Documentation projet Algorithmique et Programmation

Julien LEMAIRE - Pierre-Emmanuel NOVAC

23 mai 2016

1 Présentation

Le programme se présente sous la forme d'une interface graphique séparée en deux zones : à gauche le choix des critères à appliquer, à droite le résultat. Il permet de filtrer et trier les données d'un fichier lu au format CSV tant que celui ci utilise ";" comme délimiteur et que sa première ligne contient le nom des colonnes.
Le programme se lance avec la commande suivante :

```
./run.sh [fichier.csv]
```

Si aucun nom de fichier n'est indiqué, fr-esr-insertion_professionnelle-master.csv est utilisé par défaut. Il est téléchargeable à l'adresse https ://data.enseignementsup-recherche.gouv.fr/explore/dataset/fr-esr-insertion_professionnelle-master/download ?format=csv.
Si le fichier utilisé n'est pas fr-esr-insertion_professionnelle-master.csv, les critères ajoutés et les colonnes sélectionnées par défaut ne correspondront pas, il faudra les définir manuellement.
Le menu **Columns** permet de sélectionner les colonnes à afficher dans le résultat.

1.1 Critères

Les filtres peuvent être ajoutés à la volée en cliquant sur le bouton **Add**. La liste déroulante de gauche permet de sélectionner une colonne sur laquelle appliquer le filtre, la case à cocher permet de l'activer ou non, et la liste déroulante de droite permet de choisir le contenu de cette colonne de façon à extraire les lignes correspondantes. Dans cette version, la seule possibilité de comparaison repose sur l'égalité des chaînes de caractères.

1.2 Tri

Le tri se fait en fonction d'une comparaison classique de chaîne de caractères (ou de nombre à virgule flottante le cas échéant) sur une unique colonne. On peut sélectionner l'ordre croissant ou décroissant.

1.3 Limite de résultat

Une zone de texte qui n'accepte que des nombres entiers permet de spécifier une limite au nombre de résultat. Elle est par défaut de 10.

2 Choix de développement

Langue : Au départ nous avions des noms de variables et de mé thodes en français et d'autres en anglais. Dans un soucis d'homogénéité, nous avons fait le choix de n'utiliser que des noms anglais, les commentaires et le texte de l'interface sont donc eux aussi rédigés en anglais.

Javadoc : L'intégralité des classes et des méthodes possèdent des commentaires au format Javadoc. Un répertoire *javadoc/* est donc mis à disposition et contient la Javadoc générée à partir de l'ensemble du code et propose alors une vue globale du code du projet. En revanche certaines méthodes complexes contiennent des commentaires plus spécifiques pour certaines instructions et n'apparaissent donc pas dans la Javadoc.

Java 8 : Le coeur du programme repose sur les nouveautés de Java 8 comme les *Stream*, les références vers des méthodes, les expressions lambda. Toutes les fonctions du programme auraient pu être écrites sans utiliser ces fonctionnalités du langage, mais cela simplifie la programmation et permet de produire un code plus modulaire, qui pourra s'adapter à différentes situations. Il en résulte un code qui ne repose pas sur de l'algorithmique classique et fait appel a de nombreuses fonctionnalités avancées de Java. Les performances peuvent donc être discutables et plus difficilement évaluables. Les tests avec fr-esr-insertion_professionnelle-master.csv ne révèlent pas de soucis de ce côté.

3 Structuration du code

Nous avons essayé au maximum de procéder à une séparation cohérente du code. Dans un premier temps, le code consistait principalement en trois classes : une pour lire le fichier CSV, une pour trier et filter les données, et une pour l'interface graphique. Cette dernière étant très dense, peu lisible et extensible, elle a été décomposée en différent sous-paquets contenant différentes classes.

3.1 sticmacpiernov.spreadsheet

- **Config** contient quelques constantes de configuration ou valeurs par défaut.
- **DataCSV** effectue le traitement des données. Le tableau *filterList* est un tableau de fonctions correspondant aux filtres à appliquer. Lors d'une requête, *toArray()* s'exécute et le tableau des données généré par **ReadCSV** est converti en **Stream**. Puis la méthode *filter()* est appliquée pour chacun des filtres de *filterList* (c'est le travail de *applyFilter()*). Puis la méthode *sorted()* est appelée avec notre comparateur *compare()* (qui essaye d'abord de comparer en tant **Float** puis **String**). Puis les colonnes désirées sont extraites en créant un nouveau **Stream** de **String**[] à l'aide de *map()*, lequel agit en utilisant *map()* sur le **Stream** issu du tableau d'indices des colonnes pour récupérer la colonne correspondante et la transformer en **String**[] avec *toArray()*. *limit()* est alors utilisé pour ne conserver qu'un certain nombre de résultat. Enfin le **Stream** est finalement transformé en **String**[][] avec *toArray()*.
- **GUI** créé l'interface graphique. Elle hérite de JFrame et représente donc la fenêtre du programme. Elle créé les différents éléments de l'interface et les affiche, et traite l'action sur le bouton **Search**. Ces différents éléments sont d'une part la barre de menu, dont le contenu est instancié à partir des classes de **sticmacpiernov.spreadsheet.menu**, et d'autre part la zone de critère et la zone de résultat, éléments instanciée à partir des classes de **sticmacpiernov.spreadsheet.panel**.
- **ReadCSV** est la classe qui permet de décoder le fichier CSV. Elle lit d'abord la première ligne du fichier pour récupérer les noms des colonnes dans une liste de chaînes de caractères, puis lit la suite pour générer un tableau à deux dimensions. Le fichier est lu lignes par lignes, et chacune des lignes est lue caractères par caractères pour la décomposer en fonction du délimiteur (par défaut ;). Le délimiteur et le nom de fichier par défaut sont des constantes de cette classe.
- **Spreadsheet** est le point d'entrée du programme. La fonction *main()* lit éventuellement un argument sur la ligne de commande et instancie les différents objets, ce qui a pour effet de lire le fichier CSV et créer la fenêtre de l'interface.

3.1.1 sticmacpiernov.spreadsheet.struct

Deux interfaces pour représenter des données à faire transiter entre l'interface graphique et les méthodes d'extraction de données.

- **Filter** qui représente un filtre à appliquer sur les données : ne conservent que les lignes dont la colonne retournée par *getColumn()* correspond à la valeur retournée par *getValue()*. Il aurait été intéressant d'ajouter d'autres possibilités de critères, tel que "contient", "supérieur à", etc...
- **ResultSet** qui représente le résultat de la requête composée de la liste des colonnes et du tableau de données.

3.1.2 sticmacpiernov.spreadsheet.menu

Les classes correspondant aux différents menu de l'interfaces, toutes héritent de **JMenu**.

- **Columns** est un menu de cases à cocher qui permet de sélectionner les colonnes à afficher dans le résultats. Les colonnes sont lues directement du fichier CSV et sont ajoutées au menu lors de l'instanciation de l'objet. Les valeurs par défaut des cases cochées sont lues à partir de la constante *defaultColumns* définie dans **Config**. Les colonnes à afficher sont récupérées par la suite à l'aide la méthode *getSelected()*.
- **Options** contient simplement un bouton pour quitter.
- **Plus** contient un bouton pour afficher une boîte de dialogue "à propos".

3.1.3 sticmacpiernov.spreadsheet.layouthelpers

Ces deux classes succintes sont inspirées de code trouvé sur Internet et n'apportent que quelques ajustements à l'interface graphique.

- **StayOpenCheckBoxMenuItemUI** empêche la fermeture du menu **Columns** lors d'un clic sur une entrée.
- **JTableAutoSize** créé une JTable dont les colonnes se dimensionnent automatiquement en fonction de leur contenu.

3.1.4 sticmacpiernov.spreadsheet.panel

- **Criteria** représente la liste des critères et permet d'en ajouter à la volée.
- **Criterion** représente un critère sur l'interface.
- **Limit** contient la zone de texte pour entrer la limite de résultat à retourner. Utilise un JFormattedTextField pour n'autoriser que les nombres entiers.
- **ResultsTable** est un JScrollPane contenant une JTableAutoSize, utilisée pour afficher les résultats. De cette manière on a un tableau qu'on peut faire défiler horizontalement et verticalement.
- **Sort** qui contient une liste déroulante pour sélectionner la colonne selon laquelle trier, et les deux boutons radios de l'ordre du tri.