Université Bordeaux I
U.F.R. Mathématiques et Informatique

IUP MIAGe 3^{eme} année
Année 2000-2001

Mardi 24 avril, 14 heures - 16 heures
Tous documents autorisés.

Programmation Parallèle
Alain Griffault

Numéro d'anonymat :

---

1 Le non-déterminisme (5 points)

Soit le programme CSP suivant :

```
Repetition ::

// déclaration et initialisation des variables
  var A, B, C, aux : integer;

// une structure répétitive qui contient les commandes gardées.
  do [
     (A < B) --> aux = A; A = B; B = aux;
        []
     (B < C) --> aux = B; B = C; C = aux;
  ] od
```

Question 1.1 (2 points) *Complétez pour les deux exécutions de ce programme le tableau suivant :*

	Repetition		Repetition	
	entrée	sortie	entrée	sortie
A	3		5	
B	7		3	
C	5		7	

Question 1.2 (2 points) *Complétez pour ce programme le tableau suivant :*

Repetition	Oui/Non	Justification
Le programme échoue t-il ?		
Le programme termine t-il ?		

Question 1.3 (1 point) *Que fait exactement ce programme ?*

2 Le problème des lecteurs-rédacteurs (15 points)

Dans ce problème, des lecteurs consultent un objet (composé de deux attributs) modifié par des rédacteurs. **Si les lecteurs obtiennent des valeurs différentes, ils considèrent cela comme une erreur et s'arrêtent aussitôt.** Voici le source d'une première version.

Le fichier `MonObjet.java`

```
class MonObjet{
    private int a=0;
    private int b=1;

    public void setA(int v){a = v;}
    public void setB(int v){b = v;}
    public int getA(){return a;}
    public int getB(){return b;}
}
```

Le fichier `Lecteur.java`

```
class Lecteur extends Thread {
    MonObjet mo;
    int a,b;

    Lecteur(MonObjet mo) {
        this.mo = mo;
    }

    public void run() {
        do {
            a = mo.getA();
            b = mo.getB();
            System.out.print(",");
        } while (a == b);
        System.out.print("P");
    }
}
```

Le fichier `Redacteur.java`

```
import java.util.*;
class Redacteur extends Thread {
    Random alea = new Random(); // pour générer aléatoirement des entiers
    MonObjet mo;

    Redacteur(MonObjet mo) {
        this.mo = mo;
    }

    public void run() {
        while (true){
            int aux = alea.nextInt()%10;
            mo.setA(aux);
            mo.setB(aux);
            System.out.print(".");
        }
    }
}
```

Le fichier `LecteursRedacteurs.java`

```
import java.util.*;
public class LecteursRedacteurs {
    public MonObjet mo;
    public Redacteur[] redacteurs;
    public Lecteur[] lecteurs;

    LecteursRedacteurs(String[] args){
        mo = new MonObjet();
        redacteurs = new Redacteur[Integer.parseInt(args[0])];
        lecteurs = new Lecteur[Integer.parseInt(args[1])];
        for (int i=0; i<redacteurs.length; i++)
            redacteurs[i] = new Redacteur(mo);
        for (int i=0; i<lecteurs.length; i++)
            lecteurs[i] = new Lecteur(mo);
        for (int i=0; i<redacteurs.length; i++)
            redacteurs[i].start();
        for (int i=0; i<lecteurs.length; i++)
            lecteurs[i].start();
    }

    public void execution(){
        for (int i=0; i<lecteurs.length; i++) // Les lecteurs sont attendus
            try {lecteurs[i].join();}
            catch(InterruptedException ie){}
        for (int i=0; i<redacteurs.length; i++) // Les redacteurs sont arrétés
            redacteurs[i].stop();
        System.out.println(mo.getA()==mo.getB());
    }

    public static void usage () {
        System.err.println("syntaxe : LecteursRedacteurs NbRedact NbLect");
    }

    public static void main(String []args) {
        boolean pasArgument = args.length != 2;
        if (pasArgument) usage();
        else {
            System.out.println("DEBUT");
            LecteursRedacteurs lr = new LecteursRedacteurs(args);
            lr.execution();
            System.out.println("FIN");
            System.exit(1);
        }
    }
}
```

et voici une sortie possible de l'exécution : `java LecteursRedacteurs 2 2`

```
DEBUT
........................,..,P..,....................,..,..,..,..,..,..,..,..,..,
..,..,..,..,..,..,..,..,..,..,..,..,..,..,..,..,..,..,..,..,..,...,..P.........
....................true
FIN
```

Question 2.1 (2 points) *Expliquez l'exécution ayant provoqué la sortie précédente.*

Question 2.2 (2 points) *Positionnez des blocs synchronisés dans le code des deux classes suivantes afin d'éviter le problème précédent.*

Le fichier `Lecteur.java`

```
class Lecteur extends Thread {
    MonObjet mo;
    int a,b;

    Lecteur(MonObjet mo) {
        this.mo = mo;
    }

    public void run() {
        do {
            a = mo.getA();
            b = mo.getB();
            System.out.print(",");
        } while (a == b);
        System.out.print("P");
    }
}
```

Le fichier `Redacteur.java`

```
import java.util.*;
class Redacteur extends Thread {
    Random alea = new Random(); // pour générer aléatoirement des entiers
    MonObjet mo;

    Redacteur(MonObjet mo) {
        this.mo = mo;
    }

    public void run() {
        while (true){
            int aux = alea.nextInt()%10;
            mo.setA(aux);
            mo.setB(aux);
            System.out.print(".");
        }
    }
}
```

Il est probable que votre solution puisse produire la sortie suivante pour l'exécution : `java LecteursRedacteurs 2 2`

```
DEBUT
,,P....................P..................................true
FIN
```

Question 2.3 (2 points) *Expliquez l'exécution ayant provoqué cette nouvelle sortie. Quel est le défaut qui subsiste ?*

Pour corriger le défaut, on se propose d'ajouter un moniteur dans la classe `MonObjet.java`. Les contraintes sont les suivantes :
- un lecteur devra demander l'autorisation de lire, puis signalera lorsqu'il a fini de lire,
- de même, un rédacteur demandera l'autorisation d'écrire, puis signalera lorsqu'il aura terminé,
- une lecture ne sera possible que si au moins une écriture a déjà eu lieu,
- plusieurs lecteurs pourront accédés simultanément aux objets a et b,
- plusieurs rédacteurs ne pourront pas accédés simultanément aux objets a et b.

Le fichier `Lecteur.java` devient

```
class Lecteur extends Thread {
    MonObjet mo;
    int a,b;

    Lecteur(MonObjet mo) {
        this.mo = mo;
    }

    public void run() {
        do {
            mo.debutLire();
            a = mo.getA();
            b = mo.getB();
            mo.finLire();
            System.out.print(",");
        } while (a == b);
        System.out.print("P");
    }
}
```

Le fichier `Redacteur.java` devient

```
import java.util.*;
class Redacteur extends Thread {
    Random alea = new Random(); // pour générer aléatoirement des entiers
    MonObjet mo;

    Redacteur(MonObjet mo) {
        this.mo = mo;
    }

    public void run() {
        while (true){
            int aux = alea.nextInt()%10;
            mo.debutEcrire();
            mo.setA(aux);
            mo.setB(aux);
            mo.finEcrire();
            System.out.print(".");
        }
    }
}
```

Question 2.4 (4 points) *Complétez le code des méthodes* `debutEcrire()` *et* `finEcrire()` *de la classe* `MonObjet.java` *afin que toutes les contraintes soient respectées.*

```
class MonObjet{
    private int a=0;
    private int b=1;
    private boolean ecrire = false;
    private boolean initialise = false;
    private int nbLecteurs = 0;

    public void setA(int v){a = v;}
    public void setB(int v){b = v;}
    public int getA(){return a;}
    public int getB(){return b;}

    public synchronized void debutLire(){
        while (!initialise){
            System.out.println("attend initialisation");
            try {wait();}
            catch(InterruptedException e){}
        }
        nbLecteurs++;
        while (ecrire){
            System.out.println("attend fin ecriture");
            try {wait();}
            catch(InterruptedException e){}
            notifyAll();
        }
    }

    public synchronized void finLire(){
        nbLecteurs--;
        notifyAll();
    }

    public synchronized void debutEcrire(){









    }

    public synchronized void finEcrire(){





    }

}
```

Lors des compilations des différentes versions, le message suivant apparaît à l'écran :

```
Note: LecteursRedacteurs.java uses or overrides a deprecated API.
Note: Recompile with -deprecation for details.
```

Et voici un extrait de la documentation Java :

Why is `Thread.stop` deprecated ?

Because it is inherently unsafe. Stopping a thread causes it to unlock all the monitors that it has locked. If any of the objects previously protected by these monitors were in an inconsistent state, other threads may now view these objects in an inconsistent state. Such objects are said to be damaged. When threads operate on damaged objects, arbitrary behavior can result. This bahavior may be subtle and difficult to detect.

What should I use instead of `Thread.stop`

Most uses of `stop` should be replaced by code that simply modifies some variable to indicate that the target thread should stop running. The target thread should check this variable regularly, and return from its run method in an orderly fashion if the variable indicates that it is to stop running.
For example, suppose your applet contains the following `start`, `stop` and `run` methods :

```
private Thread blinker;
public void start() {
    blinker = new Thread(this);
    blinker.start();
}
public void stop() {
    blinker.stop();    // UNSAFE!
}
public void run() {
    Thread thisThread = Thread.currentThread():
    while(true) {
        try {thisThread.sleep(interval);}
        catch (InterruptedException e) {}
        repaint();
    }
}
```

You can avoid the use of `Thread.stop` by replacing the applet's `stop` and `run` methods with :

```
public void stop() {
    synchronized(blinker){blinker = null;}
}
public void run() {
    Thread thisThread = Thread.currentThread():
    synchronized(blinker){
        while(blinker == thisThread) {
            try {thisThread.sleep(interval);}
            catch (InterruptedException e) {}
            repaint();
        }
    }
}
```

Question 2.5 (2 points) *La sortie suivante est-elle possible dans la version sans moniteur ? Justifiez.*

```
DEBUT
,,P....................P.....................................false
FIN
```

Question 2.6 (2 points) *En utilisant la technique décrite dans la documentation, modifiez la méthode* `execution()` *de la classe* `LecteursRedacteurs.java`*, et la méthode* `run()` *de la classe* `Redacteur.java`*.*

```
public void execution(){
    for (int i=0; i<lecteurs.length; i++) // Les lecteurs sont attendus
        try {lecteurs[i].join();}
        catch(InterruptedException ie){}
    for (int i=0; i<redacteurs.length; i++) // Les redacteurs sont arrétés


    System.out.println(mo.getA()==mo.getB());
}
public void run() {



        int aux = alea.nextInt()%10;
        mo.debutEcrire();
        mo.setA(aux);
        mo.setB(aux);
        mo.finEcrire();
        System.out.print(".");
    }
}
```

Question 2.7 (1 point) *Dans la version avec moniteur, quelle est l'utilité de la correction précédente ?*