

TD 3

Exercice 1 : Reprise du TD1

R cup rer la librairie de gestion de commandes du TD1. Note : il pourra aussi  tre n cessaire de r cup rer la classe `Paire` (en la recopiant ou en la rajoutant comme d pendance   votre projet).

Question 1 : analyser la classe `DAO`. Pour chaque m thode utilisant l API Stream (`produits`, `selectionCommande`, `selectionCommandeSurExistanceLigne`, `selectionProduits`), en donner une version ne l utilisant pas. Discuter.

Question 2 : refactorer classe `Commande`   l aide de l API Stream. Pour cela :

-   d finir une fonction `formatteurLigne` permettant d obtenir la repr sentation sous forme de cha ne de caract re d une ligne de commande
-   r  crire la m thode `toString` en utilisant `formatteurLigne`, `map` et `collect`
-   faire en sorte que l on puisse utiliser diff rents formatteurs de ligne pour diff rentes commandes (un m me formatteur  tant utilis  pour toutes les lignes d une m me commande), `formatteurLigne`  tant utilis  par d faut si l on ne pr cise rien
-   r  crire la m thode `count` avec `map` et `reduce`
-    crire une m thode g n rique `regrouper` permettant de regrouper des lignes (quelque soient les types dans les paires). Comparer   `Collectors::groupingBy`.
-   r  crire la m thode `normaliser` en utilisant `regrouper`, `forEach` et `reduce`. Comparer   la version initiale. Discuter.

Question 3 : refactorer s l y a lieu vos r ponses   l exercice 4 du TD1 et v rifier que les r sultats sont les m mes.

Exercice 2 : Reprise du TD2

R cup rer la librairie de gestion universitaire du TD2 (en la recopiant ou en la rajoutant comme d pendance   votre projet).

Question 1 : refactorer vos r ponses   l exercice 3 du TD2 en utilisant l API Stream.

Vous  crirez pour cela les fonctions suivantes :

```
  // mati res d'une ann e
  public static final Function<Annee, Stream<Matiere>> matieresA = ???

  // mati res d'un  tudiant
  public static final Function<Etudiant, Stream<Matiere>> matieresE = ???

  // mati res coefficient es d'un  tudiant (version Entry)
  public static final Function<Etudiant, Stream<Entry<Matiere, Integer>>>
matieresCoefE_ = ???
```

```

Ê // transformation d'une Entry en une Paire
Ê public static final Function<Entry<Matiere, Integer>, Paire<Matiere, Integer>>
entry2paire = ???

Ê // matieres coefficients d'un Étudiant (version Paire)
Ê public static final Function<Etudiant, Stream<Paire<Matiere, Integer>>>
matieresCoefE = ???

Ê // accumulateur pour calcul de la moyenne
Ê // ((asomme, acoefs), (note, coef)) -> (asomme+note*coef, acoef+coef)
Ê public static final BinaryOperator<Paire<Double, Integer>> accumulateurMoyenne =
???

Ê // zero (valeur initiale pour l'accumulateur)
Ê public static final Paire<Double, Integer> zero = ???

Ê // obtention de la liste de (note, coef) pour les matieres d'un Étudiant
Ê // 1. obtenir les (matiere, coef)s
Ê // 2. mapper pour obtenir les (note, coef)s, null pour la note si l'Étudiant est
DEF dans cette matiere
Ê public static final Function<Etudiant, List<Paire<Double, Integer>>>
notesPonderees = ???

Ê // obtention de la liste de (note, coef) pour les matieres d'un Étudiant
Ê // 1. obtenir les (matiere, coef)s
Ê // 2. mapper pour obtenir les (note, coef)s, 0.0 pour la note si l'Étudiant est
DEF dans cette matiere
Ê public static final Function<Etudiant, List<Paire<Double, Integer>>>
notesPondereesIndicatives = ???

Ê // replie avec l'accumulateur spécifique
Ê public static final Function<List<Paire<Double, Integer>>, Paire<Double, Integer>>
reduit = ???

Ê // calcule la moyenne à partir d'un couple (somme pondérée, somme coef)s
Ê public static final Function<Paire<Double, Integer>, Double> divise = ???

Ê // calcul de moyenne fonctionnel
Ê // composer notesPonderees, réduit et divise
Ê // exception en cas de matiere DEF
Ê public static final Function<Etudiant, Double> computeMoyenne = ???

Ê // calcul de moyenne fonctionnel
Ê // composer notesPondereesIndicatives, réduit et divise
Ê // pas d'exception en cas de matiere DEF
Ê public static final Function<Etudiant, Double> computeMoyenneIndicative = ???

Ê // calcul de moyenne
Ê public static final Function<Etudiant, Double> moyenne = e -> (e == null || aDEF
.test(e)) ? null

```

```
Ê      : computeMoyenne.apply(e);
```

```
Ê // calcul de moyenne indicative
```

```
Ê public static final Function<Etudiant, Double> moyennelndicative =  
computeMoyenneIndicative;
```