exo2:

exo2.1

L’exécution de programme donne:

```
la thread 2 d ébute: Thread id9
la thread 8 d ébute: Thread id18
la thread 7 d ébute: Thread id16
la thread 6 d ébute: Thread id13
la thread 3 d ébute: Thread id15
la thread 9 d ébute: Thread id10
la thread 5 d ébute: Thread id11
la thread 1 d ébute: Thread id17
la thread 4 d ébute: Thread id12
la thread 0 d ébute: Thread id14
la thread 2 travaille : Thread id 9
la thread 2 termine : Thread id 9
la thread 9 travaille : Thread id 10
la thread 9 termine : Thread id 10
la thread 8 travaille : Thread id 18
la thread 8 termine : Thread id 18
la thread 5 travaille : Thread id 11
la thread 5 termine : Thread id 11
la thread 4 travaille : Thread id 12
la thread 4 termine : Thread id 12
la thread 0 travaille : Thread id 14
la thread 1 travaille : Thread id 17
la thread 1 termine : Thread id 17
la thread 0 termine : Thread id 14
la thread 7 travaille : Thread id 16
la thread 6 travaille : Thread id 13
la thread 6 termine : Thread id 13
la thread 7 termine : Thread id 16
la thread 3 travaille : Thread id 15
la thread 3 termine : Thread id 15
```

exo2.2

Dans ce contexte multithread, on remarque que la valeur de nextID est differente d’un processus à un autre, il n’ya pas un accès concurrent sur nextID ,c’est comme si Chaque thread possède une variable nextID propre à lui . Ceci est est assurée grace au mécanisme de la classe `ThreadLocal (la classe interne de la classe ThreadID)` .

Exo2.3

En concurrence , tous les threads d'un même processus partagent le même espace d’adressage. Donc, les données situées dans une variable statique ou globale sont exactement au même emplacement mémoire pour tous les threads, et correspondent donc à la même entité.

Pour éviter qu’une donnée soit partagée par plusieurs threads, La classe ThreadLocal permet d' encapsuler des données qui seront accessibles par tous les traitements exécutés dans le thread.

La classe ThreadLocal implémente un mécanisme qui permet d'associer une donnée à un thread et de permettre son accès dans tous les traitements exécutés par ce thread.

Le principe de ThreadLocal : stocker des données dans le Thread courant afin de les récupérer plus tard, dans une autre méthode par exemple, sans avoir à les passer en argument à toute la chaine d’appel.