Faculté SCIENCES ET TECHNIQUES Le Mans

Master STS Mention Informatique Parcours ISI
Unité 178EN003
Programmation distribuée

TD1

Ce premier TD porte sur les threads et les problèmes de synchronisation en Java.
Après avoir étudié l'annexe, résolvez les exercices suivants :

Exercice n°1 :

```
class Compteur extends Thread {
  static final int NITER = 1000000;
  static int compteur = 0;

  public void run() {
    for (int i=0; i<NITER; i++) {
      compteur++;
    }
  }
}

class CompteurTest {
  public static void main(String[] args) throws InterruptedException {

    // Nombre de threads, par défaut 4
    int nbThreads = (args.length < 1) ? 4 : Integer.parseInt(args[0]);
    Compteur[] tc = new Compteur[nbThreads];

    // Crée et démarre les threads
    for (int i = 0; i < nbThreads; i++) {
      tc[i] = new Compteur();
      tc[i].start();
    }

    // Attends que les threads aient terminé
    for (int i = 0; i < nbThreads; i++)
      tc[i].join();

    // Compare la valeur de compteur avec la valeur attendue
    if (Compteur.compteur != Compteur.NITER*nbThreads)
      System.out.println("Erreur ! compteur = " + Compteur.compteur);
    else
      System.out.println("OK ! compteur = " + Compteur.compteur);
  }
}
```

Étudiez le programme précédent. Donne-t-il toujours le résultat attendu ? Si non, corrigez-le pour qu'il fonctionne toujours correctement.

Exercice n°2 :

Ecrire un calcul de la fonction *f* de Fibonacci, définie par la récurrence :

- $f(0)=0$;
- $f(1)=1$;
- $f(2)=1$;

Fibonacci

- $f(0)=0$;
- $f(1)=1$;
- $f(2)=1$;
- $f(n+2)=f(n+1)+f(n)$;

en utilisant des threads JAVA. L'idée est de paralléliser l'algorithme récursif naturel permettant de calculer *f*. Mais pour calculer $f(n+2)$, au lieu de s'appeler soi-même deux fois pour calculer $f(n+1)$ et $f(n)$, on créera un thread pour calculer $f(n+1)$ et un autre pour calculer $f(n)$. Ainsi, on définira une classe *Fibonnacci* (étendant la classe *Thread*) dont la méthode *run*() sera en charge de calculer la fonction *f*.

Exercice n°3 :

```
/*
  Objectif :

  Imprimer la chaîne "Exemple multithread simple" avec 3 threads.
  Le thread qui écrit "multithread " doit attendre le thread qui
  écrit "Exemple " et le thread associé à main() doit attendre le thread
  qui écrit "multithread ".

  Description :

  La fonction main() crée 2 threads et les démarre dans un ordre qui doit
  pouvoir être quelconque, l'un pour imprimer la chaîne "Exemple ",
  l'autre pour imprimer la chaîne "multithread ". Le thread principal
  imprime lui la chaîne "simple".

*/

class Exemple extends Thread {
  public void run() {
    System.out.print("Exemple ");
  }
}

class Multithread extends Thread {
  public void run() {
    System.out.print("multithread ");
  }
}

class MT2 {
  public static void main(String[] args){
    System.out.println("simple");
  }
}
```

Complétez le programme précédent pour afficher `Exemple Multithread Simple` en utilisant trois threads, chacun écrivant un mot. La synchronisation des threads doit être obtenue en utilisant la méthode `join()`.

Exercice n°4 :

Refaire l'exercice précédent mais, cette fois, la synchronisation des threads doit être obtenue à l'aide du couple de méthode `wait()/notify()` et de conditions.