La manipulation des chaînes de caractères, la lecture et écriture de fichiers

Objectif(s)

- ✯ Manipuler les chaînes de caractères et des `StringBuilders`
- ✯ Lire et écrire des fichiers de texte et des fichiers sérialisés

String *vs* StringBuilder

Exercice 1 – Une question d'efficacité

Sur eCampus, vous trouvez une classe `Etudiant`. Dans une nouvelle classe `StringVsBuilder` (disponible sur eCampus et à compléter) remplissez un tableau de 1000 `Etudiant` avec des instances ayant des valeurs aléatoires :

— chaque prénom et chaque nom contient entre 4 et 9 lettres, dont la première est en majuscule.

Utilisez la méthode `nextInt(int)` d'une instance de la classe `Random`.

Une manière de générer une lettre majuscule est d'exploiter le fait que la valeur ASCII de 'A' est 65, qu'il y a 26 lettres dans l'alphabète anglais et qu'un `char` est juste un `int` transtypé :

```
char lettre = (char)(65+r.nextInt(26));
```

Les lettres minuscules se génèrent de la même manière, mais avec la valeur ASCII de 97 pour 'a' 97.

— pour le genre, utilisez la méthode `nextBoolean()`. Si la valeur est `true`, on génère une étudiante, sinon un étudiant.

Afin de construire le prénom et le nom, vous utilisez

1. une chaîne de caractères, en utilisant l'opérateur $+=$.
2. un `StringBuilder`, en utilisant la méthode `append(char)`.

On va mésurer le temps d'exécution des deux options en utilisant la méthode `long currentTimeMillis()` de la classe System. Vous stockez le temps avant et après la boucle dans des variables, et affichez la différence. Un example est donné dans la classe `StringVsBuilder`.

Exercice 2 – Écrire et lire un fichier de texte

Question 1

Implémentez les classes `EmployeHoraire` et `Commercial` du TD 4. La classe abstraite `Employe` est disponible sur eCampus.

Le premier mot de la chaîne de caractères retournée par la méthode `toString()` doit être le nom de la classe de l'instance, donc soit "EmployeHoraire", soit "Commercial". Les valeurs des différents attributs sont séparées par des virgules.

Question 2

Dans une nouvelle classe, créez au moins quatre employé.es (un mélange d'`EmployeHoraire` et de `Commercial`).

Sauvegardez les informations de ces employé.es dans un fichier de texte, ainsi que son salaire, avec un employé.e par ligne du fichier :

— un objet fichier peut être créé en utilisant la méthode de classe `of()` de la classe `Path` (comme montré pendant le CM)

— une instance de la classe `BufferedWriter` peut être créée en utilisant la méthode `newBufferedWriter()` de la classe `Files` (comme montré pendant le CM). Le `charset` n'est pas obligatoire, ainsi que les `OpenOptions`, parce que vous allez créer un nouveau fichier.

N'oubliez pas de fermer le `BufferedWriter` !

Le nom du ficher **ne doit pas** être écrit en dur dans le code, mais dois être récupéré à partir du premier argument de la méthode `main` :

— soit vous définissez une "Run configuration" : clic-droit sur la classe que vous voulez exécuter, "Run As" → "Run Configurations…" → onglet "Arguments"

— soit vous exécutez la classe à partir de la ligne de commande

Le compilateur va vous demander de gérer une `IOException` — pour l'instant, ajouter simplement un `throws` à la méthode `main(String [])`, on verra les exceptions plus tard en R2.03.

Regardez les contenus du fichier.

Question 3

Lisez les lignes du fichier que vous venez de créer (en utilisant un `BufferedReader`) et affichez chaque ligne, précédée du numéro de la ligne.

Question 4

Créez deux employé.es de plus, rouvrez le fichier dans lequel vous avez écrit, **sans l'écraser** (pensez à l'énumération `StandardOpenOption` — `https://docs.oracle.com/javase/7/docs/api/java/nio/file/StandardOpenOption.html`) et écrivez les deux nouveaux employé.es dedans, précédés de la ligne "Mise-à-jour :".

Regardez les contenus du fichier et vérifiez que les informations écrites avant ne sont pas disparues.

Question 5

Lisez les contenus du fichier et reconstituez les instances.

Le premier mot de chaque ligne — soit "EmployeHoraire", soit "Commercial" — vous aide à identifier la classe de l'instance et vous pouvez diviser chaque ligne via les virgules.

Afin de traduire une chaîne de caractères en un autre type, il faut utiliser les méthodes `parseXXX(String)` des classes pertinents, p. ex. `parseDouble(String)` de la classe `Double`.

Attention : il y a une ligne qui ne contient pas d'employé.e.

Affichez les employé.es et vérifiez que le salaire stocké est bien le salaire dérivé à partir des informations de l'employé.e.

Exercice 3 – La sérialisation

Comme vous avez vu, lire des représentations textuelles des instances de différentes classes peut être difficile (en plus, stocker des fichiers textuels est tellement inefficace au niveau d'espace).

On va donc exploiter le mécanisme de la sérialisation.

Question 1

Faites en sorte que les deux classes `EmployeHoraire` et `Commercial` implémentent l'interface `Serializable`.

Question 2

Écrivez les six employé.es créés auparavant dans un fichier binaire en utilisant la méthode `writeObject(Object)`.

Le nom du fichier à créer est stocké dans le deuxième argument de la méthode `main(String [])`.

N'oubliez pas de fermer le fermer ni le flux, ni le fichier.

Ouvrez le fichier et regardez ses continus. En plus, utilisez `ls -lh` (ou l'équivalent sur Windows) afin de comparer la taille des deux fichiers.

Question 3

Rouvrez le fichier dans lequel vous venez d'écrire et lisez les instances stockées dedans. Affichez-les.

Question 4

Vous avez sans doute rencontrez quelque chose comme "Exception in thread "main" java.io.InvalidClassException : employees.EmployeHoraire ; no valid constructor" → parce que nous utilisons le constructeur de la classe `Employe`, elle aussi doit implémenter `Serializable`. Réctifiez le problème.

Question 5

Sans utilisation de l'`java.io.EOFException`, il n'est pas facile de gérer la lecture d'un nombre inconnu d'objets.

Par contre, les tableaux sont aussi sérialisable !

Si vous ne l'avez pas encore fait, stockez vos employé.es dans un tableau est utilisez `writeObject()` et `readObject()` afin d'écrire/de lire le tableau.