Exam

Lundi 19 mars—11h à 12h

Les 3 exercices sont indépendants. Pour les 2 premiers exercices, pour chacune des questions si la réponse est positive, vous justifierez brièvement et si la réponse est négative vous donnerez un contre exemple.

Exercice 1.— On propose l'implémentation suivante d'une file:

```
public class FileConcur {
    final static int QSIZE=20000;
    private int head=0;
    private int tail=0;
    private int [] cell= new int [QSIZE];
    public synchronized void enq(int o){
        cell[(tail++)%QSIZE]=o;
    }
    public synchronized int deq(){
        return cell[(head++)%QSIZE];
    }
}
```

1. Cette implémentation est-elle atomique? est-elle wait free?
2. Pour la fonction deq on a maintenant:

```
public int deq(){
    return cell[(head++)%QSIZE];
}
```

Cette implémentation est-elle atomique? est-elle wait free?

Exercice 2.— On suppose que vous avez réalisé une implémentation atomique pour les fonctions scan et update du snapshot. Par ailleurs, vous avez réalisé une fonction collect qui fait une lecture séquentielle des éléments du tableau.

```
public interface Snapshot<T> {
    public void update(T v);
    public T [] scan();
    public T [] collect();
}
```

On suppose que le tableau est de taille n ($n > 2$) et est initialisé avec -1. On suppose que la thread i exécute:

```
update(i); col=collect();sc=scan();
```

On notera T_i la valeur de la variable T à la fin de l'exécution de la thread i .

Si T et T' sont deux tableaux, $T \subseteq T'$ si et seulement si $(\forall i \ T[i] \neq -1 \Rightarrow T[i] = T'[i])$

Soient deux threads i et j :

1. A-t-on $col_i \subseteq col_j$ ou $col_j \subseteq col_i$?
2. A-t-on $sc_i \subseteq sc_j$ ou $sc_j \subseteq sc_i$?
3. A-t-on $col_i \subseteq sc_i$? (indication: on pourra se référer à l'implémentation atomique du snapshot vu en cours/TD)
4. Garde-t-on une implémentation atomique et wait free si on implémente le `scan()` en faisant des `collect` jusqu'à ce que deux `collect` retourne le même tableau ?

Exercice 3.— On suppose que l'on a utilisé la classe `ReentrantReadWriteLock` pour implémenter `ReadWriteLock`. On a 5 threads qui utilisent un verrou V `ReadWriteLock` crée par `ReentrantReadWriteLock(true)`. Les 3 threads A, B et C exécutent:

```
V.ReadLock.lock(); CODE1; V.ReadLock.unlock();
```

et les 2 threads X et Y exécutent:

```
V.WriteLock.lock(); CODE2; V.WriteLock.unlock();
```

1. A un instant donné, combien de threads peuvent exécuter `CODE1`?
2. A un instant donné, combien de threads peuvent exécuter `CODE2`?
3. A un instant donné, combien de threads peuvent exécuter `CODE1` et `CODE2`?
4. Si A exécute `CODE1` (A débute à l'instant 0 et termine à l'instant 5), puis X commence son exécution à l'instant 1, puis B commence son exécution à l'instant 2, puis Y à l'instant 3.
 - (a) Dans quel ordre les threads exécutent `CODE1` et `CODE2` (on indiquera les instants auxquels débute les `CODEi`)?
 - (b) Si on voulait que l'ordre d'exécution soit "début `CODE1` de A; début `CODE1` de B; début `CODE2` de X, début `CODE2` de Y " sans changer l'ordre auquel débute chaque thread que devrait-on faire?