TP DE JAVA AVANCÉ N°4

Exercice 1. Création de flux

1. À l'aide de la méthode statique `of` de `Stream`, créer un flux contenant les `Strings` : `Vive`, `les`, `...` (choisir la dernière chaîne). On pourra afficher le flux fini `stream` avec

   ```
   System.out.println(Arrays.toString(stream.toArray()));
   ```

2. Créer le même flux mais avec la méthode `builder` de `Stream` mais toujours en une seule ligne

3. À l'aide de la méthode `stream` de `Collection`, créer un flux identique au précédent

4. À l'aide de la méthode statique `stream` de `Arrays`, créer un flux *d'entier* depuis un tableau d'entiers

5. À l'aide de la méthode statique `generate` de `LongStream`, créer un flux infini des long correspondant à l'heure retournée par `System.nanotime`

6. À l'aide de la méthode statique `iterate` de `Stream`, créer un flux infini des entiers (`BigInteger`) pairs.

Exercice 2. Utilisation

1. À l'aide de la méthode `forEach`, afficher les contenus de tous les flux finis précédents.

2. Avec en plus la méthode `limit`, afficher les 1000 premiers éléments de tous les flux précédents.

3. À l'aide de la méthode `mapToObj`, convertir chaque entier de l' `IntStream` de 1.4 en chaîne de caractères dans un nouveau flux

4. Écrire une méthode qui prend un `BigInteger` en entrée et, à l'aide des méthodes `bitLength` et `testBit` de `BigInteger`, produit le flux des bits

(sous forme d'`int`) de cet entier, dans l'ordre *MSB first*.

5. À l'aide de la méthode `flatMapToInt`, créer le flux des caratères (`IntStream`, il n'y a pas de `CharStream`) correspondant à l'impression de la suite des entiers du flux 1.6 représentés en binaire. Si le flux est :

   > 0 2 4 6 8 10 ...

   le flux en sortie sera la suite des *caractères* (le ␣ est un espace) :

   > 0 , ␣ 1 0 , ␣ 1 0 0 , ␣ 1 1 0 , ...

6. À l'aide des méthodes `filter` et `reduce` de `Stream` ainsi que `isProbablePrime` de `BigInteger`, calculer la somme des 1000 premiers nombres premiers.

7. Tester toutes ces opérations avec des flux en parallèle et voir si c'est plus rapide

Exercice 3

1. Implémenter des listes doublement chaînées non mutable pouvant se concaténer de manière efficace
2. À l'aide de la méthode `collect` de `Stream`, récupérer tous les éléments du flux dans votre implémentation

Exercice 4

Implémenter un crible d'Eratosthène à l'aide de flux :

- Partir d'un flux des entiers comme flux *courant*
- Itérer :
  - Récupérer un élément `x` sur le flux *courant*
  - Insèrer cet élément dans le flux de *sortie*
  - Ajouter un filtre sur le flux *courant* qui supprime les éléments multiples de `x`

Calculer la complexité.