TP : Java - Processus légers

Le code retour d'un programme doit être 0 si le programme se déroule normalement sinon le code retour est un entier positif (distinct selon le problème).
Utilisez la méthode statique `System.exit(int).`

Le site http://java.sun.com/javase/6/docs/api/ contient la description complétée des classes (avec leurs méthodes) de Java SE 1.6.

Vos programmes doivent être stockés `$HOME/ProgConc/TP` (répertoire à créer).

Une variable partagée par plusieurs processus légers est une variable dont la valeur peut être lue et modifiée par plusieurs processus légers.

1. Programmez la classe `Cpt`.
- **a.** Indiquez le nombre maximun de processus légers co-existants au cours de l'exécution de `Cpt.`
- **b.** Indiquez le nombre d'incrémentation de la variable global `_cpt,` par chaque processus léger.
- **c.** Qu'elle doit être la valeur affichée à la fin de l'exécution du code ?
- **d.** Expliquez les résultats obtenus lors de l'exécution de `java Cpt` (faire plusieurs exécutions).

```
public class Cpt extends Thread {
   private static int _cpt = 1;
   private String _name;

   public Cpt(String name) { _name = name; }
   public static int getValeur() {return _cpt;}

   public void run() {
          for (int i = 1; i <= 100000; i++) {
                int _c = _cpt;  _cpt = _c+1;       }
  }

  public static void main(String args[]) {
      System.out.println("VALEUR "+Cpt.getValeur());
      Cpt thr1 = new Cpt("Processus1");
      Cpt thr2 = new Cpt("Processus2");
      thr1.start();
      thr2.start();
      try { thr1.join(); thr2.join(); }
      catch (Exception e) {System.out.println(e);}
      System.out.println("VALEUR "+Cpt.getValeur());    }
}
```

2. Le code de la classe `Cptb` est identique au code de `Cpt` à l'exception de la méthode `main`, donnée ci-dessous.
- **a.** Quelle est la différence entre `thr1.run()` et `thr1.start()`?
- **b.** Indiquez le nombre maximun de processus légers co-existants au cours de l'exécution de `Cptb.`
- **c.** Indiquez la variable partagée par `thr1` et `thr2.`

d. Indiquez le nombre d'incrémentation de la variable global `_cpt,` par chaque processus léger.

e. Qu'elle doit être la valeur affichée à la fin de l'exécution du code ?

f. Expliquez les résultats obtenus lors de l'exécution de `java Cptb` (faire plusieurs exécutions).

```
public static void main(String args[]) {
     Cpt thr1 = new Cpt("Processus1");
     Cpt thr2 = new Cpt("Processus2");
      thr1.run();
      thr2.start();
       try { thr2.join(); } catch (Exception e) {System.out.println(e);}
      System.out.println("VALEUR "+Cpt.getValeur());      }
```

3. Ecrivez en Java un programme P2 qui crée trois processus légers. Chaque processus léger exécute le code suivant (`CoureurEgoiste`) .

```
public void run() {
    while (tick < 40000000) {
        tick++;
        // tick est un entier initialisé à 0
        // tick représente le nombre de millisecondes écoulées depuis
        //       le début de l'exécution du CoureurEgoiste
        if ((tick % 5000) == 0) {
            System.out.print("Thread #" + Thread.currentThread().getName()+
                        " tick = " + tick);
        }  }   }
```

4. Complétez le programme P2 (P2b) pour afficher le nombre de processus légers (thread) créés durant l'exécution de ce programme. Pour chaque processus léger, indiquez son nom, son groupe, sa priorité, et s'il s'agit d'un processus démon ou non.

5. Complétez le programme P2b (P2c) pour qu'il attend la terminaison des processus légers avant de terminer par les instructions : `System.out.println("fin de P2c"); System.exit(0);`

6. Changez la priorité de deux des processus légers du programme P2c, un processus à la priorité minimum et l'autre à la priorité maximum.

7. Programmez la classe `Train.` Expliquez les affichages.

```
class Train extends Thread {
   private int _vi ;
   private String _nom ;

   public Train(int v , String n) { _vi = v; _nom = n; }

   public void run () {
     System.out.println ("le train "+nom+" part" );
      try{
            Thread.sleep(vi*500) ;
            System.out.println("le train "+_nom+" roule");
            Thread.sleep(vi*500);
            System.out.println("le train "+_nom+" s'arrête");
      } catch (Exception e) { }
  }

   public static void main ( String args [ ] )  {
      Thread tgv = new Thread(new Train(10 , "TGV")) ;
      Thread corail = new Thread(new Train(20,"CORAIL"));
      tgv.start () ; corail.start ( ) ;
      System.out.println("fin du main") ;
  }
}
```

8. Donnez le diagramme de sequence du programme suivant. Utilisez trois couleurs, une pour chaque Threads.

```
import java.util.Date;

class Calculus extends Thread  {
     String _s;
     long _l;
     Calculus (long l) { _l = l ;}

     public void run() {
         Date t = new Date(_l);
         _s = new String(t.toString());
         try{ Thread.sleep(100);}
             catch (Exception e) {System.out.println(e);}
     }

     String get() { return _s; }

     public static void main (String args[]) {
            Calculus c1= new Calculus(0);
            Calculus c2= new Calculus(4560000);
            Calculus c3= new Calculus(System.currentTimeMillis());
            c1.start();
            c2.start();
            try {
                  c1.join();
                  c2.join(); }
            catch (Exception e) {System.out.println(e);}
            String s1 = c1.get();
            String s2 = c2.get();
            System.out.println(s1);
            System.out.println(s2);
            c3.start();
            System.out.println(c3.get());
     }
}
```