Université de Nice-Sophia Antipolis — POLYTECH
ELEC5-TR — 2017–2018

Examen de Langage Java

Durée : 1h
Aucun document autorisé
Mobiles interdits *Note : la qualité des commentaires, avec notamment la présence d'affirmations significatives, ainsi que les noms donnés aux variables, et la bonne indentation rentreront pour une part importante dans l'appréciation du travail.*

Un thread générateur tire, à intervalles de temps réguliers, des nombres entiers de façon aléatoire sur l'intervalle $[0; n[$. Chaque nombre tiré sera affecté, à tour de rôle, à une variable partagée.

Des threads consultants consultent la variable partagée. Si celle-ci est différente de 0, elle met le thread consultant en attente jusqu'à ce que la variable partagée passe à 0. Quand la variable partagée passe à 0, elle reveille un thread consultant endormi, qui alors s'achève après avoir écrit le message *j'ai eu mon zéro* 😀.

► 1. Écrivez les trois classes, `Générateur`, `Consultant` et `VariablePartagée` qui réalisent le système décrit précédemment.

```
/**
 * Cette classe gère l'accès à une variable partagée par différents
 * threads :
 * - un générateur qui la met à jour à l'aide de la méthode setVal
 * - des consultants qui accède à la valeur avec getZéro(). Cette
 *   méthode met en attente les threads consultants jusqu'à ce que la
 *   variable partagée devienne égale à 0
 *
 */
public class VariablePartagée {
    private int n = -1;

    public synchronized void getZéro() {
        try {
            while (this.n!=0) wait();
        }
        catch (InterruptedException e) {}
    }

    public synchronized void setVal(int n) {
        this.n = n;
        if (n==0) notifyAll();
    }
}

/**
 * Ce thread consulte une variable partagée jusqu'à ce qu'elle
 * devienne égale à 0
 *
 */
public class Consultant implements Runnable {
    Thread proc = new Thread(this);
    VariablePartagée c;
```



```
    public Consultant(VariablePartagée c) {
        this.c = c;
        this.proc.start();
    }

    public void run() {
        c.getZéro();
        System.out.print(proc.getName() + " a eu son zéro");
    }
}

import java.util.Random;
/**
 * Ce thread produit des nombres entiers tirés aléatoirement toutes
 * les 300 ms dans une variable partagée
 *
 */
public class Générateur implements Runnable {
    private Thread proc = new Thread(this);
    private VariablePartagée vp;
    private int n; //définit l'intervalle [0 ; n [

    public Générateur(VariablePartagée vp, int n) {
        this.vp = vp;
        this.n = n;
        this.proc.start();
    }
    public void run() {
        Random r = new Random();
        try {
            while (!Thread.interrupted()) {
                Thread.sleep(300);
                //affecter à la variable partagée un
                //nombre aléatoire pris dans [0 ; n [
                vp.setVal(r.nextInt(this.n));
            }
        }
        catch (InterruptedException e) {}
    }
}
```

.....

► 2. Écrivez une classe de test qui lance le système.

```
public class Test {
    public static void main(String [] args) {
        VariablePartagée vp = new VariablePartagée();
        //produire des nombres pris dans [0;10[
        //à affecter à la variable partagée vp
        new Générateur(vp, 10);
        //on crée 3 consultants qui attendent leur 0
        new Consultant(vp);
        new Consultant(vp);
        new Consultant(vp);
    }
}
```

.....