

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

TP9: Expressions lambda et streams

L'objectif des exercices ci-dessous est d'utiliser des expressions lambda dans vos méthodes et de s'initier aux manipulations des streams.

Exercice 1

L'objectif de cet exercice est de vous familiariser avec les Streams, notamment avec les opérations lazy et les opérations finales.

JAVA Streams

```
stringLists.stream()
    .map(str -> str.toUpperCase())
    .collect(Collectors.toList());
```

Intermediate		Terminal	
filter	limit	reduce	findAny
distinct	skip	collect	findFirst
map	sorted	toArray	forEach
flatMap	peek	count	allMatch
		max	anyMatch
		min	

Vous pourrez par exemple visionner (une partie de) la vidéo <https://www.youtube.com/watch?v=YnzisJh-ZNI> et <https://www.youtube.com/watch?v=vuFCTdywMtE>

- `IntStream.iterate()`
- `IntStream.range()`
- `peek()`
- `map()`
- `filter()`
- `forEach()`
- `distinct()`
- `orElse`
- `orElseThrow`
- `orElseGet`
- `limit()`
- `collect(Collectors.toList())`

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

- Téléchargez le fichiers mcdonalds.csv sur <https://introcs.cs.princeton.edu/java/data/mcdonalds.csv>
- Placez ce fichier dans le « resource folder » data
- Créez une classe Restaurant (**private double** latitude, longitude ; **private** String name, address, city, state;) avec accesseurs et constructeur normal
- Créez une classe McDonalds dont le constructeur par défaut crée une List<Restaurant> à partir de la lecture du fichier mcdonalds.csv
- Créez les méthodes (style fonctionnel) suivantes :
 - **int** getSize() //Question 1 : compter le nb de restaurants
 - **long** nbCitiesHaveMcDo() // Question 2 : le nombre de villes qui disposent d'un restaurant McDonald
 - Map.Entry<String, Long> cityHasMostMcDo()// Question 3 : La ville qui dispose du plus grand nombre de MacDonald
- Créez une classe Main.java pour tester vos 3 méthodes

```
* ----- *
* TP java                                     *
*                                           *
* @author Frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD9 - ex1                         *
* ----- *
```

nb of McDos = 13912

The number of cities that have a McDonald : 4927

The city has the most MacDonald : Houston=135

Exercice 2

On désire savoir si Shakespeare était un bon joueur de scrabble (anglais)! Plus précisément, on va s'intéresser au fichier contenant l'intégralité du vocabulaire utilisé dans son œuvre

- Téléchargez les fichiers suivants à l'adresse : [https://introcs.cs.princeton.edu/java/data/](https://introcs.cs.princeton.edu/java/data/words.shakespeare.txt)
 - [words.shakespeare.txt](#) (renommez-le shakespeare.txt)
 - [ospd.txt](#) (le dictionnaire des mots anglais autorisés au scrabble)

- Placez ces fichiers dans le « resource folder » data
- On donne les attributs de classes suivants à placer dans votre fichier Scrabble.java

```
// score of each letter in scrabble english game
static final int[] scrabbleENScore = {
//   a,   b,   c,   d,   e,   f,   g,   h,   i,   j,   k,   l,   m,   n,
0,   p,   q,   r,   s,   t,   u,   v,   w,   x,   y,   z
1,   3,   3,   2,   1,   4,   2,   4,   1,   8,   5,   1,   3,   1,   1,
3,   10,  1,   1,   1,   1,   4,   4,   8,   4,   10 };

// enable occurrences of English distribution of scrabble coins
private static final int[] scrabbleENDistrib = {
//   a,   b,   c,   d,   e,   f,   g,   h,   i,   j,   k,   l,   m,   n,
0,   p,   q,   r,   s,   t,   u,   v,   w,   x,   y,   z
9,   2,   2,   1,  12,  2,   3,   2,   9,   1,   1,   4,   2,   6,   8,
2,   1,   6,   4,   6,   4,   2,   2,   1,   2,   1 };
```

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

- Créez une classe une classe Main.java contenant les réponses aux questions suivantes :
1. Écrivez **Function<String, Integer>** **computeScore** et calculez le score du mot BONJOUR
//3 1 1 8 1 1 1
 2. Comptez le nombre de mots par score dans l'œuvre de Shakespeare. On écrira le résultat sous la forme d'un Dictionnaire clé=score / valeur=nombre de mots,
 3. Limitez le traitement de la question 2 et affichez les mots dont le score est >36
 4. Affichez les mots qui existent dans le dictionnaire officiel et dont le score est >28
 5. Comptez le nombre de mots par score dans l'œuvre de Shakespeare en s'assurant que la distribution des jetons du jeu de scrabble en anglais permet d'écrire. On procédera par étape :
 - (a) Commencez par écrire **Function<String, Map<Integer, Long>>** **letterCount** qui retourne un Map ayant pour clé le code ascii de chaque lettre du mot à évaluer et pour valeur le nombre d'ocurences. Par exemple, letterCount(« bonjour ») = {114=1, 98=1, 117=1, 106=1, 110=1, 111=2}
 - (b) Écrivez le **Predicate<String>** **availableWord**
 - (c) Utilisez **letterCount** et **availableWord** pour écrire les mots qui existent dans le dictionnaire officiel et dont le score est >28
 6. Affichez les mots qui correspondent aux critères de la question 5 dont le score est >24
 7. Au scrabble, il existe deux « blancs » qui peuvent prendre la valeur de la lettre de son choix et avec un score de 0. Ecrivez **Function<String, Integer>** **nbBlankNeeded** . Vérifiez votre fonction avec le mot "buzzards"
 8. Ecrivez une version améliorée de votre fonction qui calcule le score : **Function<String, Integer>** **computeScoreV2**.Vérifiez le score du mot « Buzzards » //19
 9. Affichez les mot de Shakespeare, qui sont dans le dictionnaire du scrabble, qu'il est possible d'écrire en utilisant forcément au moins 1 blanc et dont le score est > 18

Découvrir la programmation orientée objet avec Java

Travaux pratiques

frederic.rallo@univ-cotedazur.fr

```
* ----- *
* TP java *
* *
* @author Frédéric rallo - frederic.rallo@univ-cotedazur.fr *
* @version TD9 - ex2 *
* ----- *
```

question1: score de "BONJOUR"=16

question2: {1=20, 2=79, 3=168, 4=452, 5=750, 6=1357, 7=1905, 8=2362, 9=2704, 10=2866, 11=2851, 12=2812, 13=2406, 14=2044, 15=1687, 16=1328, 17=981, 18=738, 19=487, 20=371, 21=257, 22=174, 23=105, 24=79, 25=62, 26=39, 27=24, 28=22, 29=10, 30=9, 31=6, 32=3, 33=3, 34=2, 36=1, 40=2}

question3: {40=[honorificabilitudinitatibus, Nebuchadnezzar]}

question4: {33=[whizzing], 29=[buzzards]}

question5: {2=26, 3=74, 4=241, 5=426, 6=808, 7=1145, 8=1422, 9=1602, 10=1646, 11=1505, 12=1315, 13=990, 14=700, 15=531, 16=353, 17=233, 18=119, 19=65, 20=40, 21=14, 22=10, 23=7, 24=5, 25=1, 26=1}

question6: {25=[quickly], 26=[squeezes]}

question7: nb Blank = 1

question8: score Buzzards => (old) 29 (new) 19

question9: {19=[buzzards], 20=[hazarded, vizarded], 23=[whizzing]}