Remarques préliminaires :

Cet énoncé est par nature incomplet, car il invite à prendre conscience de l'importance de la documentation de l'API Java pour résoudre les problèmes posés. Vous devez remettre individuellement sous forme d'une archive ZIP les codes sources commentés de l'application avant le dimanche 26/05/2024.

Le Professeur Tournesol souhaite conserver une trace des évaluations de ses élèves. Chaque élève est représenté par un nom et un prénom. Une évaluation est caractérisée par un coefficient (un entier entre 0 et 10) et une note (un réel entre 0 et 20). Les différentes évaluations d'un même élève doivent être utilisées pour calculer sa moyenne.

Une grande partie de l'application est fournie, et vous devez compléter une partie du modèle.

Aucune classe ne doit être ajoutée.

A Organisation et compréhension du code fournie

1. L'application fournie suit l'architecture **MVC**. Commencez par organiser le code en packages.
2. Parcourez le code et essayez de le comprendre. Exécutez-le.

B Lire et écrire des fichiers texte

1. Procédez à une lecture rapide de la Javadoc des classes `FileWriter`, `BufferedWriter` et `PrintWriter` du package `java.io`. Concentrez-vous sur la documentation fournie en préambule de la classe, en survolant la liste des attributs, des constructeurs et des méthodes. Quelles sont les différences entre ces classes ? Répétez la même analyse avec les classes `BufferedReader` et `FileReader`.
2. Complétez les méthodes d'écriture/lecture dans un fichier texte de la classe `StudentFileAccess`. Ces méthodes pourront prendre en paramètre le nom du fichier de sauvegarde à lire ou à écrire. Procédez à l'écriture et à la lecture des données depuis l'application via les boutons dédiés à cet effet.
3. Modifiez l'application pour qu'elle ne lise/écrive pas la moyenne, mais pour la recalculer au moment de la lecture du fichier.

C Lire et écrire des fichiers binaires

1. Étudiez la Javadoc des classes `DataInputStream` et `DataOutputStream`. À quoi servent-elles ? Complétez les méthodes `writeToBinary` et `freadFromBinaryFile` pour qu'elles utilisent ces classes. D'autres classes sont-elles nécessaires ? Pourquoi ? Corrigez l'application pour qu'elle puisse fonctionner.
2. Comment procéder pour sauvegarder la moyenne dans le fichier binaire ? Apportez les modifications nécessaires et vérifiez le bon comportement de l'application.
3. Comparez les tailles des fichiers de sauvegarde en mode texte et en mode binaire. Quelles conclusions peut-on tirer ?
4. Supprimez (provisoirement) l'appel à la méthode `close()` dans les programmes de lecture et écriture de fichiers texte et binaire. Que peut-on observer quant au fonctionnement des programmes, et au contenu des fichiers ?

D Lire et écrire des objets

1. En Java, il est possible de lire et d'écrire des objets à travers l'interface `java.io.Serializable`. Étudiez la Javadoc de l'interface `Serializable`. Que doit-on faire pour implémenter cette interface ? Faites le nécessaire pour les classes concernées dans l'application.
2. La lecture et l'écriture des objets s'effectuent respectivement avec les classes `ObjectInputStream` et `ObjectOutputStream`. Quelles sont les méthodes offertes par ces classes ? Complétez l'application pour lire et écrire les données avec ces flux d'objets.
3. Les deux classes précédentes permettent de redéfinir le mécanisme de lecture/écriture d'un objet. Quelles sont alors les méthodes à utiliser ? Faites évoluer l'application pour ne pas sauvegarder la moyenne mais la recalculer au moment de la lecture du fichier.