Université de Bourgogne
Licence STS première année — année 2023-2024
Info 2a — programmation objet

Examen terminal du jeudi 16 mai 2024

Modalités

- ✓ Durée : 2h.
- ✓ Vous pouvez utiliser une feuille de notes manuscrites — écrites à la main non photocopiées — recto-verso de taille quelconque.
- ✓ Rendez l'annexe jointe si vous l'utilisez
- ✓ Dans ce devoir,
  - vous pourrez supposer que tous les accesseurs nécessaires sont déjà définis,
  - la classe Lire.java est directement utilisable : elle est supposée placée au bon endroit,
  - les listes d'éléments peuvent être représentées au choix par des `ArrayList` ou par des tableaux (Utilisez le même système de représentation pour tout l'examen).

I. Première étape

(voir schéma conceptuel en bas de page et le premier diagramme de classe de l'annexe)

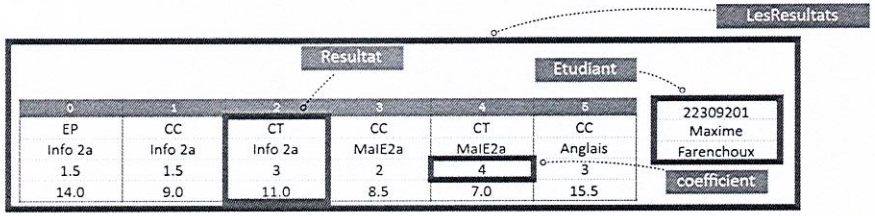
Un résultat permet de représenter le résultat de l'évaluation d'un-e étudiant-e dans une unité d'enseignement donnée et pour une épreuve donnée. Plus précisément, une instance de `Resultat` :

- ✓ a une nature : épreuve pratique ("EP"), contrôle continu ("CC") ou contrôle terminal ("CT"),
- ✓ réfère à une unité d'enseignement ("UE"), représentée par son nom ("Info 2a" par exemple),
- ✓ est affectée d'un coefficient (1.5, 3.0...),
- ✓ contient une note sur 20 (12.0 par exemple) donnée par un-e enseignant-e.

1. Complétez le premier diagramme de l'annexe avec les déclarations des attributs (variables d'instances) nécessaires pour représenter une instance de la classe `Resultat`.
2. Écrivez la méthode `getPonderee()` qui restitue la note pondérée associée à un résultat : la note sur 20 multipliée par le coefficient.

Les résultats d'un étudiant (Classe `LesResultats`) représente la collection ordonnée des résultats de cet étudiant correspondant à différentes épreuves. Pour un étudiant donné, elle...

- ✓ référence cet étudiant,
- ✓ permet de regrouper des résultats de cet étudiant (tableau ou `arrayList`).



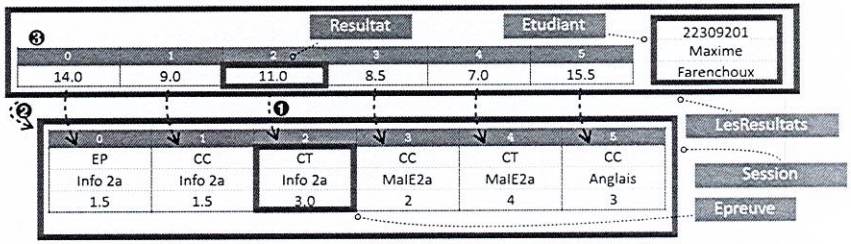
3. En supposant qu'il existe une classe `Etudiant` — dont la description n'a pas d'importance ici —, complétez le premier diagramme de l'annexe pour exprimer les propriétés d'une instance de `LesResultats`.
4. Écrivez la méthode d'instance `getCoeff()` de `LesResultats` qui calcule et restitue le coefficient global d'une liste de résultats : c'est la somme des coefficients des résultats regroupés.
5. Écrivez la méthode `getNote()` qui calcule et restitue la moyenne pondérée des notes d'une liste de résultats : c'est la somme des notes pondérées des résultats de la liste divisée par son coefficient de la question 4.

II. Deuxième étape

(voir schéma conceptuel en bas de page et le deuxième diagramme de classe en annexe)

Dans la représentation de la première étape, il y a une redondance d'informations. En effet, pour une épreuve donnée, en plus de la note, chaque résultat de chaque étudiant contient l'ensemble des informations sur l'examen : nature, unité d'enseignement, coefficient. Pour éviter cela, une classe `Epreuve` est définie, dont les instances contiennent ces informations indépendamment d'un étudiant. Chaque instance de `Resultat` ne contient alors plus qu'une note et la référence à une instance de `Epreuve` (flèche ❶ dans le schéma ci-dessous) qui indique à quelle épreuve correspond la note — ce qui permet de retrouver toutes les informations liées à la note vues dans la représentation précédente.

1. Ajoutez au deuxième diagramme de l'annexe les déclarations d'attributs et les relations entre classes qui rendent compte de ces nouvelles spécifications.
2. Créez des méthodes `toString` pour la classe `Epreuve` et pour la classe `Resultat`, qui devront restituer des résultats conformes aux exemples suivants :
   - ✓ `Epreuve` : "CT de info 2a (coefficient 3.0)"
   - ✓ `Resultat` : "11.0/20 pour l'épreuve CT de info 2a (coefficient 3.0)".
3. Écrivez la méthode d'instance `setNote()` (sans paramètre) de la classe `Resultat` qui demande une note sur vingt à l'utilisateur et l'affecte comme valeur de la note du résultat. L'affichage pour l'utilisateur sera du genre : "Entrez la note pour l'épreuve CT de info 2a (coefficient 3.0)".
4. Écrivez le constructeur de la classe `Resultat` qui a pour paramètre une référence à une épreuve et qui donne une valeur de note à l'instance en la demandant à l'utilisateur. Ce constructeur fera appel à la méthode `setNote()`.



De plus, pour une session d'examen donnée, les étudiant/e/s d'un parcours passent tous les mêmes épreuves. On définit alors une classe `Session` qui contient la liste des épreuves sans référence à la notion d'étudiant (voir schéma ci-dessus, partie du bas).

La nouvelle représentation de la classe `LesResultats` est destinée à recueillir les résultats d'un étudiant pour une session donnée. Elle doit donc contenir...

- ✓ la référence à un étudiant (Comme précédemment),

- ✓ la référence à une session (flèche ❷ dans le schéma ci-dessus),
- ✓ la collection des résultats de l'étudiant pour la session, c'est-à-dire un tableau ou une `ArrayList` d'instances de `Resultat` où il y a un résultat pour chaque épreuve de la session (collection ❸ dans le schéma).

5. Ajoutez au deuxième diagramme de l'annexe les éléments nécessaires pour rendre compte des nouvelles spécifications.
6. Pour créer et remplir la collection ❸, écrivez la méthode `setResultats()` — sans paramètre — de `LesResultats`. Cette méthode doit créer une collection de résultats en même nombre que les épreuves de la session. Son principe est de parcourir les épreuves de la session et...
   - ✓ pour chaque épreuve, elle demande à l'utilisateur une note sur vingt,
   - ✓ elle crée ensuite une instance de `Resultat` contenant cette note et référençant l'épreuve.
   - ✓ cette instance est ajoutée à la collection ❸.
7. Sans écrire de programme, indiquez ce que deviennent les méthodes `getCoeff()` et `getNote()` de la classe `LesResultats` de la première représentation ?

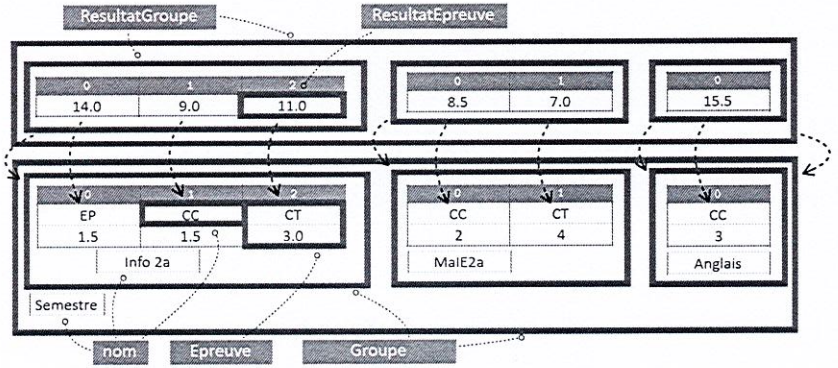
III. Troisième — et dernière — étape

(voir schéma conceptuel en bas de page et le troisième diagramme de classe en annexe)

En dernier lieu, on souhaite pouvoir modéliser la structure plus générale en bas de page (La notion d'étudiant n'est plus représentée pour simplifier).

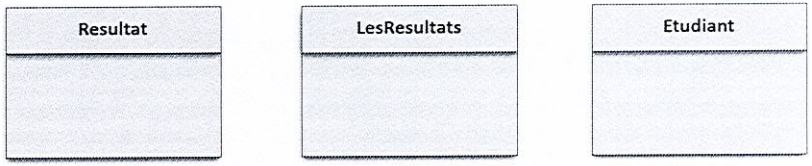
Il y a des entités qui ont un nom ("CC", "info 2a", "semestre") et un coefficient. Elles peuvent être de deux types : des épreuves ou des groupes. Les épreuves ont un coefficient donné. Les groupes sont composés d'entités, c'est-à-dire d'épreuves ou d'autres groupes. Ils ont pour coefficient la somme des coefficients de leur composants.
Il y a aussi des résultats qui référencent chacun une entité et ont une note. Il y a deux types de résultats : les résultats d'épreuves (ResultatEpreuve), qui référencent une épreuve et ont une note pour cette épreuve donnée par un-e enseignant-e. Il y a aussi les résultats de groupes (ResultatGroupe) qui sont composés de résultats et référencent un groupe. Leur note est la moyenne pondérée de leurs composants.

1. Complétez le troisième et dernier schéma de l'annexe avec les déclarations d'attributs et les relations qui correspondent à cette dernière représentation.
2. Indiquez sans programmer comment et oécrire les méthodes `getNote()` et `getCoeff()`.

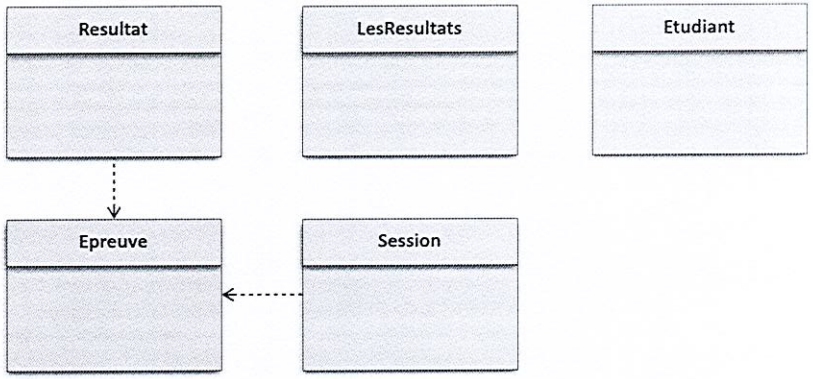


Annexe

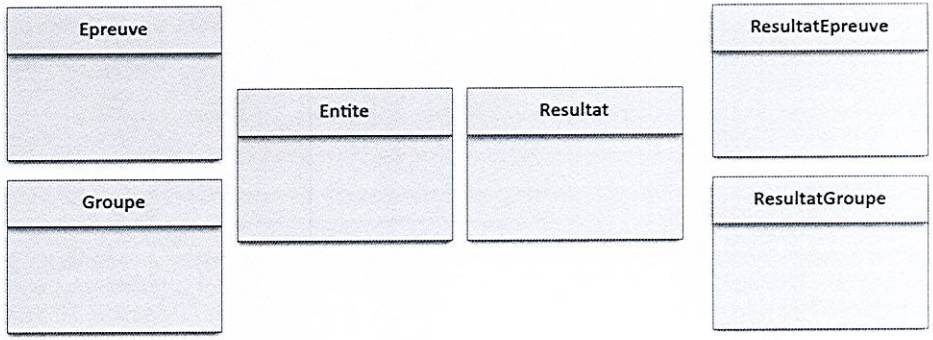
I. Première étape



II. Deuxième étape



III. Troisième étape



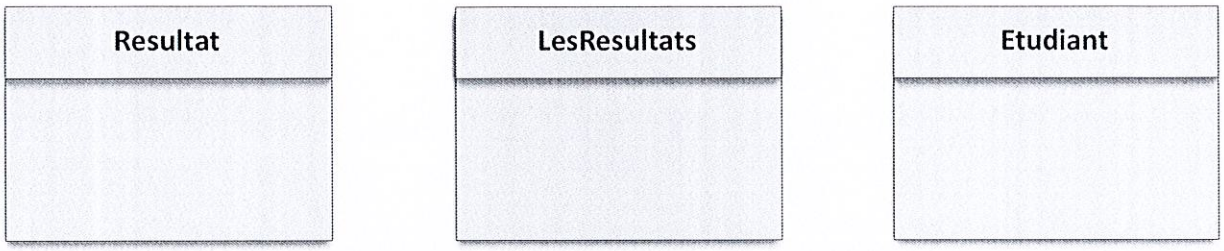
Légende

Relation d'utilisation	----------→
Relation sorte-de (généralisation-spécialisation)	——→

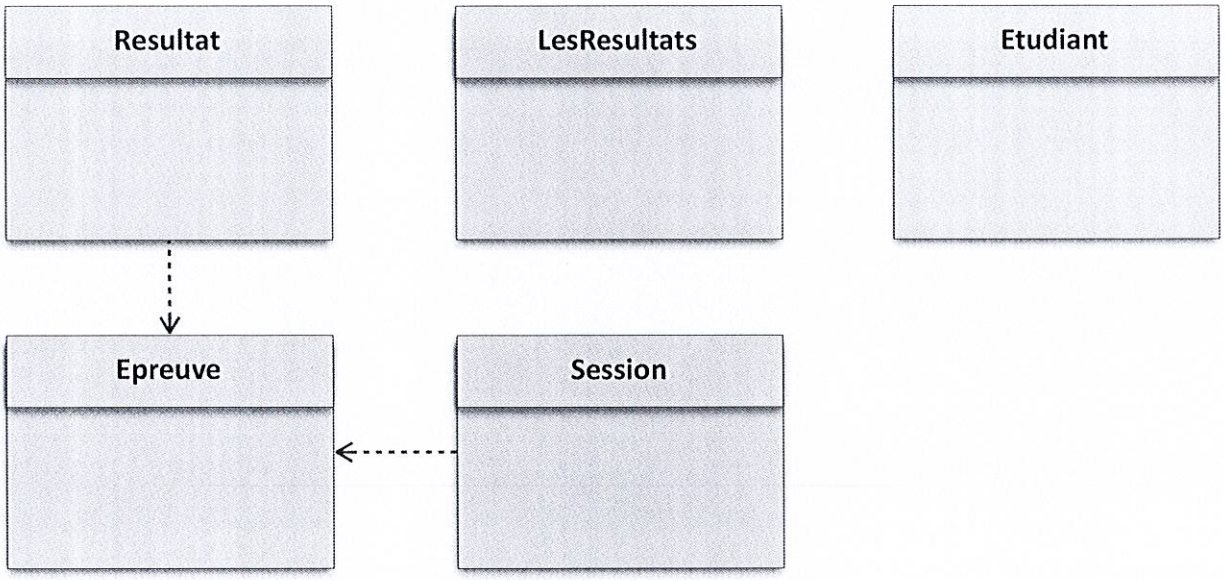
N° d'anonymat :

Annexe

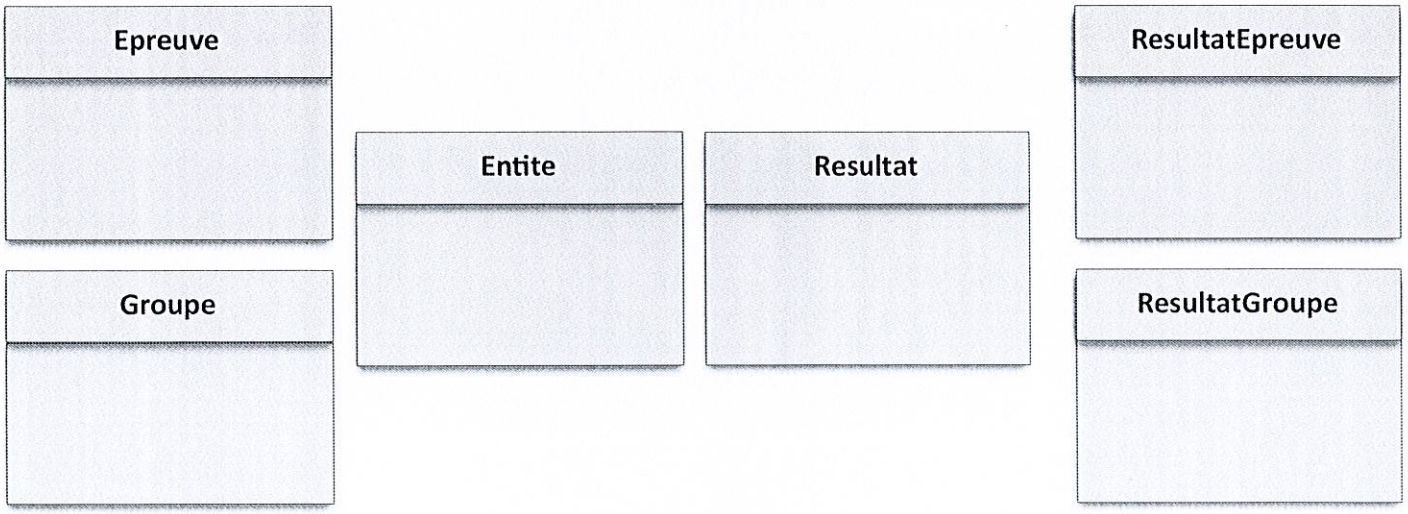
I. Première étape



II. Deuxième étape



III. Troisième étape



Légende

Relation d'utilisation	---------->
Relation sorte-de (généralisation-spécialisation)	———

N° d'anonymat :