TP 3 : Programmation Parallèle : Threads & Synchronisation

Exercice 1 :

Il s'agit ici d'implémenter un système de **producteurs/consommateurs**. Ces producteurs / consommateurs communiquent par un **buffer partagé** pouvant contenir des entiers. Si le buffer est vide, On met le consommateur en attente, un **producteur** a le droit de le **remplir** et il signale ensuite à tout le monde qu'il l'a rempli. Au contraire si le buffer contient une chaîne de caractères (c'est-à dire qu'il est plein), le producteur doit attendre qu'un consommateur consomme ce qu'il y a dans le buffer.

Quant au **consommateur**, il essaie de **lire** ce qu'il y a dans le buffer, s' il y arrive, il vide le buffer et il **signale cela à tout le monde**, sinon il attend qu'un producteur produise une donnée dans le buffer.

Ecrivez les classes **Consommateur et Producteur** avec une classe **Buffer** implémentée par vos soins.

Indication : Pour cet exercice, on utilisera les méthodes ***wait*** et ***notifyAll*** de la classe Thread. Lorsque la méthode ***wait*** est invoquée à partir d'une méthode ***synchronized***, en même temps que l'exécution est suspendue, le verrou posé sur l'objet par lequel la méthode a été invoquée est relâché. Dès que la condition de réveil survient, le thread attend de pouvoir reprendre le verrou et continuer l'exécution. La méthode ***notify*** réveille un seul thread. Si plusieurs threads sont en attente, c'est celui qui a été suspendu le plus longtemps qui est réveillé. Lorsque plusieurs threads sont en attente et qu'on veut tous les réveiller, il faut utiliser la méthode ***notifyAll***().

Producteur	***Consommateur***	***Buffer***
Tant que vrai Produire (m) Buffer. Mettre(m) Fin tant que	Tant que vrai Buffer. Prendre() Consommer() Fin tant que	Prendre () **{…}** Mettre () **{…}**