Formation : INFO2

INFO
INGÉNIEUR
ESIPE

Rapport du TD n°03
Concurrence

Guillaume CAU
09/10/2019

UNIVERSITÉ PARIS-EST
MARNE-LA-VALLÉE

Esipe
INGÉNIEURS PARIS EST

INGENIEURS
2000
L'EXCELLENCE AU SERVICE DE L'ALTERNANCE

Exercice 01 :

Pour créer une classe dite Thread-Safe, il faut utiliser les blocs Syncronized. Ces blocs demande un « *lock* » en paramètre. Ce lock permet à un thread de rentrer dans une partie du code en étant assuré qu'aucun autre thread n'exécutera le même code en même temps. Le lock doit respecter certaines caractéristiques :

- Le lock doit être un objet. Les méthodes nécessaires au Syncronized sont définies dans la classe Object.
- Le lock doit être private et final. Ces conditions sont nécessaires pour éviter les effets de bords.
- L'objet ne doit pas utiliser d'interning.

Exercice 02 :

La classe HonorBoard n'est pas Thread-Safe. En effet, la fonction *set* n'est pas atomique. Les threads peuvent donc être interrompus en plain milieu de la méthode *set*.

De ce fait, il existe au moins 3 situations :

1. Le thread **T1** rentre dans la fonction set et n'est pas interrompu lors de son exécution. Le nom prénom reste donc cohérent.
2. Le thread **T1** exécute la fonction set et est interrompu après la première ligne. C'est le thread **T2** qui prend la main ensuite et écrase la modification de **T1** (ligne 1) puis exécute la ligne 2. **T1** reprend ensuite la main et exécute la ligne2.
   Un problème de cohérence à donc été créé. La ligne 1 peut avoir la valeur de **T2** et la ligne 2 la valeur de **T1**.
3. Le thread **T1** n'est pas interrompu et exécute *set* en entier. **T2** exécute ensuite la ligne 1 mais est interrompu par le thead **T3** qui affiche le résultat. Il affiche donc la ligne 1 de **T2** et la ligne 2 de **T1**.

Exercice 03 :

Lorsque l'on execute la méthode main de la classe StupidRendezVous sans mettre la ligne en commentaire, l'algorithme fonctionne comme prévu.
Le thread main créé un thread T1 qui attend 5 secondes puis modifie la valeur de la variable *value* à null.
Lorsque main repère que la valeur de *value* à été modifiée, il stoppe le programme.

Lorsque la ligne du *sleep* est commentée, le programme ne se termine plus.
L'explication que j'ai trouvée est la suivante :

Le JIT optimise le code pour un thread uniquement. Il a donc remarqué que la valeur de *value* ne changera jamais une fois que l'exécution de la boucle aura commencée.
Il remplace donc la boucle par une boucle infinie, et c'est pourquoi le programme ne termine plus.

Effectivement, avec la commande TOP, nous pouvons voir que le programme fait de l'attente active et mobilise donc des ressources pour rien.