Environnement de développement

Licence 3 Informatique

Durée : 3 heures. Tous les documents sont permis. Chaque problème est noté avec 8 points. Tout code écrit doit être commenté.

Rendu : En plus de votre copie papier, vous devez envoyer par couriel à l'adresse `sighirea@liafa.jussieu.fr` une archive compressée contenant les sources demandées à chaque question. Ces sources seront organisées en répertoires, un par projet. Pour construire une archive compressée à partir d'une liste de répertoires, veuillez utiliser `tar` et `gzip` comme suit à l'extérieur des répertoires :

```
> tar cvf login.tar GestionNotes/ EditeurCSV/
> gzip login.tar
```

1 Utilisation de JDT

Dans ce problème, il s'agit d'utiliser JDT pour développer une application simple.

1. Créer un nouveau projet Java appelé `GestionNotes`. Ce projet doit contenir un paquet `edi.notes`.
2. Dans le paquet `edi.notes` créer une classe `Etudiant` ayant quatre champs (privés) : le numéro de carte d'étudiant (`String` unique par étudiant), son nom (`String`), son prénom (`String`) et sa note (un entier entre 0–20 si la note est connue ou -1 sinon). Définir le constructeur par défaut (champ note inconnu) et le constructeur avec les quatre champs (champ note vérifié valide).
3. Dans la classe `Etudiant`, pour chaque champ de cette classe, définir les accesseurs et des modificateurs. La modification d'une note peut se faire qu'en faveur de l'étudiant.

4. Dans le paquet `edi.notes`, créer une classe `Notes` qui gère une collection d'étudiants représentés pas la classe `Etudiant`. Pour cette collection, les opérations élémentaire seront l'ajout d'un étudiant, la modification de sa note et le tri selon leur note.

   L'ajout et la modification de note doivent s'assurer qu'un seul étudiant existe avec le même numéro d'étudiant et que la note n'est pas modifiée en défaveur de l'étudiant.

   Le tri selon les notes doit renvoyer une collection (pas nécessairement du même type) contenant le même ensemble d'étudiants mais trié en ordre croissante des notes, les notes inconnues en tête.

Pour chaque question, indiquer rapidement (3-4 lignes) dans votre copie comment vous avez utilisé JDT pour écrire votre code.

2 Test unitaire avec JUnit

Dans ce problème, il s'agit de tester la classe `Etudiant` écrite au problème précédent.

1. Dans le projet `Gestion Notes`, créer un paquet `edi.notes.test` et ajouter la librairie JUnit.
2. Créer dans le paquet `edi.notes.test` une classe de test, appelée `EtudiantTest`, pour la classe `Etudiant`.
3. Compléter cette classe pour tester la méthode qui permet de modifier une note. Pour cela, donner dans le commentaire attaché à la méthode une spécification complète de son comportement. Ecrire ensuite le code de la méthode de test afin de tester la conformité de la méthode avec cette spécification.
4. Utiliser Junit pour lancer le test. Quel est le résultat de vos tests ? Indiquer la couverture de instructions obtenue avec votre test en utilisant Coverlipse.

Pour chaque question, indiquer rapidement (3-4 lignes) dans votre copie comment vous avez utilisé JDT.

3 Editeur multi-pages pour les fichiers de notes

Pour l'entrée des notes obtenues par les étudiants, on dispose de fichiers en format CSV. Le format CSV est un format textuel pour la représentations de listes de données structurées. Chaque élément de la liste est représenté

sur une ligne. Les champs d'un élément sont séparés par un point-virgule. Le fichier de notes comporte des éléments ayant quatre champs (dans cet ordre) : le numéro de la carte d'étudiant, le nom, le prénom et la note. Le champ note peut être vide, c'est-à-dire que le séparateur de ligne (`\n`) arrive immédiatement après le caractère ';'.

En utilisant l'assistant de *plug-in*, écrire un éditeur à deux pages tel que :

- La première page permet d'éditer le fichier CSV pour introduire les notes ; le format CSV est visible, les ';' séparent les champs d'un élément.
- La deuxième page, non éditable, affiche la même liste mais ordonnée en ordre croissant des notes (les notes vides au début) et avec les ';' remplacés par deux caractères de tabulation. En plus, à la fin de cette page est affichée la moyenne des notes non-vides introduites.

Par exemple, si le fichier CSV à éditer est :

```
20606185;ABADA ATEBA;CHRISTIAN BRICE;17
20505572;AMZOUG;SAID;16
20400111;BASSO;OLIVIER;
```

la deuxième page affichera :

```
20400111 BASSO OLIVIER
20505572 AMZOUG SAID 16
20606185 ABADA ATEBA CHRISTIAN BRICE 17
Moyenne 16,5
```

Un exemple de fichier CSV plus significatif se trouve à `/ens/sighirea/EDI/exam07/EDI2.csv`.

Dans votre copie papier, vous devez répondre aux questions suivantes (en max. 3-4 lignes par question) :

1. Quel assistant vous avez utilisé ?
2. Comment indiquer que le *plug-in* considère des fichiers avec l'extension CSV ? Dans quel fichier du projet de *plug-in* se retrouve cette information ?
3. Comment avez vous lu le fichier CSV et dans quelle structure de données avez vous stocké ses éléments ?
4. Quel est l'algorithme que vous avez utilisé/implémenté pour trier les données selon les notes ?