Introduction

Bienvenue dans le deuxième laboratoire!

Comme pour le premier laboratoire, vous aurez le fichier Java à gauche et le guide à droite. Pour rappel, les **tests d'évaluation** doivent être lancés au moins une fois mais peuvent être lancés plusieurs fois.

Vous pouvez également **exécuter** votre programme depuis le guide. Le **débogueur** est aussi actif.

Commentaires de documentation

Écrivez de la documentation Javadoc pour au moins une méthode, en vous assurant de bien respecter le format et les bonnes pratiques. Utilisez `@param` et `@return` **au moins une fois** dans tout le projet. Générez et **vérifiez** votre documentation à l'aide des boutons de la barre de menu.

Vous pouvez documenter en français ou en anglais, mais ne mélangez pas les deux langues dans la documentation que vous écrivez. Dans les deux cas, surveillez également la qualité de la langue.

---

warning

Important

Assurez-vous d'avoir lancé **tous les tests d'évaluation** avant la remise du travail, car ++les résultats des tests sont utilisés pour calculer automatiquement la note++.

Votre travail sera **automatiquement remis à la date limite**. Assurez-vous d'avoir complété l'ensemble des étapes avant cette date. Vous pouvez remettre votre travail en avance en choisissant *Mark as Completed* à la fin du guide ou dans le tableau de bord.

Classe de gestion de note

La classe `Grade` permet de gérer une note, représentée par un **pourcentage** et une **lettre**. La classe stocke également d'autres informations sur le **cours**.

Dans les sections suivantes, vous allez implémenter l'ensemble de la classe `Grade` : ses attributs, ses constructeurs, etc.

warning

Important

Veillez à respecter précisément les **noms des attributs et méthodes**, sinon ceux-ci ne seront pas trouvés par les tests d'évaluation.

Attributs

Implémentez dans la classe `Grade` les attributs suivants :

- `department`, une **chaîne de caractères** contenant l'acronyme du département qui offre le cours. Par exemple : *LOG*, *GTI*, *MTR*, etc.
- `courseNum`, un **nombre entier**, de type primitif, représentant le numéro associé au cours. Par exemple : *100*, *121*, *801*, etc.
- `percent`, un **nombre réel à simple précision**, de type primitif, entre 0 et 100. Ce pourcentage correspond à la note obtenue dans le cours.
- `letter`, un seul **caractère**, de type primitif, qui correspond à la représentation alphabétique de l'attribut `percent` selon une échelle donnée. Les valeurs possibles sont *A*, *B*, *C*, *D* et *E*.

Implémentez un **accesseur** (*getter*) pour chacune des variables d'instance : `getDepartment`, `getCourseNum`, `getPercent` et `getLetter`.

info

À noter

Déterminez la visibilité afin que les attributs soient correctement **encapsulés**. Notez que la classe `Grade` devrait être **immuable**.

On laissera dans la classe le *constructeur sans paramètre* qui, sans contenu dans son implémentation, va initialiser tous les attributs à leur **valeur par défaut**, ou suivre les initialisations qui sont faites avec la déclaration des attributs.

Constructeurs

En plus du constructeur sans paramètre, on peut créer les objets de la classe `Grade` de trois autres façons différentes :
1. Avec le **département** et le **numéro du cours**
1. Avec le **département**, le **numéro du cours** et un **pourcentage**
1. Avec le **département**, le **numéro du cours** et une **lettre**

L'échelle de la relation entre les pourcentages et les lettres est la suivante :

Pourcentage	Lettre
[75, 100]	A
[66, 75 [	B
[57, 66 [	C
[50, 57 [	D
[0, 50 [	E

Notez que tous les objets créés par la classe `Grade` doivent avoir des **valeurs pour leurs quatre attributs**, selon les règles suivantes :
- La valeur par défaut pour l'attribut `percent` est *0*
- La valeur par défaut pour l'attribut `letter` est *E*
- Les attributs `percent` et `letter` doivent **toujours être cohérents**. En d'autres termes, quand on crée un objet avec un pourcentage, on doit sélectionner la lettre qui correspond à cette valeur. Similairement, quand on crée un objet avec une lettre, le pourcentage à utiliser sera la ++moyenne entre les deux valeurs de l'intervalle qui correspond++. Par exemple, pour la lettre *B*, le pourcentage est $\frac{66+75}{2} = 70.5$. On peut définir des méthodes auxiliaires privées pour faire ces calculs.

La relation entre les lettres et les pourcentages doit être ++stockée++ **seulement une fois**, peu importe le nombre d'objets créés de cette classe. Vous devez **toujours** utiliser cette relation quand un nouvel objet de la classe `Grade` est créé.

topic

Rappel

Une variable de classe est une variable qui est rattachée à la classe et non à une instance particulière. Il n'y en a **qu'une seule copie**, peu importe le nombre d'instances.

Vous pouvez, à l'aide d'une méthode `main`, effectuer des tests avec **différents cas de figure** pour corriger les éventuelles erreurs.

Méthode de classe

Pour des objectifs statistiques, la classe `Grade` doit garder une trace du **nombre d'objets créés** de cette classe et doit retourner ce nombre quand on appelle la méthode de classe `getInstanceCount`.

topic

Rappel

Une méthode de classe est une méthode qui n'est pas rattachée à une instance particulière et qui peut être appelée **directement sur la classe**.

Instanciations

Définissez pour la classe `Grade` la méthode d'instance `convertToString`, qui renvoit sous forme de chaîne de caractères les valeurs des quatre attributs d'un objet comme illustré ci-dessous.

topic

Rappel

Une méthode d'instance est une méthode qui doit s'appeler **sur un objet**. Elle a accès à l'état (i.e., les attributs) de celui-ci.

department	***courseNum***	***percent***	***letter***	**Affichage**
LOG	100	-	A	`LOG100 A 87.5`
GTI	121	68.5	-	`GTI121 B 68.5`
LOG	121	-	-	`LOG121 E 0.0`
LOG	320	-	B	`LOG320 B 70.5`
MAT	144	57	-	`MAT144 C 57.0`
MAT	210	-	E	`MAT210 E 25.0`
MAT	350	74.99	-	`MAT350 B 74.99`

info

À noter

- signifie qu'aucun paramètre n'est fourni pour cet attribut. Par exemple, le premier objet est construit seulement avec une lettre, le département et le numéro d'un cours.

Dans la méthode `main` de la classe, créez des notes avec les valeurs ci-dessus et affichez-les à l'aide de la méthode `convertToString`.

Relevé de notes

La classe `Transcript` fournit des méthodes de classe qui permettent d'afficher un **relevé de notes** avec une moyenne.

Calcul de moyenne

Implémentez, dans la classe `Transcript`, deux méthodes de classe `getReport` qui retournent, dans une chaîne de caractères, une liste de notes et qui calculent la moyenne :
- La première version reçoit un **tableau d'objets** de type `Grade`. Cette méthode doit ajouter toutes les notes dans la chaîne de caractères puis calculer et ajouter leur moyenne.
- La seconde version reçoit un **tableau d'objets** de type `Grade` ainsi que l'acronyme d'un **département**. Cette méthode doit ajouter les notes ++du département++ dans la chaîne de caractères puis calculer et ajouter leur moyenne.

topic

Rappel

La définition de plusieurs méthodes avec le même nom mais des signatures différentes s'appelle de la **surcharge**, ou *overloading* en anglais.

Des exemples d'appels de la méthode et du format attendu sont donnés ci-dessous.

Paramètres de *getReport*	Chaîne retournée
getReport(myGrades)

```
LOG100 A 87.5
GTI121 B 68.5
LOG121 E 0.0
LOG320 B 70.5
MAT144 C 57.0
MAT210 E 25.0
MAT350 B 74.99
Mean = 54.784283
```

|
| *getReport(myGrades, "LOG")* |

```
LOG100 A 87.5
LOG121 E 0.0
LOG320 B 70.5
Mean = 52.666668
```

|
| *getReport(myGrades, “MAT”)* |

```
MAT144 C 57.0
MAT210 E 25.0
MAT350 B 74.99
Mean = 52.329998
```

|

info

À noter

Faites tous vos calculs en **simple précision** (`float`).

Pour représenter un **retour à la ligne** dans une chaîne de caractères, on utilise le caractère `\n`. Il n’y a **pas de retour à la ligne** à la fin du rapport.

Vous pouvez, à l’aide d’une méthode `main`, effectuer des tests avec **différents cas de figure** pour corriger les éventuelles erreurs.

Affichages

Dans la méthode `main` de la classe, créez le tableau de notes `myGrades` avec les valeurs données en exemple dans le chapitre précédent à la page *Instanciations*.

Faites les appels suivants et affichez les valeurs retournées dans la console :

1. *getReport(myGrades)*
2. *getReport(myGrades, “LOG”)*
3. *getReport(myGrades, “MAT”)*
4. *getInstanceCount()*