Ecole Polytechnique

# Cours : “Parallélisme”
# Travaux dirigés

E. Goubault & S. Putot & O. Bouissou

# TD 1

18 janvier 2006

## 1 Démarrage de threads simples

**Question** Faire un programme JAVA qui démarre deux threads:

- Le premier qui attend 1 seconde avant de remplir un entier en mémoire partagée avec la valeur 42,
- Le deuxième qui examine en boucle la valeur de cet entier et attend que cette valeur devienne égale à 42.

Pour ce faire, il faut créer deux classes (une par processus) qui chacune sont des sous-classes de Thread, et une autre classe qui contient un main lançant les deux processus. L'entier en mémoire partagée sera en fait une “classe enveloppante” pour les entiers, on pourra utiliser par exemple la classe suivante:

```
public class UnEntier {
    int val;

    public UnEntier(int x) {
        val = x;
    }

    public int intValue() {
        return val;
    }

    public void setValue(int x) {
        val = x;
    }
}
```

## 2 Threads récursifs: Fibonacci

**Question** Ecrire un calcul de la fonction $f$ de Fibonacci, définie par la récurrence,

- $f(0) = 1$,
- $f(1) = 1$,
- $f(n+2) = f(n+1) + f(n)$.

en utilisant des threads JAVA. L'idée est de paralléliser l'algorithme récursif naturel permettant de calculer $f$ (que l'on pourra écrire dans un premier temps si l'on ne se sent plus très à l'aise avec JAVA). Mais pour calculer $f(n+2)$, au lieu de s'appeler soi-même deux fois pour calculer $f(n+1)$ et $f(n)$, on créera un thread pour calculer $f(n+1)$ et un autre pour calculer $f(n)$. Ainsi, on définira une classe,

```
public class Fibon extends Thread {
...
    public void run() {
    ...
    }
}
```

dont la méthode run() sera en charge de calculer la fonction $f$. Pour ce faire il faut que la classe Fibon contienne un champ argument, et un champ résultat, ce dernier pouvant "survivre" dans la mémoire partagée au thread qui le calcule. Ainsi nous devons utiliser une "classe enveloppante" pour le résultat. On pourra utiliser pour ce faire la classe UnEntier vue précedemment.

On programmera donc,

- un constructeur Fibon(int x, UnEntier intref),
- le main de la classe: public static void main(String[] args),
- et enfin la méthode public void run() du thread.

## 3 Crible d'Erathostène

**Question** Ecrire un programme JAVA "par passage de messages" affichant la suite des nombres premiers en utilisant la méthode du crible : un processus est chargé de générer les entiers naturels, dont on élimine d'abord les multiples de 2, puis 3, 5, etc. au moyen de processus filtrants successifs. On utilisera les deux classes suivantes:

```
import java.util.*;

public class MsgQueue {
    Vector queue = new Vector();

    public synchronized void send(Object obj) {
        queue.addElement(obj);
    }

    public synchronized Object recv() {
        if (queue.size() == 0)
            return null;
        Object obj = queue.firstElement();
        queue.removeElementAt(0);
        return obj;
    }
}
```

Et:

```
public class Process {
    MsgQueue In;
```

```
    MsgQueue Out;

    public Process(MsgQueue i, MsgQueue o) {
        In = i;
        Out = o;
    }

    public void send(int x) {
        Out.send(new Integer(x));
// System.out.println(Thread.currentThread().getName()+": send("+x+")");
    }

    public int recv() {
        Object x = In.recv();
        while (x == null)
            x = In.recv();
        int res = ((Integer) x).intValue();
// System.out.println(Thread.currentThread().getName()+": recv "+res);
        return res;
    }
}
```

## 4 Calcul de $\pi$ en parallèle

Implémenter en utilisant des threads JAVA le calcul de $\pi$ en parallèle par la formule suivante,

$$\begin{aligned} \pi &= \int_0^1 \frac{4}{1+x^2} dx \\ &\cong \sum_{i=1}^{n} \frac{1}{n} \frac{4}{1+\left((i-\frac{1}{2})\frac{1}{n}\right)^2} \end{aligned}$$

Une solution est le paradigme **Maître/Esclave**: Un maître va lancer `N` esclaves chargés de calculer les sommes partielles,

$$P_k = \sum_{i=k*n/N+1}^{(k+1)*n/N} \frac{1}{n} \frac{4}{1+\left((i-\frac{1}{2})\frac{1}{n}\right)^2}$$

pour $k = 0, \cdots,$ `N-1`.

## 5 Tri de Hoare

**Question** Programmer le *quicksort* d'une liste d'entiers en essayant de paralléliser l'algorithme séquentiel. Que gagne-t'on en complexité (moyenne, pire cas ?)