

# TP1: Sockets et RMI

## 1) *Programmation sockets*

L'objectif de cet exercice est de lister le contenu d'un répertoire distant. Ecrire un programme client envoyant au serveur le nom du répertoire à lister. Le serveur liste le répertoire, retourne le nombre de fichiers (où -1 en cas de problème) puis le nom des fichiers un à un. Le protocole de communication utilisé sera TCP et le traitement côté serveur s'effectuera dans un thread.

## 2) *Programmation RMI*

L'objectif de ce sujet est de mettre en place un serveur de discussion. Le logiciel client et le logiciel serveur seront des applications. La communication entre le client et le serveur mettra en œuvre le mécanisme des RMI.

### a) Serveur horaire

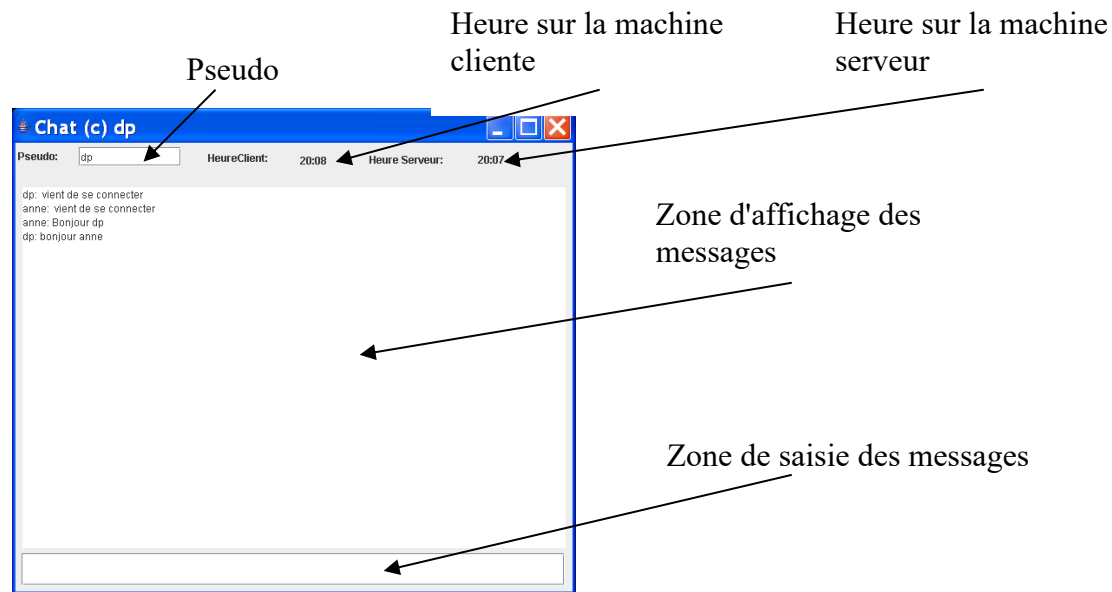
Développer un serveur capable de renvoyer la date et l'heure. On pourra utiliser les classes *GregorianCalendar*, *Calendar* incluses dans le package *java.util*. Le serveur exposera sa méthode distante dans l'interface suivante:

```
import java.rmi.* ;

public interface InterfaceHeureServeur extends Remote
{
    public String getHeure () throws RemoteException;
}
```

### b) Développement d'un logiciel de discussion

On se propose de développer un logiciel de discussion en direct basé sur un modèle client/serveur et utilisant les RMI. La topologie choisie est une étoile: tous les clients se connectent à un serveur central et chaque message tapé par un utilisateur sera envoyé, par l'intermédiaire du serveur, à tous les utilisateurs connectés. L'interface utilisateur pour les clients pourra ressembler à ceci:



Le serveur exposera les méthodes distantes suivantes:

```
import java.rmi.* ;

public interface InterfaceChatServeur extends Remote
{
    /** Connexion sur le serveur
     * @param pseudo          pseudo choisi
     * @param url              url sur l'objet client distant
     * @throws RemoteException
     */
    public void connect (String pseudo,String url) throws RemoteException ;
    /**
     *
     * @param pseudo          pseudo qui se deconnecte
     * @throws RemoteException
     */
    public void disconnect (String pseudo) throws RemoteException ;
    /**
     *
     * @param msg Message diffuse a tous les utilisateurs
     * @throws RemoteException
     */
    public void broadcastMessage (Message msg) throws RemoteException ;
}
```

La classe *Message* sera définie ainsi:

```
public class Message implements Serializable
{
    private String msg ;
    private String pseudo ;

    public Message(String pseudo,String msg)
    {
        this.pseudo = pseudo ; this.msg = msg ;
    }

    public String getMessage () { return msg ; }
    public String getPseudo () { return pseudo ; }
}
```

Question: pourquoi la classe *Message* doit-elle implémenter l'interface *java.io.Serializable* ?

Le client exposera la méthode distante suivante:

```
import java.rmi.* ;

public interface InterfaceChatClient extends Remote
{
    /**
     *
     * @param m  Message reçu de la part du serveur
     * @throws RemoteException
     */
    public void diffuseMessage (Message m) throws RemoteException ;
}
```

Remarque : tester l'application en plaçant le serveur et les clients sur des machines différentes.

L'architecture de l'application sera celle-ci:

