TP 2 : Exclusion Mutuelle & Thread Java

1 Wait ou Yield ?

A.1 Mettez 100 threads en attente en utilisant *wait*(). Regardez les ressources utilisée. Faite de même avec *yield*() et une boucle et comparez l'utilisation des ressources avec wait().

Que constatez vous ? Quel conclusion pouvez vous en tirer.

2 Exclusion Mutuelle

Voici un code :

```
public class Shared{
  volatile int var=0;
  public int getVar(){
    return var;
  }
  public void setVar(int var){
    this.var=var;
  }
}
```

```
import java.util.Random;


public class Calcul implements Runnable {

    static Shared shared;
    static Random rand = new Random();

    public static void main(String[] args) {
        Thread[] th = new Thread[100];
        shared = new Shared();
        for (int a = 0; a < th.length; a++) {
            th[a] = new Thread(new Calcul(a));
            th[a].start();
        }
        for (int a = 0; a < th.length; a++) {
            try {
                th[a].join();
            } catch (Exception ex) {
            }
        }
        System.out.println("Le resultat est : " + shared.getVar());
    }
    int var;

    Calcul(int var) {
        this.var = var;
    }

    public void run() {
```

```
        try {
            System.out.println("depar de " + var);
            int tmp = shared.getVar();
            /* Simulation d'attente d'une reponse d'un peripherique
                en fonction de la valeur en memoire partagee
             */
            Thread.sleep(rand.nextInt(50));
            tmp = tmp + var;

            shared.setVar(tmp);
            System.out.println("arret de " + var);
        } catch (InterruptedException ex) {
        }

    }
}
```

B.1 Que constatez-vous après plusieurs executions ?

B.2 Comment protèger le calcul des interférences venant d'autres thread ?

B.3 Implémentez votre méthode et testez là.

3 DeadLock

Voici un code :

```
run () {/*Thread 1*/
  GetLock(Object1);
  GetLock(Object2);
  ReleaseLock(Object2);
  ReleaseLock(Object1);

}
run () {/*Thread 2*/
  GetLock(Object2);
  GetLock(Object1);
  ReleaseLock(Object1);
  ReleaseLock(Object2);

}
```

C.1 L'execution du programme fini-t-elle ? Expliquez pourquoi.