Aix*Marseille université
Initiative d'excellence
Année 2020-21

Domaine Sciences et Technologies
Licence informatique

Programmation 2 : TP 1
Code UE : SIN3U02

Gestion des notes des étudiants

Le but de ce TP est de créer des classes permettant de représenter des étudiants (classe `Student`), des notes (classe `Grade`), des résultats à une unité d'enseignement (classe `TeachingUnitResult`) et des promotions d'étudiants (classe `Cohort`).

Pour ce TP, vous ne partirez pas de zéro, nous vous fournissons une première version des classes `Student`, `Grade`, `TeachingUnitResult` et `Cohort`. Les fichiers correspondant sont disponibles sur le dépôt `git` du cours : `https://etulab.univ-amu.fr/nasr/programmation2`, dans le répertoire `tp1`.

Pour l'utilisation de `git`, reportez vous aux transparents qui se trouvent sur le site du cours.

Veillez à respecter les consignes suivantes :

— À chaque modification de programme, faites un `commit` en sélectionnant les fichiers modifiés. Chaque `commit` doit contenir un message précisant la nature des modifications effectuées.
— À chaque tâche terminée, faites un `push` de votre travail.
— Ceci est le minimum. Vous pouvez faire plus de `commit` et plus de `push`, ainsi que des `pull` pour récupérer le travail de votre binôme.
— Si vous avez un problème et souhaitez l'aide de votre instructeur en dehors des séances, un `push` lui permet de voir votre programme.

1 La classe `Grade`

Cette classe représente une note obtenue par un étudiant. Une note est une valeur décimale comprise entre 0 et 20.

Cette classe contient les éléments suivants qui sont corrects :

— `private static final int MAXIMUM_GRADE` : un attribut statique représentant la valeur de la note maximale qui est égal à 20.
— `private final double value` : la valeur de la note comprise entre 0 et `MAXIMUM_GRADE`.
— `public Grade(double value)` : constructeur évident.
— `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux notes.

Votre but est de compléter les instructions des méthodes suivantes :

— `public double getValue()` : retourne la valeur (`value`) de la note.
— `String toString()` : retourne une représentation de la note sous forme de chaîne de caractères. Pour une note ayant une valeur 12, cette méthode devra retourner la chaîne de caractères : `"12.0/20"`.
— `public static Grade averageGrade(List<Grade> grades)` : calcule et renvoie la moyenne d'une liste de notes.

Assurez-vous que les méthodes que vous avez implémenté sont correctes à l'aide de la classe `TestGrade`. Cette dernière implémente trois tests : `testToString`, `testGetValue` et `testAverageGrade` qui testent le comportement des méthodes `toString`, `getValue` et `averageGrade`.

2 La classe `TeachingUnitResult`

Cette classe représente un résultat obtenu par un étudiant, c'est-à-dire une note associée à une Unité d'Enseignement (UE).

Cette classe contient les éléments suivants qui sont corrects :

— `private final String teachingUnitName` : le nom de l'unité d'enseignement du résultat.
— `private final Grade grade` : la note du résultat.
— `public TeachingUnitResult(String teachingUnitName, Grade grade)` : constructeur évident.
— `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux résultats.

Votre but est de compléter les instructions des méthodes suivantes :

— `public Grade getGrade()` : retourne la note associée au résultat.

— `public String toString()` : renvoie le nom de l'unité d'enseignement suivi de `" : "` suivi de la représentation en chaîne de caractère de la note. Par exemple, un résultat d'une UE de Programmation 2 avec une note de 20 devra renvoyer la chaîne de caractères suivante : `"Programmation 2 : 20.0/20"`.

Inspirez vous de la classe `TestGrade` pour créer la classe `TestTeachingResults`, qui implémente des méthodes `testToString` et `testGetGrade`. Utilisez cette classe pour tester les méthodes que vous avez implémenté.

3 La classe `Student`

Cette classe représente un étudiant.

Cette classe contient les éléments suivants qui sont corrects :

— `private final String firstName` : le prénom de l'étudiant.

— `private final String lastName` : le nom de famille de l'étudiant.

— `private final List<TeachingUnitResult> results` : les résultats de l'étudiant.

— `public Student(String firstName, String lastName)` : constructeur initialisant le nom et prénom de l'étudiant avec les valeurs données et créant une liste vide pour les résultats.

— `public boolean equals(Object o)` : méthode permettant de tester l'égalité de deux étudiants.

Votre but est de compléter les instructions des méthodes suivantes :

— `public void addResult(String teachingUnitName, Grade grade)` : ajoute un nouveau résultat à partir du nom de l'UE et d'une note.

— `public List<Grade> getGrades()` : renvoie la liste des notes associées aux résultats de l'étudiant.

— `public String toString()` : renvoie le nom de l'étudiant, c'est-à-dire son prénom, suivi d'un espace, suivi de son nom.

— `public Grade averageGrade()` : renvoie la moyenne des notes associés aux résultats de l'étudiant.

— `public void printResults()` : affiche les résultats de l'étudiant en sortie standard. Un étudiant nommé Arnaud Labourel et ayant 20 en Programmation 2 et en structures discrètes devra produire l'affichage suivant (avec un saut de ligne à la fin) :

```
Arnaud Labourel
Programmation 2 : 20.0/20
Structures discrètes : 20.0/20
Note moyenne : 20.0/20
```

Inspirez vous de la classe `TestGrade` pour créer la classe `TestStudent`, qui permet de tester les méthodes de la classe `Student`. Utilisez cette classe pour tester les méthodes que vous avez implémenté.

Assurez-vous que votre classe est correcte en exécutant à nouveau les tests et en vérifiant que votre classe passe les tests

4 La classe `Cohort`

Cette classe représente une promotion d'étudiants. La classe `Cohort` contiendra les attributs, méthodes et constructeurs suivants :

Cette classe contient les éléments suivants qui sont corrects :

— `private final String name` : le nom de la promotion

— `private final List<Student> students` : les étudiants de la promotion

— `public Cohort(String name)` : constructeur à partir du nom de la promotion et initialisant à vide la liste des étudiants

Votre but est de compléter les instructions des méthodes suivantes :

— `public void addStudent(Student student)` : ajoute un étudiant à la promotion.

— `public List<Student> getStudents()` : renvoie la liste des étudiants de la promotion.

— `String toString()` : retourne une représentation de la promotion correspondant à son nom.
— `public void printStudentsResults()` : affiche les résultats de l'étudiant en sortie standard. Une promotion ayant pour nom L2 informatique et deux étudiants devra produire l'affichage suivant (avec un saut de ligne à la fin)

```
L2 informatique

Paul Calcul
Programmation 2 : 10.0/20
Structures discrètes : 20.0/20
Note moyenne : 15.0/20

Pierre Kiroul
Programmation 2 : 10.0/20
Structures discrètes : 0.0/20
Note moyenne : 5.0/20
```

Créer la classe `Cohort` avec les méthodes demandées, ainsi que la classe `TestCohort`.

5 La classe `Main`

Vous allez maintenant ajouter une classe `Main` au projet. Ajoutez dans votre classe `Main` le code d'une méthode `public static void main(String[] args)` qui :

1. crée des instances de `Student` ayant les noms et prénoms des membres du projets,
2. ajoute à ces étudiants les notes en "Programmation 2" et "Structures discrètes" que vous aimeriez avoir,
3. crée une promotion (instance de `Cohort`) nommée "L2 informatique",
4. ajoute les étudiants créés à la promotion et
5. affiche les résultats de la promotion.

6 Tâches optionnelles

Si vous avez fini les tâches précédentes, vous pouvez améliorer votre projet en rajoutant les fonctionnalités suivantes :

— Ajout d'une méthode comptant le nombre d'étudiants ayant validé leur année (moyenne supérieure ou égale à 10) dans une promotion.
— Changement de la classe `Grade` pour qu'elle permette de stocker des notes correspondant à une absence du résultat (affiché `ABS`).
— Calcul du nombre d'absents d'une promotion.
— Calcul de la note maximum et minimum (hors absence) d'une promotion.
— Création d'une classe `TeachingUnit` qui permet d'associer des crédits à une UE.
— Calcul pondéré de la moyenne de résultats en fonction du nombre de crédits des UE.
— Calcul de la moyenne (hors absence) d'une promotion.