

Section critique et Deadlock

Exercice 1 - Modification de plusieurs variables en concurrence

On souhaite créer deux threads qui change les coordonnées d'un même point

```
public class Point {
    private int x;
    private int y;

    public void move(int x,int y) {
        this.x=x;
        this.y=y;
    }

    @Override public String toString() {
        return "("+x+", "+y+")";
    }

    public static void main(String[] args) {
        final Point p=new Point();

        for(int i=0;i<2;i++) {
            final int id=i;
            new Thread(new Runnable() {
                public void run() {
                    for(;;) {
                        p.move(id,id);
                        System.out.println(p);
                    }
                }
            }).start();
        }
    }
}
```

- 1 Vérifier en utilisant grep que le code affiche des points (0,1) ou (1,0). Expliquer pourquoi.
- 2 Créer des sections critiques en utilisant un bloc synchronized là où il faut pour que seuls des points (0,0) et (1,1) puisse être affichés.
- 3 Expliquer pourquoi il n'est pas conseillé de synchroniser sur this et modifier votre code en conséquence.
- 4 Pourquoi les champs x et y n'ont pas besoin d'être déclarés volatile.

Exercice 2 - Section critique et Verrou

- 1 Expliquer dans quels cas il est préférable d'utiliser un lock ou un bloc synchronized.
- 2 Créer une classe LockedPoint ayant la même sémantique de la classe Point de l'exercice précédent mais en utilisant un verrou de la classe `java.util.concurrent.locks.ReentrantLock`

Exercice 3 - Interblocage

Voici un code simplifié d'un code fourni par un étudiant :

```
public class Oups {
```

```

private int value;
public void setValue(int value) {
    synchronized(readLock) {
        synchronized(writeLock) {
            this.value=value;
        }
    }
}
public int getValue() {
    synchronized(readLock) {
        return value;
    }
}
public void performs() throws InterruptedException {
    Thread t=new Thread() {
        @Override public void run() {
            setValue(12);
        }
    };
    synchronized(writeLock) {
        t.start();
        Thread.sleep(1000);
        System.out.println(getValue());
    }
}
private final Object readLock=new Object();
private final Object writeLock=new Object();
public static void main(String[] args) throws
InterruptedException {
    Oups oups=new Oups();
    oups.test.perform();
}
}

```

Exécuter le code ci-dessus et expliquer.

Où se situe l'interblocage ?

La machine virtuelle hotspot possède un mécanisme de détection des interblocages taper lors de l'exécution Ctrl+\ sous linux (resp. Ctrl+Break sous windows) et vérifier avec ce que vous aviez pronostiqué.

Exercice 4 - Interblocage plou dur

Où se situe l'interblocage dans le code ci-dessous :

```

public class FunnyDeadLock {
    static synchronized void m() {
        System.out.println("hello");
    }
    static {
        Thread t=new Thread() {
            @Override public synchronized void run() {
                m();
            }
        };
        t.start();
    }
    try {

```

```
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    synchronized(t) {
        System.out.println("hello 2");
    }
}
```