

Atomicité, CAS

Exercice 1 - Liste concurrente et atomicité

On souhaite écrire une implantation de liste chaînée concurrente n'utilisant ni section critique ni verrou.

La liste devra posséder deux méthodes `addLast(E e)` et `toString()` permettant respectivement d'ajouter un élément en queue de liste et d'afficher la liste. Lors de la création, la liste sera créée avec un maillon.

L'idée pour éviter toutes synchronisations lors de l'ajout consiste à récupérer le maillon à la fin de la liste puis à tester en une opération atomique si le maillon est toujours la fin de la liste et si oui à lui assigner le nouveau maillon créé. Si ce n'est pas le dernier maillon, on récupère de nouveau la fin de la liste et l'on recommence le même algorithme.

Dans un premier temps, nous allons définir la liste comme contenant juste un élément ainsi qu'une référence sur un autre élément de la liste. L'ajout à la fin de la liste se fera par un parcours du chaînage.

- 1 Définir la classe `ConcurrentLink` avec ces champs `element` et `next` ainsi que son constructeur `ConcurrentLink(E firstElement)`.
- 2 Expliquer pourquoi le champ `element` doit être déclaré `final` ?
- 3 Implanter la méthode `addLast` en utilisant pour tester et assigner en une opération atomique, la classe `AtomicReferenceFieldUpdater` du package `java.util.concurrent.atomic`.
- 4 Implanter la méthode `toString`.

On souhaite maintenant séparer l'implantation de la liste de celle d'un maillon dans le but de stocker le dernier maillon pour effectuer un ajout plus efficace.

- 1 Définir la classe `ConcurrentList` pour qu'elle contienne deux champs `head` et `tail` correspondant respectivement au début et à la fin de la liste ainsi que la classe interne `Entry` correspondant à un maillon de la liste chaînée.
Pour simplifier les choses, la liste ne pourra être vide, elle sera donc construite avec un élément.
- 2 Implanter la méthode `addLast`
Attention, il y a un petit piège lorsque l'on met à jour la référence correspondant à la fin de la liste.
- 3 Implanter la méthode `toString`.

Si vous avez réussi bravo, vous venez de ré-implanter une partie de la classe `ConcurrentLinkedQueue`.