Remarques préliminaires :
Cet énoncé est par nature incomplet, il invite à prendre conscience de l'importance de la documentation de l'API Java pour résoudre les problèmes posés. Remettre individuellement sous forme d'une archive ZIP les codes sources commentés de l'application avant la fin de la séance.

Le Professeur Tournesol souhaite conserver une trace des évaluations de ses élèves. Un élève est représenté par un nom et un prénom. Une évaluation correspond à un coefficient (entier entre 0 et 10) et une note (entier entre 0 et 20). Les différentes évaluations d'un même élève donnent lieu au calcul d'une moyenne.

A. Lire et écrire des fichiers texte

1. Écrire la classe `Evaluation` et la compiler. Écrire la classe `Eleve` et la compiler. Écrire un programme de test et l'exécuter.

2. Procéder à une lecture rapide de la javadoc des classes `FileWriter`, `BufferedWriter`, `PrintWriter` du package java.io (en se focalisant sur la documentation fournie en préambule de la classe, et en survolant la liste des attributs, constructeurs, et méthodes). Quelles sont les différences entre ces classes ? Procéder à la même analyse avec les classes `BufferedReader`, `FileReader`.

3. Ajouter des méthodes d'écriture / lecture dans un fichier texte aux classes `Eleve` et `Evaluation`. Ces méthodes pourront prendre en paramètre le nom du fichier de sauvegarde à lire ou écrire. Procéder à l'écriture et la lecture des données depuis votre programme de test.

4. Modifier l'application pour ne pas lire / écrire la moyenne, mais pour la recalculer au moment de la lecture du fichier.

B. Lire et écrire des fichiers binaires

1. Étudier la javadoc des classes `DataInputStream` et `DataOutputStream`. A quoi servent-elles ? Modifier votre application pour que celle-ci utilise ces classes. D'autres classes sont-elles nécessaires ? Pourquoi ? Corriger l'application pour qu'elle puisse fonctionner.

2. Comment procéder pour sauvegarder la moyenne dans le fichier binaire ? Apporter les modifications nécessaires et vérifier le bon comportement de l'application.

3. Comparer les tailles des fichiers de sauvegarde en mode texte et en mode binaire. Quelles conclusions peut-on tirer ?

4. Supprimer (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 programme, et au contenu des fichiers ?

C. Lire et écrire des objets

1. En java, il est possible de lire et d'écrire des objets au travers de l'interface `java.io.Serializable`. Étudier la javadoc de l'interface `Serializable`. Que doit-on faire pour implémenter cette interface ? Faire 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 ? Faire évoluer 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 ? Faire évoluer l'application pour ne pas sauvegarder la moyenne mais la recalculer au moment de la lecture du fichier.