# TP – Synchronisation

# License DANT – PITTON Olivier

## Exercice 1 – Thread safe file

Ouvrir la classe NotThreadSafeList et implémenter les méthodes :

* add : Ajoute un élément à la liste. Cette méthode est bloquante tant que la file ne fait pas moins de MAX\_SIZE éléments.
* poll : Méthode non bloquante. Retire le premier élément de la liste et le retourne, null si la liste est vide.
* take : Méthode bloquante. Retire le premier élément de la liste et le retourne. Si la liste est vide, bloque tant qu’un élément n’a pas été ajouté.

Vous utiliserez instructions synchronized, volatile, wait, notify uniquement.

## Exercice 2 – SpinLock

Un SpinLock est un verrou qui ne met pas en attente le thread appelant, mais va continuer d’utiliser du CPU jusqu’à ce que le verrou soit libéré. On utilise des SpinLock lorsque l’on sait que la section critique à protéger est très courte, et que cela sera moins coûteux de garder le CPU en le faisant tourner dans le vide, plutôt que d’endormir le thread et le réveiller.

Ecrire une classe SpinLock et définir un test sur un compteur, comme vu en cours, pour s’assurer qu’il fonctionne.

## Exercice 3 – Semaphore

Un sémaphore est un verrou autorisant un certain nombre de threads avant de bloquer. Un sémaphore qui n’autorise qu’une entrée est l’équivalent d’un verrou en écriture.

Ecrire une classe Semaphore en utilisant les ReentrantLock et définir un test sur un compteur, comme vu en cours, pour s’assurer qu’il fonctionne.