Fiche 1 : Architecture Logicielle

BUT 2 Info : R4.01

kamel.bouchefra@univ-paris13.fr

2024 - 2025

Nom - Prénom(s) :

Règles du TP

- Les travaux sont à réaliser de manière autonome, **individuelle**.
- Tout support, recherches internet **encouragés**.

Cet énoncé aborde des mécanismes Java qui seront mis en œuvre dans ce module. Certains sont des rappels. Au programme il y a :

- l'implémentation d'une interface ;
- la manipulation de chaînes de caractères ;
- le développement d'une classe pour l'accès en écriture / lecture d'un fichier de données ;
- le développement d'une classe permettant le hachage et la structuration de données ;
- La gestion des écritures dans un fichier, de sorte à n'ajouter que des données nouvelles.

1 Exercice 1

On donne l'interface suivante :

```java
public interface InterfaceData
{
  public void setData(String item);
  public String getData();
  public void parse();
  public void clean();
  public void afficher();
}
```

Créez une classe Java, nommée « Parse.java », implémentant l'interface, elle contiendra aussi la méthode principale.

La classe « Parse.java » comportera ainsi :

- Deux attributs : (1) Le premier de type chaînes de caractères. (2) Le second, un tableau de chaînes de caractères.
- La méthode principale dans laquelle :
  1. On définit la chaîne suivante :

```
String s = " abcd " + " , " + " efgh ";
```

  2. On appelle les traitements spécifiés ci-dessous, ainsi que la méthode d'affichage, après chaque traitement.
- Une implémentation des méthodes de l'interface :
  1. Les deux traitements suivants :
     a) *parse()* : Pour le parse de la chaîne de caractères.
     b) *clean()* : Pour supprimer l'espace autour des sous-chaînes produites.
  2. L'affichage des sous-chaînes produites.

2 Développement de classes : Gestion de données

On donne le fichier de données : « donnees.txt ». Pour la classe à développer, vous importerez « **java.io.*** ».

2.1 Lecture et écriture fichier

En vous basant sur les éléments du cours :

- Créez la classe « Fichier.java » : **Sans attributs**.
  - On commence par développer la méthode « lire() » :
    - Cette méthode ne prend aucun argument, et ne retourne aucun résultat.
    - Elle permet d'ouvrir le fichier de données et de lire chaque ligne du fichier de données.
    - Affichez aussi chaque ligne lue.
- Pour tester la classe précédente, créez la classe « FichierMain.java », contenant la méthode principale :
  - Écrire le code permettant d'utiliser la méthode « lire » de la classe « Fichier »
  - Exécutez le code, en corrigeant les éventuelles erreurs, jusqu'à son bon fonctionnement.
- Dans la classe « Fichier.java », ajoutez la méthode « ecrire() » :
  - La méthode ne prend aucun argument, et ne retourne aucun résultat.
  - Elle doit permettre d'écrire la donnée suivante :

```
String donnee = "boule, bill\n";
```

- Dans la classe « FichierMain.java », ajoutez le code permettant d'utiliser la méthode « ecrire ».
  - Exécutez le code, en corrigeant les éventuelles erreurs, jusqu'à son bon fonctionnement.
  - Vérifiez notamment que la donnée a été ajoutée.
- Ajoutez dans la méthode « ecrire » les instructions permettant l'ajout de ces données :

```
String data[] = {
                  "boule, bill\n",
                  "donald, picsou\n",
                  "homere, bugs bunny"
                };
```

  - Exécutez le code, en corrigeant les éventuelles erreurs, jusqu'à son bon fonctionnement.
  - Vérifiez que les nouvelles données ont été ajoutées. On peut par exemple, appeler à la suite la méthode « lire ».

- Ajoutez à la classe « Fichier.java » la méthode « lireFichier » :
  - La méthode ne prend aucun argument, et **retourne une chaîne de caractères**.
  - Cette méthode doit permettre de lire tout le contenu du fichier de données.
  - Elle retourne ce contenu dans une chaîne de caractères, et elle n'affiche rien.
- Dans la classe « FichierMain.java », ajoutez le code permettant d'utiliser la méthode « lireFichier ».
  - Exécutez le code, en corrigeant les éventuelles erreurs, jusqu'à son bon fonctionnement.
  - Ajoutez l'affichage du contenu retournée par « lireFichier ».
  - Vérifiez l'affichage obtenu.

2.2 Hachage

- Créez la classe « Hachage.java », elle contient :
  - Les imports suivants : « java.util.HashMap », « java.util.Set ».
  - L'attribut « contenu », de type chaîne de caractères.
  - L'attribut « fichier », de type « Fichier ».
  - L'attribut « acolyte », de type « HashMap ».
  - Un constructeur initialisant les attributs :
    - la chaîne « contenu » doit être initialisée avec le contenu du fichier de donnée de la section précédente.
    - On donne le code pour acolyte :

```
acolyte = new HashMap<String, String>();
```

- Ajoutez la méthode « void personnages() », elle réalise les traitements suivants :
  - Un « split » de « contenu » dans un tableau « lignes » : On obtient ainsi chaque ligne du fichier.

- On traite ensuite chaque ligne :
  - On extrait (avec un « split »), les noms des deux acolytes donnés par chaque ligne du fichier.
  - **Note** : Veillez à supprime les espaces entourant ces noms.
  - Enregistrez ces deux noms dans la « HashMap » (dans « acolyte ») : **Le premier nom sert concrètement de clé**.
- Ajoutez la méthode « lireAcolytes() », elle doit permettre d'afficher la clé et la valeur de chaque contenu de la « HashMap ». Concrètement, les noms des deux acolytes.
- Dans la classe « FichierMain.java », ajoutez le code permettant d'utiliser les méthodes « personnages » et « lireAcolytes ».

2.3 Ajouts de données dans le fichier

- Ajoutez la méthode « boolean ajoutAcolyte(String n) », dans la classe « Hachage.java ».
  - La méthode sera appelé pour vérifier si on peut ajouter des données dans un fichier ou pas.
  - On peut reprendre une bonne partie de la méthode « lireAcolytes() ».
  - Le paramètre « n » comporte deux Strings.
  - La méthode « ajoutAcolyte » doit permettre de comparer le paramètre « n » avec chaque contenu de la « HashMap » :
    - Si les noms des deux acolytes contenus dans « n » sont présents dans la « HashMap », la méthode doit retourner un boolean (par exemple, false), qui empêchera l'ajout de « n » dans un fichier.
    - Sinon, elle doit retourner l'autre valeur booléene (par exemple, true).
- Ajoutez la méthode « void sauvegarde() », dans la classe « Fichier.java ».
  - On peut reprendre une bonne partie de la méthode « ecrire() ».
  - La méthode « sauvegarde » doit permettre de vérifier si une nouvelle donnée est présente ou non dans le fichier avant de l'y ajouter :
    - La méthode « sauvegarde() » appelle les méthodes « personnages » et « ajoutAcolyte » de la classe « Hachage ». On y créera donc une instance de cette classe.
    - « ajoutAcolyte » est appelée avant tout ajout.
  - Dans la classe « FichierMain.java », ajoutez le code permettant d'utiliser la méthode « sauvegarde ». Si une donnée est déjà présente dans le fichier, on ne doit pas pouvoir l'ajouter.