détermine pour un semestre le nombre maximal d'ECTS pouvant être validés lors du semestre, c'est la somme des ECTS associés à chaque résultat d'UE d'un résultat semestriel. Ce nombre est en général 30 mais peut être supérieur dans certaines formations.
- `moyenne` qui détermine pour un semestre la moyenne obtenue. Cette moyenne est celle des notes obtenues, pondérées par le nombre d'ECTS auxquelles elles correspondent. Cette moyenne est donc calculée comme la somme pour toutes les UEs des produits de la note obtenue pour l'UE par le nombre d'ECTS de l'UE, divisée par le nombre total d'ECTS.
- `semestreValidé` qui détermine si le semestre est validé au vu des résultats semestriels. Un semestre avec compensation est validé si la moyenne obtenue est supérieure ou égale à 10. Un semestre sans compensation est validé si la note de chaque UE est supérieure ou égale à 10.
- `nbECTSacquis` qui détermine combien d'ECTS sont acquis pour un résultat semestriel donné. Pour un résultat semestriel avec compensation, il s'agit de la somme des ECTS des UE acquises (dont la note est supérieure à 10) si la moyenne est inférieure à 10, et sinon il s'agit du nombre total d'ECTS pouvant être acquis dans le semestre. Pour un résultat semestriel sans compensation, il s'agit de la somme des ECTS des UE acquises (dont la note est supérieure à 10).

Le code Java de la classe ResultatUE vous est donné au listing 1.

Question 1. (5 points) Complétez le diagramme de la figure 3 et le code des listings 2, 3 et 4 de manière à y ajouter les méthodes manquantes : `moyenne`, `semestreValidé` et `nbECTSacquis`. Il s'agit ici de :

- décider où placer les méthodes dans la hiérarchie
- décider de la signature des méthodes (y compris décider lesquelles seraient abstraites)
- placer les méthodes ainsi élaborées sur le diagramme de classe
- implémenter les méthodes ainsi élaborées dans les classes Java, en étant cohérent avec le modèle UML.

1. dans la suite on abrégera crédits ECTS en ECTS)

Listing 1 – Code Java de ResultatUE

```
public class ResultatUE {
    private float note;
    private int nbECTS;
    private String nomUE;

    public ResultatUE(float note, int nbECTS, String nomUE) {
        this.note = note;
        this.nbECTS = nbECTS;
        this.nomUE = nomUE;
    }

    public float getNote() {return note;}
    public void setNote(float note) {this.note = note;}
    public int getNbECTS() {return nbECTS;}
    public String getNomUE() {return nomUE;}
}
```

Question 2. (4 points) Exceptions et implémentation d'association.

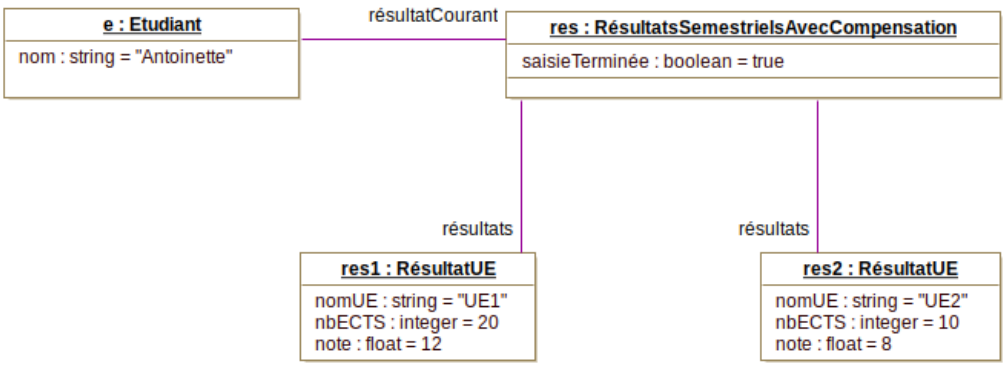
a- Dans la question précédente, nous n'avons pas pris en compte le fait que les notes pouvaient ne pas encore avoir été saisies lors des calculs (de moyenne par exemple). Donnez une nouvelle version du code de la méthode de calcul de moyenne (et uniquement celle-ci) qui cette fois-ci va retourner la moyenne (comme expliqué précédemment) si le booléen saisieTerminée est à vrai, et lever une exception NotesNonSaisiesException dans le cas contraire. Vous donnerez également l'implémentation Java de NotesNonSaisiesException.

FIGURE 1 – Etudiant et résultat courant

b- Dans la classe `Etudiant` (voir figure 1), on a une méthode d'édition de résultats, qui retourne une chaîne synthétisant les résultats pour le semestre courant de l'étudiant. Écrire en Java pour la classe `Etudiant` la déclaration de ses attributs et le code de la méthode d'édition des résultats. Pour simplifier, cette méthode retourne : la moyenne de l'étudiant si le calcul de la moyenne s'effectue correctement sans lever d'exception, et la chaîne "résultats indisponibles" dans le cas contraire.

Question 3. (2 points) On donne à la figure 2 un diagramme d'instances représentant un étudiant et ses résultats semestriels actuels. Donnez en Java l'ensemble des instructions nécessaires pour obtenir en mémoire les mêmes objets que ceux illustrés dans ce diagramme d'objets. Si vous devez ajouter des méthodes à celles déjà données dans l'énoncé ou ajoutées aux questions précédentes, vous en donnerez également l'implémentation.

FIGURE 2 – Un diagramme d'instances



Question 4. (4 points) On décide finalement que les résultats des UEs (présents dans les résultats semestriels) seront indexés par le nom de l'UE. Pour un résultat semestriel, chaque nom d'UE correspond à au plus une instance de ResultatUE. Donnez à la figure 4 et au listing 5 :

- la nouvelle version de l'association liant les résultats semestriels aux résultats des UE (modélisation UML)
- la nouvelle façon d'encoder cette association dans la classe des résultats semestriels (code Java)
- la nouvelle version du code des méthodes nbTotalECTS et addResultat (et uniquement de celles-ci) dans la classe des résultats semestriels (en Java).
- l'implémentation en Java d'une nouvelle méthode dans la classe des résultats semestriels qui retourne le résultat de l'UE dont le nom est passé en paramètre (et qui retourne null s'il n'y a pas parmi les résultats d'UE de ce nom).

2 Modélisation

Question 5. (5 points) On s'intéresse à la modélisation d'un site gérant des cagnottes : en suivant un lien d'invitation, des participants peuvent donner de l'argent pour un cadeau commun, aider un jeune entrepreneur, ou tout autre événement nécessitant finance. Modélisez la situation suivante à l'aide d'un diagramme de classes UML. Une cagnotte a un nom et un descriptif. Par exemple on peut envisager la cagnotte de nom "20 ans Akram" et de descriptif "cagnotte pour le cadeau pour les 20 ans d'Akram". Une cagnotte est organisée par un participant inscrit sur le site. Les donneurs sont des participants inscrits ou non inscrits sur le site. Pour tous les participants (inscrits ou non inscrits), on mémorise leur adresse de courrier électronique (mail) et éventuellement leur nom. Pour les participants inscrits, on mémorise en plus leur mot de passe (normalement pas en clair mais ici on ne s'intéressera pas à cette question). Les participants peuvent donc faire des dons dans une cagnotte ; un même participant peut donner dans plusieurs cagnottes différentes. Les dons sont d'un certain montant (en euros), on en mémorise la date (date où le don a été fait, ici représentée par un entier pour simplifier). Les dons peuvent être anonymes ou pas (ce qui sert lors de la communication des informations sur les dons sur le site et pour l'organisateur). On peut par exemple imaginer que dans la cagnotte "20 ans d'Akram" organisée par Svetlana (svetlana@freez.fr, de mot de passe "maythefourth") il y ait eu 2 dons :

- l'un anonyme le 01 avril 2021 de 20 euros de Bérangère (berangere@youhou.fr) qui est une participante non inscrite,
- l'autre non anonyme le 4 mai 2021 de 10 euros de Carmen Ramirez (c.ramirez@googoo.com) qui est une participante inscrite (de mot de passe "bizet34").

et que Carmen Ramirez ait également donné 15 euros le 1er mai 2021 dans la cagnotte "retraite de Yolande".

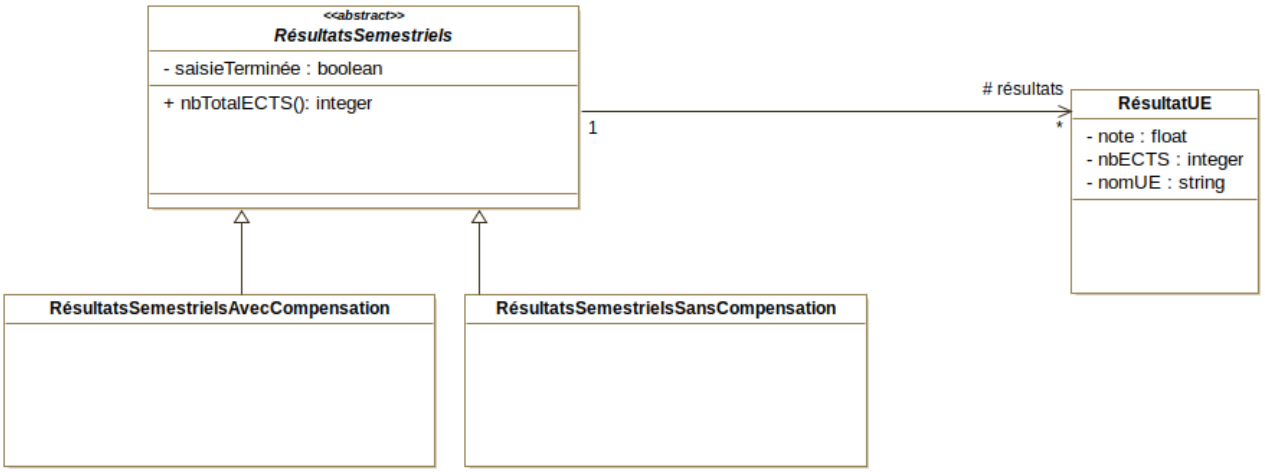
Vous placerez dans votre modèle : les classes, relations entre classes (héritage, association) et attributs que vous jugerez adéquats. Vous ne placerez pas dans votre modèle les constructeurs, ni les méthodes à l'exception de celles décrites ci-dessous :

- une méthode permettant un don (d'un participant, pour un certain montant, à la date courante, anonyme ou pas) dans une cagnotte.
- une méthode permettant de lister tous les donneurs d'une cagnotte.
- une méthode permettant d'obtenir le total des dons dans une cagnotte.

3 Partie à compléter

Question 1.

FIGURE 3 – Diagramme de classes à compléter



Listing 2 – Code Java de ResultatsSemestriels

```
public abstract class ResultatsSemestriels {
    protected ArrayList<ResultatUE> resultats =new ArrayList<ResultatUE>();
    private boolean saisieTerminee;

    public int nbTotalECTS() {
        int total=0;
        for (ResultatUE res:resultats) {
            total+=res.getNbECTS();
        }
        return total;
    }

    public boolean isSaisieTerminee() {
        return saisieTerminee;
    }

    public void setSaisieTerminee(boolean saisieTerminee) {
        this.saisieTerminee = saisieTerminee;
    }

    public void addResultat(ResultatUE res) {
        resultats.add(res);
    }
```

}

Listing 3 – Code Java de ResultatsSemestrielsAvecCompensation

```
public class ResultatsSemestrielsAvecCompensation extends ResultatsSemestriels {

}
```

Listing 4 – Code Java de ResultatsSemestrielsSansCompensation

```
public class ResultatsSemestrielsSansCompensation  extends ResultatsSemestriels{

}
```

Question 2.

a-

b-

Question 3.

Question 4.

FIGURE 4 – Diagramme à compléter

Listing 5 – Code Java de ResultatsSemestriels à compléter

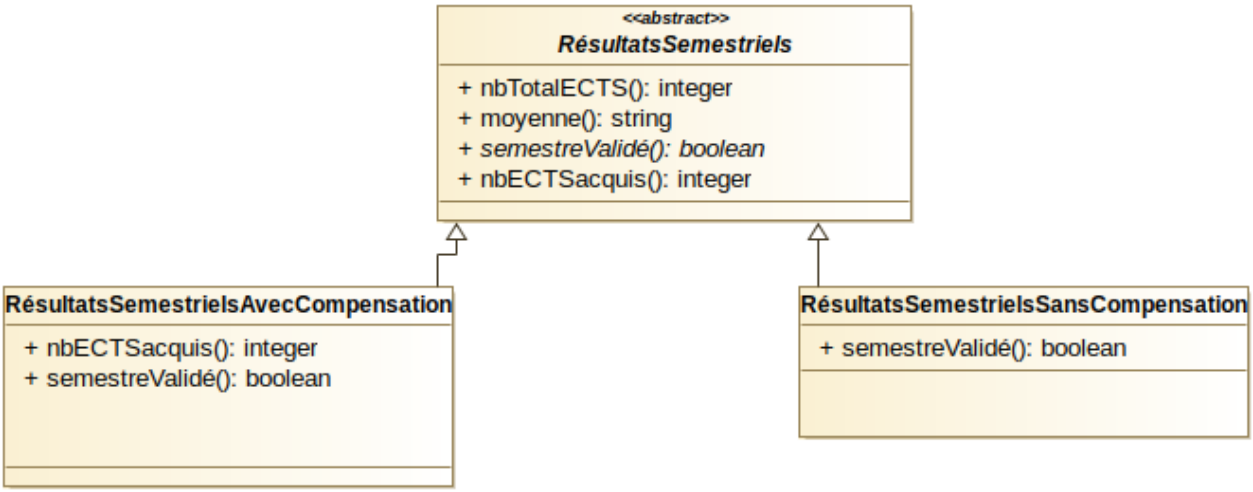
```
public abstract class ResultatsSemestriels {// A COMPLETER

    private boolean saisieTerminee=false;

    public int nbTotalECTS() { // A COMPLETER
        int total=0;



        return total;
    }

    public void addResultat(ResultatUE res) { // A COMPLETER




    }

    // ci-dessous ajouter methode pour retourner le resultat d'une UE



}
```

Question 5.

4 Correction

Question 1.

FIGURE 5 – Diagramme de classes complété



COHERENCE diag/code : 0.5 points

Listing 6 – Code Java de ResultatsSemestriels

```
public abstract class ResultatsSemestriels {
    protected ArrayList<ResultatUE> resultats =new ArrayList<ResultatUE>();
    private boolean saisieTerminee;

    public int nbTotalECTS() {
        int total=0;
        for (ResultatUE res:resultats) {
            total+=res.getNbECTS();
        }
        return total;
    }

    public boolean isSaisieTerminee() {
        return saisieTerminee;
    }

    public void setSaisieTerminee(boolean saisieTerminee) {
        this.saisieTerminee = saisieTerminee;
    }

    public void addResultat(ResultatUE res) {
        resultats.add(res);
    }

    public float moyenne() { // 1pt si presente ici et pas en dessous
        float total=0;
        for (ResultatUE res:resultats) {
            total+=res.getNbECTS()*res.getNote(); // 0.25 pour getNote()
        }
        return total/nbTotalECTS();// 0.25
    }

    public abstract boolean semestreValide(); // 1 pt pour methode ici et abstraite

    public int nbECTSAcquis() { //0.5 si concrete ici
        int total=0;
        for (ResultatUE res:resultats) {
            if (res.getNote()>=10) { // ou placer une methode UE acquise dans ResultatUE //
                0.25
                total+=res.getNbECTS();
            }
        }
        return total;
```

```
    }
}
```

Listing 7 – Code Java de ResultatsSemestrielsAvecCompensation

```
public class ResultatsSemestrielsAvecCompensation extends ResultatsSemestriels {

    public boolean semestreValide() { //0.5 si concrete ici
        return moyenne()>=10;// 0.25
    }

    public int nbECTSAcquis() { //0.5 si concrete ici
        if (semestreValide()) { //0.25
            return nbTotalECTS(); // 0.25
        }else {
            return super.nbECTSAcquis(); //0.5
        }
    }
}
```

Listing 8 – Code Java de ResultatsSemestrielsSansCompensation

```
public class ResultatsSemestrielsSansCompensation  extends ResultatsSemestriels{
    public boolean semestreValide() { //0.5 si concrete ici
        for (ResultatUE res:resultats) {
            if (res.getNote()<10) { // 0.25
                return false;
            }
        }
        return true;
    }
}
```

— sur 6,5 (le détail est ci-dessus et ci-dessous)
— les indications de points ci-dessous et ci-dessus comprennent l'UML+l'implémentation ; en vérifier la cohérence
— 0.5 cohérence modèle/code (si les 2 sont remplis)
— 1.5 moyenne : 1 si placé dans la super-classe et pas en dessous, 0.25 pour l'appel à l'accesseur, 0.25 pour l'appel à nbTotalECTS
— semestre validé : 2,5pts. 1 point si bien abstrait dans la superclasse, 0.5 par implémentation dans les sous-classes+ 0.25 pour l'appel à moyenne()+0.25 pour l'appel à getNote().
— nbECTSacquis : 2pts. 0.5 si présente concrète dans la superclasse ++0.25 pour la condition du if+ 1.5 pour méthode de la sous-classe (0.5 si présente dans la sous-classe + 0.5 point pour l'appel super.nbECTSacquis()+ 0.25 pour l'appel nbTotalECTS()) ; Si abstract dans superclasse et def ds chacune des sous classes (avec donc duplication de code) mettre 1,5/2,25

Question 2.

a-

Listing 9 – Code Java exceptions

```
    public float moyenne() throws NotesNonSaisiesException{ // 0.5 pt pour le throws

        if (saisieTerminee) { // 0,25
            float total=0;

            for (ResultatUE res:resultats) {
                total+=res.getNbECTS()*res.getNote();
            }
            return total/nbTotalECTS();
        }else {
            throw new NotesNonSaisiesException(); // 1 pt
        }
    }
____
public class NotesNonSaisiesException extends Exception { // 0.5 pts

}
```

b-

Listing 10 – Code Java exceptions

```
public class Etudiant {
    private String nom;
    private ResultatsSemestriels resultatCourant; //0.5

    public String editionResultats() { // 0.25 pas de throws
        String result;
        try { // 0.5
            result=String.valueOf(resultatCourant.moyenne()); // 0.25, accepter result=moyenne
                ()
        }catch (NotesNonSaisiesException e) { // 0.5 catch
            result="resultats␣indisponibles"; // 0.25
        }
        return result;
    }
}
```

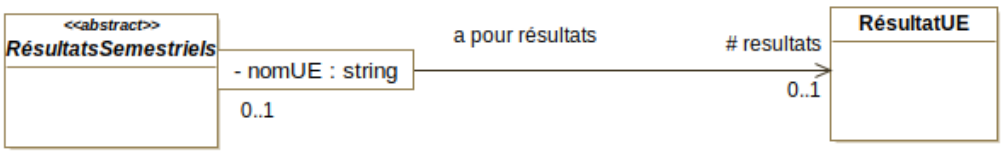
Question 3.

Listing 11 – Code Java pour diagramme d'instances

```
ResultatsSemestriels res=new ResultatsSemestrielsAvecCompensation(); // 0.25
Etudiant e=new Etudiant("Antoinette", res); // 0.5
ResultatUE res1=new ResultatUE(12, 20, "UE1"); // 0.25 pour les 2 resultats
ResultatUE res2=new ResultatUE(8, 10, "UE2");
res.addResultat(res1); // 0.5 pour les 2 ajouts
res.addResultat(res2);
res.setSaisieTerminee(true);     // 0,25
        ----------
il manque le constructeur d'etudiant␣suivant␣(ou␣autre␣constructeur␣et␣accesseurs)␣:␣//␣0.25
␣␣␣␣public␣Etudiant(String␣nom,␣ResultatsSemestriels␣resultatCourant)␣{
␣␣␣␣␣␣␣␣this.nom␣=␣nom;
␣␣␣␣␣␣␣␣this.resultatCourant␣=␣resultatCourant;
␣␣␣␣}
```

Question 4.

FIGURE 6 – Diagramme complété (1 point)

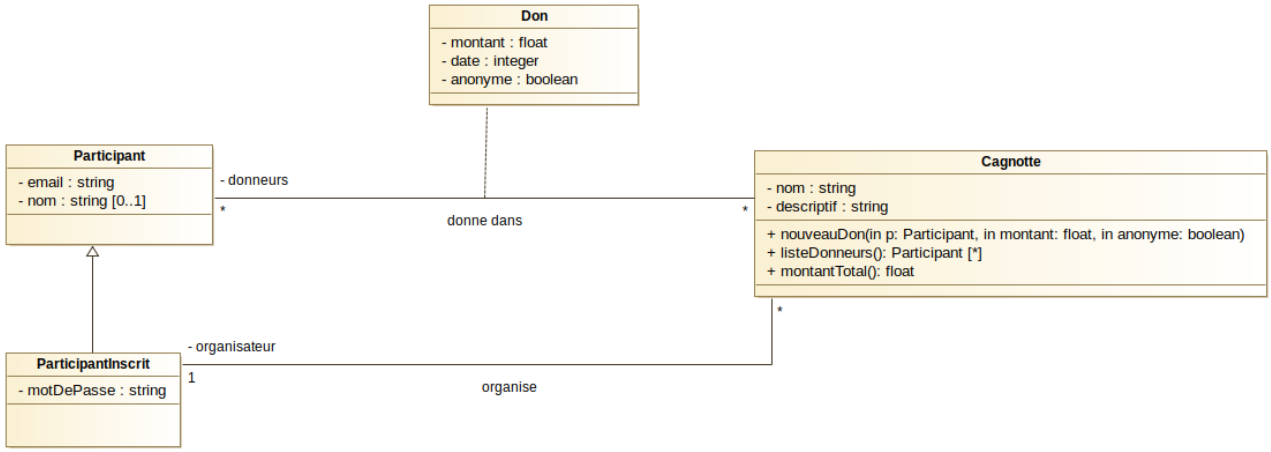


Listing 12 – Code Java de ResultatsSemestriels à compléter

```
public abstract class ResultatsSemestriels {
    protected HashMap<String,ResultatUE> resultats =new HashMap<>(); // 1 pt point
    private boolean saisieTerminee=false;

    public int nbTotalECTS() {
        int total=0;
        for (ResultatUE res:resultats.values()) { // 0.5 points
            total+=res.getNbECTS();
        }
        return total;
    }


    public void addResultat(ResultatUE res) {
        resultats.put(res.getNomUE(), res); // 0.5 pts
    }

    public ResultatUE getResultatUE(String nomUE) { // 0.5 pts (signature)
        return resultats.get(nomUE); // 0.5 pts
    }
}
```

Question 5.

FIGURE 7 – Cagnottes



— identification des classes avec dissociation des 2 types de participants (accepter 3 types de participants avec classe abstraite)– 1
— héritage entre participants – 0,5
— association organisation – 0,75
— association participation avec classe d'association – 1,25 (autre solution correcte sans classe d'association acceptée)
— méthodes (3 méthodes, avec paramètres corrects (plusieurs solutions envisageables) et type de retour correct – 1,5 (0.5 par question)
— attributs – 0,75 (tolérer quelques oublis)