NOM POUPA

Prénom Adrien

Promo L'3C 2018

Date 03/03/16

POUPA Adrien
L3PRIME - 2015

18,00

Très Bien

1/3

MATIÈRE Java 1

1/1

1) On doit mettre en oeuvre le principe d'héritage. On définit une classe "Personne" qui contient les attributs nom et prenom, "Professeur" et "Eleve" héritent de cette classe.
La classe "Personne" a des méthodes abstraites implémentées dans "Professeur" et "Eleve".

2)
```
package notes.Eleves.Professeurs;
public class Personne {

// On met la visibilité à privé
private String nom;
private String prenom;

public String getNom() {
    return nom;
}

public String getPrenom() {
    return prenom;
}

@override
abstract public toString();
}
```

Constructeur

NON

```
package notes.Eleves.Professeurs;
public class Eleve extends Personne {

    @Override
    public String toString() {
        return "( " + getPrenom() + ", " + getNom() + " )";
    }
}
```

Constructeur ?

Code dupliqué !!

```
package notes.Eleves.Professeurs;
public class Professeur extends Personne {

    @Override
    public String toString() {
        return "( " + getPrenom() + ", " + getNom() + " )";
    }
}
```

Constructeur ?

(3)

```
package notes.Eleves.Professeurs;
Public class Explication {

    private Professeur professeur;
    private Eleve eleve;
    private float note;
```

2/2

```
public void Evaluation(professeur, eleve, note) {
    this.professeur = professeur;
    this.eleve = eleve;
    this.note = note;
}

@Override
public String toString() {
    return "(" + professeur.toString() + " " + eleve.toString()
        + " " + note + ")";
}
}
```

(4) Le code suivant est à insérer dans le bloc de la classe Eleve de la question (2) :

```
private static final int = 10;
private int id;

private ArrayList<Evaluation> notes;
private HashSet<Professeur> professeurs;   PAS EN ATTRIBUT.

public void Eleve(nom, prenom, id) {
    this.nom = nom;
    this.prenom = prenom;
    this.id = id;
    notes = new ArrayList<Evaluation>();
    professeurs = new HashSet<Professeur>();
}
```

```
public void ajoutNote (Evaluation note)
        throws IllegalStateException {
    if (notes.size() == NB_EVALUATIONS) {
        throw new IllegalStateException ("no notes au maximum");
    }
    notes.add (note); //Ajout note (évaluation) dans le container
    professeurs.add (note.getProfesseur()); //Ajout professeur
}                                           //On considère que getProfesseur existe

public int getId() {
    return id;
}

public float moyenne() throws IllegalStateException {
    if (notes.size() == 0) {
        throw new IllegalStateException ("L'élève n'a pas
        de note");
    }

    Float somme = 0;

    for (int i : notes) {
        somme += i.getNote(); // Somme du container de notes
                              // On considère que getNote existe
    }

    return somme / notes.size();
}
```

2/2

2/2

NOM POUPA

Prénom Adrien

Promo L3C 2018

Date 03/03/16

2/3

Set<Professeur>

MATIÈRE Java 1

public Set getCorrecteurs() {

return professeurs; Si appel de ajout Note !!

}

1/2

//On redéfinit désormais le code toString() donné plus haut pour la classe élève (il faut prendre en compte celui-ci)

public String toString() { Surcharge toString()

return "(" + getPrenom() + ", " + getNom() + ")"

+ "id:" + id + " " + notes.size() + "notes:"

+ notes + "mogenne = " + moyenne() +

" correcteur(s) : " + professeurs.toString();

2/2

}

En haut de la classe Eleve, il faut ajouter :

import java.util.Set;

import java.util.List;

Pour que l'ajout du professeur fonctionne, il faut rajouter dans la classe Professeur :

```
@Override
public boolean equals (Object o) {
    if (this == o)
        return true;
    if (o == null)
        return false;
    if (getClass() != o.getClass())
        return false;
    Professeur prof = (Professeur) o;
    if ( prof.getPrenom().equals(getPrenom()) &&
      prof.getNom().equals(getNom()) )
        return true;
    else
        return false;
}
```

NON demandé !

(5) 2/2

```
package note.Eleve.Professeurs;
import java.util.List;
public class Promotion {
    private String nom;
    private List<Eleve>() eleves;

    public void Promotion(String nom) {
        this.nom = nom;
        eleves = new List<Eleve>();
    }

    public void setNom(String nom) {
        this.nom = nom;
    }

    public String getNom() {
        return nom;
    }

    perblic List<Eleve> get Eleves {
        return eleves;
    }
```

```
public Eleve rechercher (int id) {
    for ( Eleve e : eleves ) {
        if (e.getId() == id) {
            return e;
        }
    }
    return null;
}
}
```

11/11

⑥ Dans la classe professeur, on rajoute :

```
public Eleve rechercher (int id, Promotion promotion){
    return promotion.rechercher(id);
}
```

⑦ Dans la classe Professeur, on ajoute :

(Attention : dans l'énoncé il était indiqué que l'on pouvait choisir le contrainer de notre choix, j'ai donc pris ArrayList. Or cette question suppose désormais que le contrainer choisi était un tableau d'entier int[] id ! Il faut donc lire le code suivant en considérant que le tableau choisi était int[] id)

NOM POUPA

Prénom Adrian

Promo L'3 C 2018

Date 03/03/16

3/3

MATIÈRE Java 1

Classe Professeur (2/2)

```
public void setNote(Promotion promotion, float note,
                    int id, int i)
    throws IllegalStateException {

    //Recherche élève à partir de la méthode de Professeur
    if(rechercher(id, promotion) == null) {
        throw new IllegalStateException("Eleve non
        trouvé");
    }

Eleve eleve = rechercher(id, promotion);

Evaluation eval = new Evaluation(this, [illegible], note);
boolean modifie = False;

ArrayList<Evaluation>() nouvellesNotes = new ArrayList<Evaluation>();
int j = 0;
//On assume que getNotes() existe
ArrayList<Evaluation>() anciennesNotes = eleve.getNotes();

for(Evaluation e : anciennesNotes) {
    if(j == i) {
        nouvellesNotes.add(eval); // Ajout nouvelle note
        eleve.addCorrecteur(this); //On assume que la
        méthode existe : ajout du prof correct
        modifie = true;
    }
```

```
        else {
            nouvellesNotes.add(le);
        }
        i++; //Incrémentation compteur
    }
    //Ajout nouvelle note à la fin si nécessaire
    if (!modifie == false) {
        nouvellesNotes.add(eval);
    }
    if (nouvellesNotes.size() > 10) {
        throw new IllegalStateException("Plus de 10 notes");
    }
    eleve.setNotes(nouvellesNotes)
    //On assume que setNotes existe
}
```

(2/2)

(8) On modifie le header de la classe Élève en :

```
public class Eleve extends Personne implements
Comparable<Eleve>
```

et dans le corps de la classe, on ajoute :

```
@Override
public int compareTo(Eleve other) {
    if (moyenne() > other.moyenne())
        return 1;
    else if (moyenne() < other.moyenne())
        return -1;
    else
        return 0;
}
```

Dans la classe Promotion, on ajoute

```
public void triMoyenne() {
    Collections.sort(eleves, new TriMoyenne());
}
// TriMoyenne est définie au vers. ⇒
```

Inutile !!

Et il faut importer java.util.Collections;

```
public class TriMoyenne implements Comparator<Eleve>
{

    public int compare(Eleve e1, Eleve e2) {
        return e1.compareTo(e2);
    }

}
```