POCA — TP noté n°3
Java: la concurrence

Jean-Baptiste Yunès
Jean-Baptiste.Yunes@u-paris.fr

9 décembre 2021

Rendu final

L'intégralité du code produit, les tests, etc doit être fourni à la date indiquée par l'enseignant. Il est important de ne fournir que le code source, et de prendre bien soin que rien ne manque. Le code sera fourni sous la forme d'une archive `tar.gz` ou `zip` (pas autre chose). Le code devra comprendre un fichier `README.txt` contenant le nom et le prénom (dans cet ordre !) de l'élève concerné. Les modalités de rendu seront donnés par l'enseignant mais se feront sur `moodle.u-paris.fr` dans le cours POCA et sous aucune autre forme.

TP n°3

Dans ce troisième TP, on vise à expérimenter les différentes techniques de multithreading, de comparer leur efficacité et leur «facilité» d'écriture.

L'idée est d'obtenir une application comptant le nombre d'occurences d'un mot donné en argument de ligne de commande et à rechercher dans différents fichiers dont les noms seront aussi donnés en arguments. Ex : `java Search Italie discours1.txt discoursMedium.txt discours1.txt discours2.txt discours1.txt discours1.txt discours1.txt discours3.txt`. On simule ici la présence de nombreux fichiers par la répétition de certains d'entre eux (en prenant soin de mélanger un peu les tailles : beaucoup de petits, peu de grands).

D'autre part on prendre soin de mesurer les différents temps de calculs. Le résultat devra donc ressembler à :

```
Méthode xxxxSearch
Italie : 2356
Temps: 12345 ms.
```

On prendra soin d'essayer d'utiliser les constructions de la programmation fonctionnelle autant que faire ce peut.

Préliminaire

Sur la page du cours, on trouvera quatre fichiers texte à utiliser comme source de données.

1 Exercice

Écrire un premier programme (`SequentialSearch`) qui effectue le traitement de façon séquentielle.

2 Exercice

Écrire un second programme (`ThreadSearch`) qui effectue le traitement en utilisant le multithreading à l'aide des `Thread`s et `Runnable`s.

3 Exercice

Écrire un troisième programme (`FutureSearch`) qui effectue le traitement en utilisant les `ExecutorService`s et les `Future`.

4 Exercice

Écrire un quatrième programme (`QueueSearch`) qui effectue le traitement en utilisant les `ExecutorService`s et un `ArrayBlockingQueue` de taille 5.

5 Exercice

Écrire un cinquième programme (`StreamSearch`) qui effectue le traitement à l'aide de `Stream`s parallèles.

6 Exercice

Écrire un cinquième programme (`RecursiveSearch`) qui effectue le traitement à l'aide de `RecursiveTask`s.

7 Exercice

Fusionner tous ces tests en un seul.