TD3-Bis

TD Optionnel

Pour cette partie, vous ne devez utiliser que les classes de la bibliothèque stream de Java. Vous ne devez pas utiliser de boucles `for` ou `while`.

Exercice 1:

Écrivez une méthode qui prend une liste de chaînes de caractères en entrée et renvoie une liste contenant la longueur de chaque chaîne.

Exemple d'entrée : `["Java", "Python", "JavaScript", "C", "Ruby"]` **Sortie attendue :** `[4, 6, 10, 1, 4]`

Utilisez la méthode `map` de la bibliothèque `Stream` pour transformer chaque chaîne en sa longueur.

```
List<String> chaines = Arrays.asList("Java", "Python", "JavaScript", "C", "Ruby");
List<Integer> longueurs = chaines.stream()
                                 .map(String::length)
                                 .collect(Collectors.toList());
```

Exercice 2:

Écrivez une méthode qui prend une liste d'entiers en entrée et renvoie une liste contenant le carré de chaque entier.

Exemple d'entrée : `[1, 2, 3, 4, 5]` **Sortie attendue :** `[1, 4, 9, 16, 25]`

Utilisez la méthode `map` de la bibliothèque `Stream` pour transformer chaque entier en son carré.

```
List<Integer> entiers = Arrays.asList(1, 2, 3, 4, 5);
List<Integer> carres = entiers.stream()
                              .map(n -> n * n)
                              .collect(Collectors.toList());
```

Exercice 3:

Écrivez une méthode qui prend une liste de chaînes de caractères en entrée et renvoie une liste contenant uniquement les chaînes dont la longueur est supérieure à 5 caractères.

Exemple d'entrée : `["Apple", "Banana", "Strawberry", "Orange", "Kiwi"]` **Sortie attendue :** `["Strawberry", "Banana", "Orange"]`

Utilisez la méthode `filter` de la bibliothèque `Stream` pour sélectionner les chaînes dont la longueur est supérieure à 5 caractères.

```java
List<String> chaines = Arrays.asList("Apple", "Banana", "Strawberry", "Orange", "Kiwi");
List<String> longuesChaines = chaines.stream()
                                     .filter(s -> s.length() > 5)
                                     .collect(Collectors.toList());
```

Exercice 4:

Écrivez une méthode qui prend une liste d'entiers en entrée et renvoie la somme de tous les entiers.

Exemple d'entrée : `[1, 2, 3, 4, 5]` **Sortie attendue :** `15`

> Utilisez la méthode `reduce` de la bibliothèque `Stream` pour calculer la somme de tous les entiers.

```java
List<Integer> entiers = Arrays.asList(1, 2, 3, 4, 5);
int somme = entiers.stream()
                   .reduce(0, Integer::sum);
```

Exercice 5:

Écrivez une méthode qui prend une liste d'entiers en entrée et renvoie la moyenne de tous les entiers.

Exemple d'entrée : `[10, 20, 30, 40, 50]` **Sortie attendue :** `30.0`

> Utilisez la méthode `average` de la bibliothèque `Stream` pour calculer la moyenne de tous les entiers.

```java
List<Integer> entiers = Arrays.asList(10, 20, 30, 40, 50);
double moyenne = entiers.stream()
                        .mapToInt(Integer::intValue)
                        .average()
                        .orElse(0.0);
```

Exercice 6:

Écrivez une méthode qui prend une liste de chaînes de caractères en entrée et renvoie une seule chaîne contenant toutes les chaînes concaténées, séparées par des virgules.

Exemple d'entrée : `["Hello", "World", "Java", "Programming", "Language"]` **Sortie attendue :** `"Hello, World, Java, Programming, Language"`

> Utilisez la méthode `collect` de la bibliothèque `Stream` avec un `Collectors.joining(", ")` pour concaténer toutes les chaînes avec des virgules.

```java
List<String> chaines = Arrays.asList("Hello", "World", "Java", "Programming", "Language");
String concatenee = chaines.stream()
                           .collect(Collectors.joining(", "));
```

Exercice 7:

Écrivez une méthode qui prend une liste de chaînes de caractères en entrée et renvoie une liste contenant toutes les chaînes en majuscules.

Exemple d'entrée : `["apple", "banana", "orange", "kiwi", "strawberry"]` **Sortie attendue :** `["APPLE", "BANANA", "ORANGE", "KIWI", "STRAWBERRY"]`

Utilisez la méthode `map` de la bibliothèque `Stream` pour transformer chaque chaîne en majuscules.

```
List<String> chaines = Arrays.asList("apple", "banana", "orange", "kiwi", "strawberry");
List<String> majuscules = chaines.stream()
                                 .map(String::toUpperCase)
                                 .collect(Collectors.toList());
```

Exercice 8:

Écrivez une méthode qui prend une liste d'entiers en entrée et renvoie une liste contenant uniquement les nombres pairs, triés par ordre croissant.

Exemple d'entrée : `[3, 8, 2, 5, 10, 7]` **Sortie attendue :** `[2, 8, 10]`

Utilisez la méthode `filter` de la bibliothèque `Stream` pour sélectionner les nombres pairs, puis la méthode `sorted` pour les trier par ordre croissant.

```
List<Integer> entiers = Arrays.asList(3, 8, 2, 5, 10, 7);
List<Integer> pairs = entiers.stream()
                             .filter(n -> n % 2 == 0)
                             .sorted()
                             .collect(Collectors.toList());
```