

Site :  Luminy  St-Charles  St-Jérôme  Cht-Gombert  Aix-Montperrin  Aubagne-SATIS

Sujet de :  1<sup>er</sup> semestre  2<sup>ème</sup> semestre  Session 2      Durée de l'épreuve : 2h

Examen de : L2      Nom du diplôme : Licence d'informatique

Code du module : SIN3U02      Libellé du module : Programmation 2

Calculatrices autorisées : NON      Documents autorisés : OUI

---

## 1 Correction de l'examen

Vous trouverez ci-dessous une correction possible de l'examen. C'est une des corrections possibles mais pas la seule.

## 2 Gestion de résultats d'étudiants

### 2.1 Classe Grade

```
package fr.univ_amu.grades;

import java.util.List;

public class Grade {
    private static final int MAXIMUM_GRADE = 20;
    private final double value;
    private final boolean isAbsent;

    public Grade(double value, boolean isAbsent) {
        this.value = value;
        this.isAbsent = isAbsent;
    }

    public Grade(double value) {
        this(value, false);
    }

    public Grade(){
        this(0., true);
    }

    public double getValue() {
        return value;
    }

    public boolean isAbsent() {
        return isAbsent;
    }

    @Override
    public String toString() {
        if (isAbsent)
            return "ABS";
        return value + "/" + MAXIMUM_GRADE;
    }
}
```

## 2.2 Ajout d'exception

On change le code du constructeur `public Grade(double value, boolean isAbsent)` pour le remplacer par :

```
public Grade(double value, boolean isAbsent) {
    if (value<0 || value>MAXIMUM_GRADE)
        throw new IllegalArgumentException("Grade's value (" +value+") must be in the range 0-"++
    this.value = value;
    this.isAbsent = isAbsent;
}
```

## 2.3 Classe TeachingUnitResult

```
package fr.univ_amu.grades;

public class TeachingUnitResult {
    private final String teachingUnitName;
    private final Grade grade;

    public TeachingUnitResult(String teachingUnitName, Grade grade) {
        this.teachingUnitName = teachingUnitName;
        this.grade = grade;
    }

    public Grade getGrade() {
        return grade;
    }

    @Override
    public String toString() {
        return teachingUnitName + " : " + grade;
    }
}
```

## 2.4 Classe Student

```
package fr.univ_amu.grades;

import java.util.ArrayList;
import java.util.List;

public class Student {
    private final String firstName;
    private final String lastName;
    private final List<TeachingUnitResult> results;

    public Student(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.results = new ArrayList<>();
    }

    public void addResult(String teachingUnitName, Grade grade){
        results.add(new TeachingUnitResult(teachingUnitName, grade));
    }
}
```

## 2.5 Calcul de la moyenne d'un étudiant

Ajout de `public static Grade averageGrade(List<Grade> grades)` dans la classe `Grade`

```
public static Grade averageGrade(List<Grade> grades){
    double sumOfGrades = 0.;
    for(Grade grade : grades){
        if(grade.isAbsent())
            return new Grade();
        sumOfGrades += grade.getValue();
    }
    return new Grade(sumOfGrades/grades.size());
}
```

Ajout de `public List<Grade> getGrades()` et `public Grade averageGrade()` dans la classe `Student` :

```
public List<Grade> getGrades(){
    List<Grade> grades = new ArrayList<>();
    for(TeachingUnitResult result : results)
        grades.add(result.getGrade());
    return grades;
}

public Grade averageGrade(){
    return Grade.averageGrade(getGrades());
}
```

## 2.6 Classe Cohort

```
package fr.univ_amu.grades;

import java.util.ArrayList;
import java.util.List;

public class Cohort {
    private final String name;
    private final List<Student> students;

    public Cohort(String name) {
        this.name = name;
        this.students = new ArrayList<>();
    }

    public void addStudent(Student student){
        students.add(student);
    }
}
```

## 2.7 Affichage promotion

Ajout de `public void printResults()` dans la classe `Student` :

```
public void printResults(){
    printName();
    for (TeachingUnitResult result : results)
        System.out.println(result);
    printAverageGrade();
}
```

```

public String getName() {
    return firstName + " " + lastName;
}

private void printName() {
    System.out.println(getName());
}

private void printAverageGrade() {
    System.out.println("Note moyenne : " + averageGrade());
}

```

Ajout de public void printStudentResults() dans la classe Cohort :

```

public void printStudentsResults(){
    printName();
    System.out.println();
    for(Student student : students){
        student.printResults();
        System.out.println();
    }
}

public void printName(){
    System.out.println(getName());
}

public String getName() {
    return name;
}

```

Ajout d'une classe Main pour l'affichage :

```

package fr.univ_amu.grades;

public class Main {

    public static void main(String[] args) {

        Student studentPaul = new Student("Paul", "Calcul");
        Student studentArnaud = new Student("Arnaud", "Labourel");

        Cohort l2info = new Cohort("L2 informatique");

        l2info.addStudent(studentArnaud);
        l2info.addStudent(studentPaul);

        String[] teachingUnitNames = {"Programmation 2", "Structures discrètes"};

        Grade[] gradesPaul = {new Grade(12), new Grade(10)};
        Grade[] gradesArnaud = {new Grade(), new Grade(18)};

        for(int i = 0; i<teachingUnitNames.length; i++){
            studentPaul.addResult(teachingUnitNames[i], gradesPaul[i]);
            studentArnaud.addResult(teachingUnitNames[i], gradesArnaud[i]);
        }

        l2info.printStudentsResults();
    }
}

```

```
}  
}
```

## 2.8 Compter les étudiants validant leur année

Ajout de la méthode `public int countFilteredStudents(Predicate<Student> predicate)` à la classe `Cohort` :

```
public int countFilteredStudents(Predicate<Student> predicate){  
    int filteredStudentsCount = 0;  
    for (Student student : students) {  
        if (predicate.test(student))  
            filteredStudentsCount++;  
    }  
    return filteredStudentsCount;  
}
```

Ajout d'une classe `MinimalGradeCriterion` implémentant l'interface `Predicate<Student>` :

```
package fr.univ_amu.grades;  
  
import java.util.function.Predicate;  
  
public class MinimalGradeCriterion implements Predicate<Student> {  
    private final double minimalGrade;  
  
    public MinimalGradeCriterion(double minimalGrade) {  
        this.minimalGrade = minimalGrade;  
    }  
  
    @Override  
    public boolean test(Student student) {  
        Grade averageGrade = student.averageGrade();  
        return !averageGrade.isAbsent() && averageGrade.getValue() >= minimalGrade;  
    }  
}
```

Ajout de la méthode `public int countValidatingStudents()` à la classe `Cohort` :

```
public int countValidatingStudents(){  
    return countFilteredStudents(new MinimalGradeCriterion(10));  
}
```

Modification de la méthode `printStudentResults()` de la classe `Cohort` :

```
public void printStudentsResults(){  
    printName();  
    System.out.println();  
    for(Student student : students){  
        student.printResults();  
        System.out.println();  
    }  
    printValidatingStudentsCount();  
}  
  
private void printValidatingStudentsCount(){  
    System.out.println("Nombre d'étudiants ayant validé : "+ countValidatingStudents());  
}
```