TD threads : interruption et concurrence

Exercice 1 :

Interruption d'un thread via le mécanisme interrupt()/InterruptedException

On reprend le corrigé de l'exercice 2 du td1 en utilisant cette fois ci le principe d'interruption des threads basé sur la méthode *interrupt()* de java.lang.Thread, qui indique à un thread qu'il a reçu une demande d'interruption.

1. Celui-ci[1], s'il est actif, peut tester si on l'a interrompu avec la méthode isInterrupted().

2. S'il est en veille (Thread.sleep(1) par exemple), une InterruptedException sera levée et un try/catch permet de la prendre en compte.

Ecrire les 2 solutions au sein de la classe Compteur.

Exercice 2 :

Attente de la fin d'un thread via join()
Concurrence et thread-safety

La classe Incrementeur qui est fournie sur le serveur commun produit des résultats faux et irréguliers (la valeur finale devrait être 21). Testez le programme plusieurs fois (notamment en faisant varier la valeur du Thread.sleep() du main), constatez ces résultats et corrigez les 2 problèmes constatés :

1. vous introduirez une attente explicite de la fin des 2 incrémenteurs dans le main via la méthode *join()* de la classe Thread[2].
2. vous rendrez la classe Incrementeur *thread-safe*.

Pour le 2, il faudra, dans l'ordre :

- identifier le problème de concurrence (quelle est la ressource partagée ?) et repérer le bloc de code section critique
- avoir un objet correspondant à la ressource partagée puis verrouiller la section critique repérée

[1] Thread.currentThread() renvoie le thread courant (celui qui s'exécute).
[2] unThread.join() provoque l'attente de la fin du run() de unThread.