⑦ la classe "Client"

String url = "//Serveur1/nomService"; → nom serveur où le service fonctionne. Si il est la même machine ça sera "localhost"

Obj = (Banque) Naming.lookup(url); → nom donné pour garder l'objet à distance (pas 4)

↓ Il faut convertir au type d'objet

↳ cherche et retourner l'objet trouvé.

Obj.ouvrir("client1"); → J'utilise les méthodes de l'objet à distance.

⑧ Exécuter le "Client" (pareil au pas 6)

java -Djava.security.policy=java.policy Client

paramètre de java pour donner une politique de sécurité

fichier qui donne la politique

class (pas 7)

* RMI: Exemple Chat avec l'interface graphique.

→ J'ai pas besoin de changer le service, c'est-à-dire je ne change pas les fichiers: ChatRoom.java, chatRoomImpl.java, Serveur.java.

→ Je dois utiliser la fenêtre pour afficher le message (je vais remplacer la console par "GUIClientSwing" dans la classe ChatClientImpl.

• Si je vais utiliser la classe "GUIClientSwing", il faut créer une interface (RMI travaille avec les interfaces seulement). Par ça:

GUIClient.java

interface GUIClient { → C'est une classe de dans le dossier "chat"

void afficherMessage (Message msg);

}

• Changer la fenêtre "GUIClientSwing" pour implémenter l'interface.

public class GUIClientSwing extends JFrame implements GUIClient

{ //Attributs

...

// Constructeurs

...

// Méthode de l'interface

public void afficherMessage (Message msg)

{ _msgArea.append ("\n" + msg.toString()); } → l'affichage du message sur le composant area.

}

• J'ajoute la fenêtre à la classe "ChatClientImpl":

//Attributs

private String _nom;

private GUIClient _gui; → la fenêtre

//Constructeurs → ajouter un nouveau constructeur.

public ChatClientImpl (String name, GUIClient gui) throws RemoteException

{ _gui = gui; → On va remplacer le GUI local par l'objet qui vient par paramètre.

_nom = name; }

// On va CHANGER la méthode afficherMessage

public void afficherMessage (Message msg) throws RemoteException

{ // System.out.println(msg);

_gui.afficherMessage(msg); → On remplace la console par la fenêtre.

}

→ Fichier: Client.java

J'ai pas besoin du code à partir de la ligne 44 (BufferedReader ...) jusque la ligne 58 (System.out.pri ...). On va mettre en commentaire ces lignes parce que ces lignes utilisent la console.

Je vais utiliser la fenêtre.

• Je crée un objet de la fenêtre:

GUIClient gui = new GUIClientSwing (args[1]); → nom de l'utilisateur

// ligne 38, on va utiliser le nouveau constructeur de ChatClientImpl

cc = new ChatClientImpl (args[1], gui);

• Je dois utiliser la méthode "envoyerMessage" de ChatRoom, mais le problème est que je peux pas accéder au composant "JTextField" de la fenêtre. On va passer le objet "ChatRoom" à la fenêtre par paramètre (constructeur) ainsi la fenêtre appellera la méthode "envoyerMessage" et utilisera le composant.

* Fichier: GUIClientSwing.java

//constructeur.

public GUIClientSwing (String name, ChatRoom cr)

• Quel est le moment que j'appelle au ChatRoom pour envoyer le message? Quand je clique sur le button "envoyer" ou je tape la touche "entrée", ces deux actions sont controlés par la classe "EcouteurAction" sur la commande "envoyer".

* Fichier: EcouteurAction.java

// Attributs temporels

ChatRoom myCr; → pour remplacer créer un nouveau constructeur

actionPerformed → Commande "envoyer" → //tmpArea.append("\n" + tmpTexte.getText()); → j'ai mis en commentaire

try {

Méthode susceptible de lever RemoteException → myCr.envoyerMessage (new MessageImpl ("Client1", tmpTexte.getText()));

Je contrôle le RemoteException → } catch (RemoteException re)

{ System.out.println (re.getMessage()); }