Université Paris Cité

Master 2 - Année 2023-24

Programmation répartie

TP2-objet concurrent

Exercice 1.—
On suppose que l'on dispose de registre SRSW régulier booléen et on veut implémenter des registres SRSW réguliers pouvant contenir des entiers (on nommera cette classe RegInt). Basiquement on utilise un tableau de registres booléens tel que si on veut écrire la valeur i on écrit vrai dans le $i+1$eme élément du tableau. On lit le registre en faisant une lecture des différentes éléments du tableau en commençant à l'indice 0 jusqu'à trouver une valeur vrai.

1. Que se passe-t- il si on réalise l'écriture en effaçant d'abord l'écriture précédente (en supposant que 0 était la valeur du resistre) i.e. write(5) puis write(3) a pour effet d'écrire faux a l'indice 0 puis vrai dans l'indice 6 (pour write(5)) puis faux dans l'indice 6 puis vrai dans l'indice 4(pour write(3)).
2. Que se passe-t-il si on réalise l'écriture en écrivant d'abord la nouvelle valeur puis en effaçant ensuite l'écriture précédente.
3. En déduire qu'il ne faut pas toujours effacer les précédentes écritures. Comment réalise-t-on une écriture?
4. Implémenter la classe RegInt en utilisant un tableau de AtomicBoolean

Exercice 2.— On considère un objet TS. Il est définit par un état interne c initialisé à 1. Il n'a qu'une seule méthode ts() qui retourne un entier. Sa spécification séquentielle est :
{c=1} TS.ts(){c=0; return 1;}
{c=0} TS.ts(){return 0;}

1. Donner un exemple d'exécution de 4 threads partageant un objet TS et faisant chacune 2 appels a ts() de cet objet.
2. A l'aide de synchronized donnez une implémentation linéarisable de cet objet
3. En utilisant un AtomicBoolean du package java.util.concurrent.atomic, implémenter un objet TS
4. En utilisant un AtomicInteger du package java.util.concurrent.atomic, implémenter un objet TS

Exercice 3.— On considère un objet Compteur. Il est définit par un état interne c initialisé à 0. Il n'a qu'une seule méthode add() qui retourne un entier. Sa spécification séquentielle est :
{c=x} compteur.add(){c=x+1; return x;}

1. Donner un exemple d'exécution de 4 threads partageant un objet compteur et faisant chacune 2 appels add() de cet objet.
2. On considère l'implémentation suivante

```
public class Compteur {
     int c=0;
     int add(){ c=c+1; return c-1;}

   }
```

Cette implémentation est-ellle linéarisable ?

programme de test :

```
public class AjoutThread3 extends Thread{
    public Compteur p;
    int id;
    public AjoutThread3 ( Compteur p, int id){
        this.p=p;
        this.id=id;

    }
    public void run(){
        for(int i=1;i<11;i++)
        {
         System.out.println( p.add());
          try{Thread.sleep(10);}
          catch(Exception e) { System.out.println( "probleme" );}
          this.yield();
          }

    }
}
==============
public class Main3  {
     public static void main( String [] v)
    {

    Thread[] Th = new Thread[3];
    compteur f=new Compteur();

    for (int i=0;i<3;i++)
    {
         Th[i]= new AjoutThread3(f,i);
         Th[i].start();
    }
   for (int i=0;i<3;i++)
   {    try{Th[i].join();}
         catch(Exception e) {System.out.println( "probleme" );}
   }
 /*    for ( int i=0; i<34;i++) {
              System.out.println(cur.get().value+ " ");
              cur = cur.get().next;
    }  */
}

}
```

3. On considère l'implémentation suivante

```
public class Compteur {
     TS t[]= new TS[100];
     compteur(){
         for(int i=0;i<100; i++) t[i]=new TS();}
     int add(){
         int i=0;
         while( t[i].ts()==0)i++;
         return i;}

}
```

Cette implémentation est-elle linéarisable (on suppose qu'il y a moins de 100 appels à un objet compteur) ?