

TP 4 : Java RMI

1 Mise au point

```
class Test implements Runnable{
    int id;
    static int cpt=0;
    public void run(){
        this.inside();
    }
    synchronized void inside(){
        try{
            cpt++;
            System.out.println("in " + id+" cpt : "+cpt);
            Thread.sleep(2000);
            cpt--;
            System.out.println("out " + id);
        }catch(Exception ex){
        }
    }
    Test(int i){
        id=i;
    }
    public static void main(String arg[]){
        Thread t1=new Thread(new Test(1));
        Thread t2=new Thread(new Test(2));
        t1.start();
        t2.start();
    }
}
```

- A.1 Quelles sont les valeurs possibles de CPT ?
- A.2 Comment peut-on assurer l'exclusion mutuelle dans la fonction inside ?
- A.3 Modifiez le programme en conséquences.

2 Exemple : Serveur écho

- B.1 Téléchargez et compilez les fichiers d'exemples du cours. (www.lif.univ-mrs.fr/~martin)
- B.2 Compilez et testez.

Remarque : Par défaut, pour le registre rmi, l'adresse du serveur de l'application est 127.0.1.1. Pour changer cela vous devez mettre l'option suivante lorsque vous le lancez :

`{"-Djava.rmi.server.hostname=$HOSTNAME"}`.

Si vous avez un problème de sécurité vous pouvez ajouter : {"-Djava.security.policy=java.policy"}.

3 Calcul distribué

Le but de cet exercice est de calculer les décimales de π de façon distribuée.
En 1995 Bailey, Borwein et Plouffe ont montré la relation suivante :

$$\pi = \sum_{j=0}^n \frac{1}{16^k} \left(\frac{4}{8j+1} - \frac{2}{8j+4} - \frac{1}{8j+5} - \frac{1}{8j+6} \right)$$

Voici le un programme séquentiel :

```
import java.math.*;
public class CalculDePI {
    public String calcul (int nbDecimal){
        BigDecimal somme = BigDecimal.ZERO;
        for (int j=0; j<nbDecimal; j++){
            BigDecimal part1 = new BigDecimal(4.0/(8*j+1));
            BigDecimal part2 = new BigDecimal(2.0/(8*j+4));
            BigDecimal part3 = new BigDecimal(1.0/(8*j+5));
            BigDecimal part4 = new BigDecimal(1.0/(8*j+6));
            BigInteger part5 = BigInteger.valueOf(16).pow (j);
            part1 = part1.subtract(part2);
            part1 = part1.subtract(part3);
            part1 = part1.subtract(part4);
            part1 = part1.divide (new BigDecimal(part5));
            somme = somme.add (part1);
        }
        // On ne retourne que les decimales demandees car le reste
        // est faux
        BigDecimal nbd = new BigDecimal (10.0).pow (nbDecimal);
        // On multiplie par 10^nbdecimal
        somme = somme.multiply (nbd);
        // On prends la partie entiere
        BigInteger tempSomme = somme.toBigInteger ();
        return ((new BigDecimal(tempSomme).divide (nbd)).toString
            ());
    }
    public static void main (String args []){
        if (args.length==0) {
            System.out.println ("Syntaxe : java CalculDePI ####");
            System.out.println ("#### represente le nom de decimal
                de PI");
        }
        else {
            CalculDePI cdpi = new CalculDePI ();
            for (int i=1; i<Integer.parseInt(args[0]); i++)
                System.out.println (cdpi.calcul (i));
        }
    }
}
```

C.1 Utilisez Java RMI pour distribuer le calcul entre plusieurs ordinateurs.

4 Déchiffrage distribué (*Exercice optionnel*)

Une fonction de hashage cryptographique a pour but de calculer la hashé d'un mot ou d'une donnée relativement facilement mais il est très difficile de retrouver la donnée initiale. La façon la plus bête de retrouver le mot initiale à partir du hashé est ce qu'on appelle une recherche brutforce. Nous allons hasher toutes les combinaisons de caractères jusqu'à arriver au résultat escompté.

Voici un mot de 5 lettres chiffré en md5.

```
byte [] secret={117, -57, -42, -67, 100, 16, 73, -69, -64, -43, -13,
80, 112, 22, 44, -5}
```

Voici comment calculer le hashé d'un mot, ici "Test".

```
java.security.MessageDigest msgDigest = java.security.
    MessageDigest.getInstance("MD5");
msgDigest.update("Test".getBytes());
byte [] digest = msgDigest.digest();
```

D.1 D'une façon similaire, distribuez la recherche du mot et devinez le.