Notes de cours de Programmation objet concurrente

Yann Miguel

10 septembre 2020

Table des matières

1 Introduction **2**

2 Cours 1 **3**
2.1 Un thread 3
2.2 Éléments de synchronisation 3
2.2.1 Synchronisation par verrou 3
2.2.2 Synchronisation par signaux 4

3 Informations importantes **5**

1 Introduction

Cette UE est importante car savoir utilisé plusieurs processeurs et threads est important dans l'informatique moderne.
Les threads sont aussi appelés processus légers, à l'opposés des processus lourds, qui contiennent leur propres données.

Les erreurs sont beaucoup plus communes en programmation parallèle.

Il y aura 5 cours de 2h :

1. les threads en java
2. le problème de l'atomicité
3. présentation des outils en java
4. programmation optimiste
5. aperçu du modèle mémoire java

2 Cours 1

2.1 Un thread

Un thread est un objet qui execute du code. Afin d'écrire cela proprement, il faudra écrire des classes qui correspondront aux threads.

Le moyen le plus simple de faire ça est de créer une classe qui hérite de la classe Thread, ce qui oblige à créer une méthode run, qui sera exécutée par le thread. **IMPORTANT !** il faut utiliser start et non run pour lancer le thread.

Une autre approche, qui est peu recommandée, est de créer une classe qui hérite de Runnable, plutôt que d'hériter de Thread.

Une classe de Thread ou de Runnable peux contenir le main, et on peux lancer un thread en le créant.
Il existe une méthode nomée getName(), qui permet de récupérer le nom d'un thread. La méthode setName() permet de changer son nom. Pour accéder à un thread, on a besoin de sa référence, et non de son nom.

Les priorités d'un thread n'on que peux d'utilité. Il servent à prioriser les threads, si jamais le système à besoin ne peux pas gérer touts les threads.

Un thread peux rencontrer des instructions qui le feront sortir du runnable, par exemple un verrou.
Si un thread est dans un verrou, il doit attendre l'authorisation pour continuer.
Il entrera dans un état TimeWainting si il doit attendre un certain temps, ou bien dormir (méthode sleep(int ms)).

La méthode join() permet de mettre en pause un thread tant qu'un autre n'a pas fini.
Les méthodes sleep() et join() peuvent soulever une execption.
La méthode yield() pousse un thread à laisser d'autres s'éxecuter. Son usage est déconseillé, car cela diminue la propreté du code.

Afin qu'un thread puisse s'éxecuter, il faut qu'il ai accès à un coeur. Il n'y a à priori aucun intérêt d'avoir plus de threads que de coeur, sauf si cela peux simplifier le code.

2.2 Éléments de synchronisation

2.2.1 Synchronisation par verrou

Si plusieurs threads doivent toucher les mêmes données, on peux verrouiller les données à chaque fois qu'un thread les change. Cela permet de faire en sorte que seul un thread change les données à la fois.

Le mot clé "synchronized" permet de ne faire éxcuter du code que si on a accès au verrou. Si un thread a besoin d'un verrou, mais ne peux pas y accèder, il passe en état blocked. Il repassera en état runnable que lorsqu'il aura accès au verrou.

Prendre un verrou est appelé "verouiller", et libérer un verrou est appelé "dévérouiller".
En java, il est impossible de tenter de prendre un verrou si on utilise "synchronized". C'est à dire, que "synchronized" ne permet pas d'éxecuter du code si le verrou n'est pas libre lorsqu'on le demande.
Lorsqu'une méthode est déclarée "synchronized", elle ne peux pas être éxecutée si le thread qui veut l'éxecuter n'a pas le verrou. Si une méthode est synchronisée et statique, on demande le verrou de la classe plutôt que celui de l'objet.

2.2.2 Synchronisation par signaux

Le principe de la programmation par signaux est de régler les problèmes de type producteur-consommateur. Par exemple, le cas dans lequel un thread a besoin d'une donnée produite par un autre, il doit alors attendre que le thread "producteur" produise la donnée.

3 Informations importantes

Dans cette UE, le projet sera les rendus de TP, et comptera pour $\frac{1}{4}$ de la note finale.
Les $\frac{3}{4}$ suivant seront l'exam final.
Durant la seconde session, la note de projet sera gardée seulement si elle augmente la note finale.
MCC session2 : max($\frac{1}{4}$ TP + $\frac{3}{4}$ ET , ET)

L'examen terminal sera sur papier. Les compte rendus de TP seront un exercice à rendre.