TDM01 d'Informatique Répartie
Sockets

ASI4 - INSA Rouen
CORRECTION

Calculatrice distribuée sur entiers positifs à l'aide de Sockets

Le but de ce TP est de réaliser une calculatrice distribuée à l'aide de Sockets. Pour cela, vous respecterez les contraintes suivantes :

- Votre calculatrice ne travaillera que sur les entiers positifs.
- Elle supportera 4 types d'opération : additions, soustractions, multiplications et divisions entières.
- Le serveur recevra comme requête de la part du client une opération à réaliser sur 2 entiers, sous la forme d'un message formaté comme suit : "entier1 OPÉRATION entier2".
- Il retournera comme réponse le résultat de l'opération (un entier positif), ou un entier négatif en cas de problème (opérande ou résultat négatif ; division par 0).
- Vous travaillerez en mode connecté.
- Vous programmerez une version en C et une version en Java pour le client comme pour le serveur. Vous vérifierez le bon fonctionnement de vos programmes dans les différentes combinaisons possibles.

NB : il s'agit d'Informatique répartie donc pensez à tester votre programme avec serveur et client sur des machines différentes !

Remarques

- Vous aurez alors accès à la correction du TDM dès la fin de la séance.
- **Déposez un compte-rendu de 2 pages `TDMSockets-LOGIN.pdf` sur moodle chez TOUTES les personnes du binôme. Ce CR contiendra les informations que vous jugerez nécessaires.**
- Votre CR sera disponible pour vous lors de l'examen machine.

Correction

ServeurNonConnecte.java

```java
import java.io.*;
import java.util.*;
import java.net.*;

public class ServeurNonConnecte {

    public static void main(String[] args) throws Exception {
        InetAddress IPServeur, IPClient;
        int portServeur, portClient, e1, e2, resultat;
        DatagramSocket socket;
        byte[] requete = new byte[100], reponse;
        DatagramPacket reception, envoi;
        String operation, message = "", retour;
        StringTokenizer calcul;

        // Récupération des paramètres
        if ( args.length>=2 ) {
            IPServeur = InetAddress.getByName(args[0]);
            portServeur = Integer.parseInt(args[1]);
        }else if( args.length>=2 ){
            IPServeur = InetAddress.getByName("127.0.0.1");
            portServeur = Integer.parseInt(args[0]);
        }else {
            IPServeur = InetAddress.getByName("127.0.0.1");
            portServeur = 1024;
        }
```

```
        // Création du Socket
        socket = new DatagramSocket(portServeur, IPServeur);

        while(!message.equals("KILL")) {
            // Création du DatagramPacket et réception d'une requête
            reception = new DatagramPacket(requete, requete.length);
            socket.receive(reception);

            // Réponse
            IPClient = reception.getAddress();
            portClient = reception.getPort();
            message = new String(reception.getData(), 0, reception.
    getLength());

            if(message.equals("KILL"))
                resultat = -2;
            else {
                calcul = new StringTokenizer(message);
                e1 = Integer.parseInt(calcul.nextToken());
                operation = calcul.nextToken();
                e2 = Integer.parseInt(calcul.nextToken());
                System.out.println("(" + IPClient + ":" + portClient + ") :
    " + e1 + operation + e2);

                if( e1<=0 || e2<=0){
                    resultat = -1;
                }else if( operation.equals("+") ){
                    resultat = e1+e2;
                }else if( operation.equals("-") ){
                    resultat = e1-e2;
                }else if( operation.equals("x") ){
                    resultat = e1*e2;
                }else if( operation.equals("/") && e2!=0){
                    resultat = e1/e2;
                }else
                    resultat = -1;
            }
            retour = "" + resultat;
            reponse = retour.getBytes();
            envoi = new DatagramPacket(reponse, reponse.length, IPClient,
    portClient);
            socket.send(envoi);
        }

        //Fermeture du Socket
        socket.close();
     }
}
```

ClientNonConnecte.java

```
import java.io.*;
import java.util.*;
import java.net.*;

public class ClientNonConnecte {

     public static void main(String[] args) throws Exception {
        InetAddress adresse = InetAddress.getByName(args[0]);
        int portUDP = Integer.parseInt(args[1]);
        String calcul, texte;
        byte[] requete, reponse;
        DatagramPacket envoi, reception;
        DatagramSocket socket = new DatagramSocket();

        if(args[2].equals("KILL"))
            calcul = args[2];
        else
            calcul = args[2] + " " + args[3] + " " + args[4];
        requete = calcul.getBytes();
        reponse = new byte[1000];
        envoi = new DatagramPacket(requete, requete.length, adresse,
    portUDP);
```

```
            reception = new DatagramPacket(reponse, reponse.length);
            socket.send(envoi);
            socket.receive(reception);
            texte = new String(reception.getData(), 0, reception.getLength());
            if(Integer.parseInt(texte)>=0)
                System.out.println(calcul + " = " + texte);
            else if(Integer.parseInt(texte)==-1)
                System.out.println("Calcul impossible en entiers positifs.");
            else
                System.out.println("Serveur éteint.");
        }
    }
```