# TP 6 Systèmes concurrents et distribués

**Exercice 1 :** On a vu à la page 109 du cours une implémentation en Java d'une sémaphore booléenne.

1. Montrez que cette implémentation n'est pas starvation-free, c'est-à-dire qu'un thread qui est en attente dans P() peut ne jamais accéder à la section critique.
2. Proposez une implémentation starvation free de la sémaphore booléenne.

Pour le point 2. une idée est d'utiliser deux files d'attentes successives. Les threads sortant (méthode V()) vont d'abord laisser entrer les threads de la 2ème file d'attente. C'est seulement lorsque cette file est vide que les threads de la première file passent dans la 2ème ; la 1ère file étant toujours bloquée pour d'éventuels nouveau threads.

Avec cette stratégie, un thread dans la 2ème file sera servi après un temps fini.

Pour cet exercice utilisez les mécanismes de synchronisations Java *wait/notify* et *synchronized*.

**Exercice 2 :** On suppose que N threads exécute le programme suivant

Ti

```
X[i]=1
Y[i]=X[(i-1) mod N]
end
```

Montrez qu'une fois que les N threads ont exécutés leur programme il existe au moins un thread i pour lequel Y[1]=1.

1. Faites une preuve 'comportementale' qui utilise les flèches.
2. Utilisez la méthodes des invariants, c'est-à-dire définissez ce qu'est un état du système et montrez que votre invariant est satisfait pour chaque transition du système.