RMI (Remote Method Invocation)

- Une API Java intégré à partir de JDK 1.1 (i.e. avant JDK2);
- Un mécanisme qui permet l'appel de méthodes entre ***objets JAVA de manière transparante***;
- L'appel peut se faire sur la même machine ou bien sur des machines connectées via un réseau;
- Permettre la communication entre plusieurs JVMs indépendamment de leurs emplacements physiques;
- Permettre la réalisation d'un système d'objets distribués constitués uniquement d'objets Java contrairement au CORBA;
- Modèle de connexion et communication basé sur les sockets;
- Les méthodes des échanges utilisent le protocole RMP (Remote Method Protocol);
- Repose sur les classe de sérialisation.

Architecture RMI

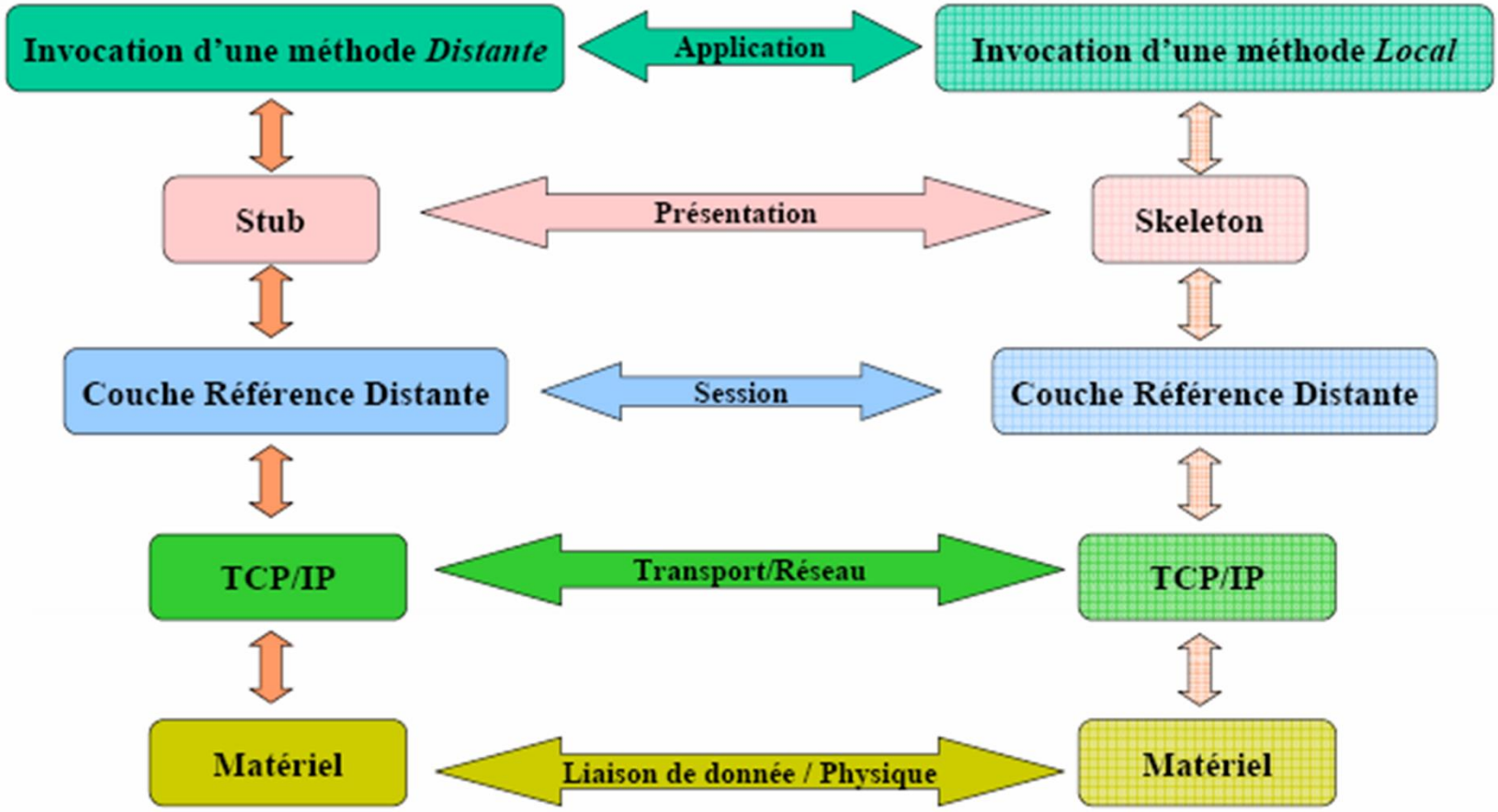

Figure 2 : Architecture RMI

Java RMI

Les amorces : Stub et Skeleton

- Adaptateurs pour le transport des appels distants;
- Réalisent les appel sur la couche réseau;
- Réalisent l'assemblage et le désassemblage des paramètres;
- Une référence d'objet distribué correspond à une référence d'amorce.

Les Stub (souches)

- Représentants locaux des objets distribués distants;
- Initient une connexion avec une JVM distante;
- Assemble les paramètres pour les transférer à la JVM distante;
- Désassemble le résultat ou l'exception retournée;
- Renvoie le résultat à l'appelant;
- S'appuient sur la sérialisation.

Java RMI

Les Skeleton (squelettes)

- Désassembler les paramètres pour la méthode distante;
- Font appel à la méthode demandée;
- Assembler le résultat renvoyé ou exception à destination de l'appelant.

Remote Reference Layer (la couche des références distants)

- Permettre l'obtention d'une référence d'objet distribué à partir de la référence locale au Stub;
- Tâche assurée par le service de noms **rmiregister** (utilise une table de hachage dont les clés sont des noms et les valeurs sont des objets distants);
- Un **rmiregister** par JVM;
- **rmiregister** accepte les demandes de service sur le port 1099.

Java RMI

- **Les étapes pour l'invocation d'une méthode d'un objet distant**

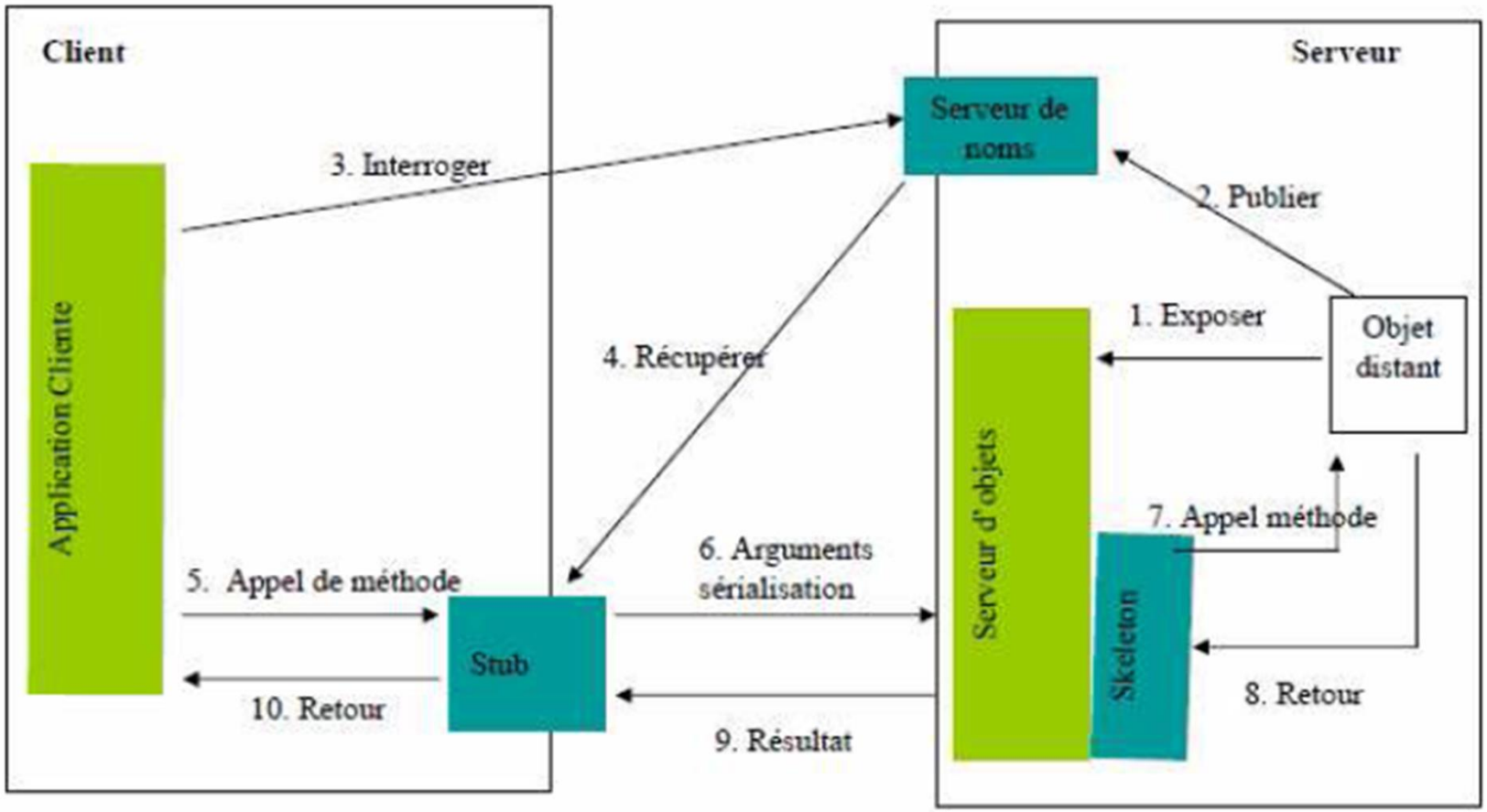

Figure 4 : Les étapes d'invocation d'une méthode distante

Java RMI

Enregistrement d'un service

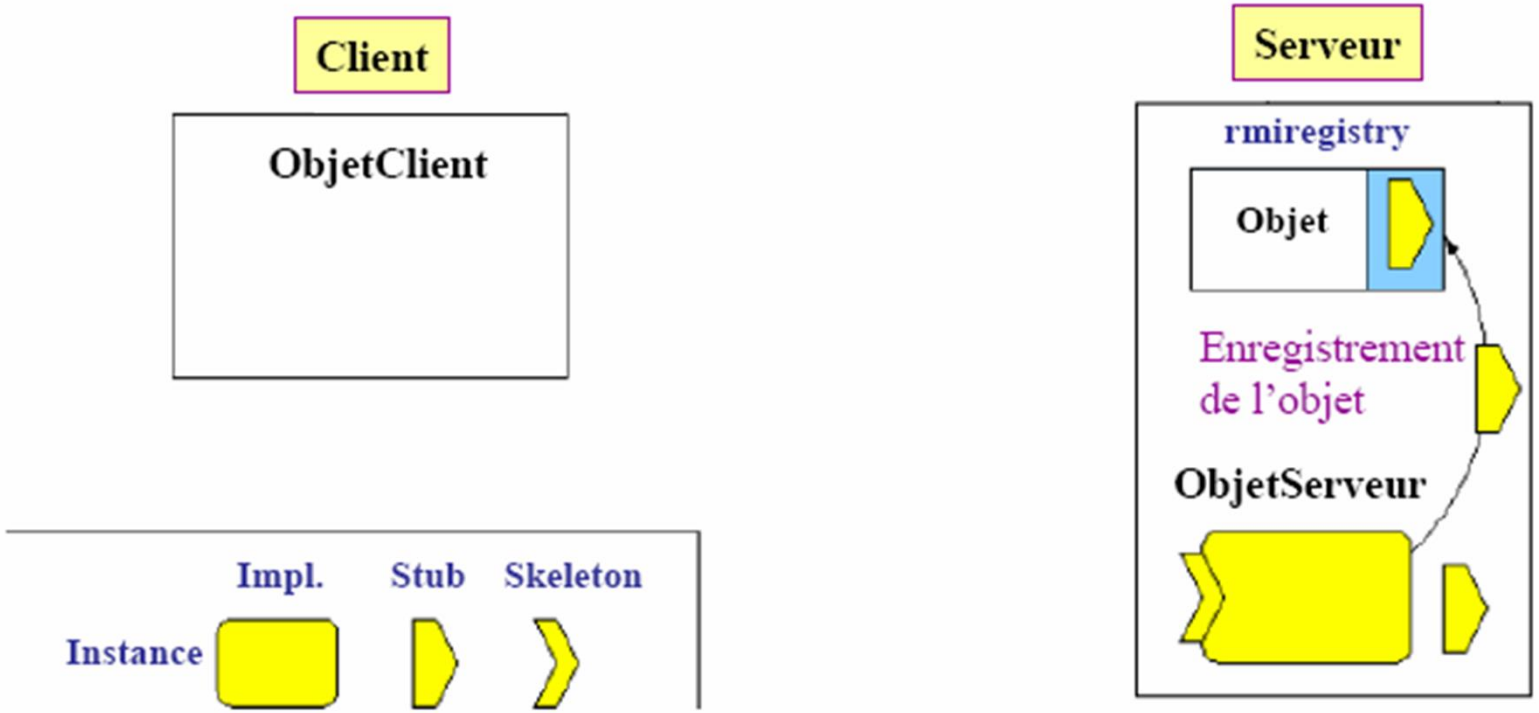

Figure 8 : Enregistrement d'un Service via rmiregistry

Java RMI

Accès à une référence distante

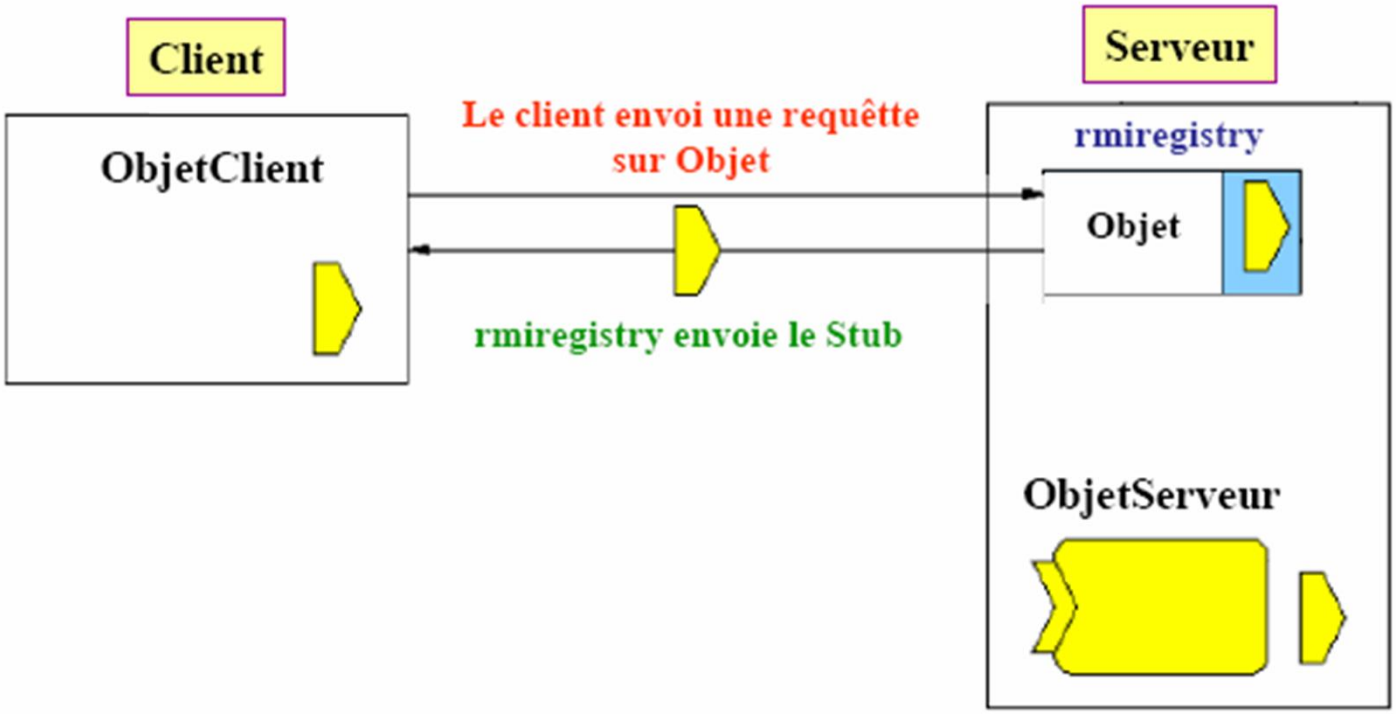

Figure 9 : Créeation de Stub (et Skeleton) d'une méthode implémenté par rmic

Java RMI

Invocation d'une méthode distante

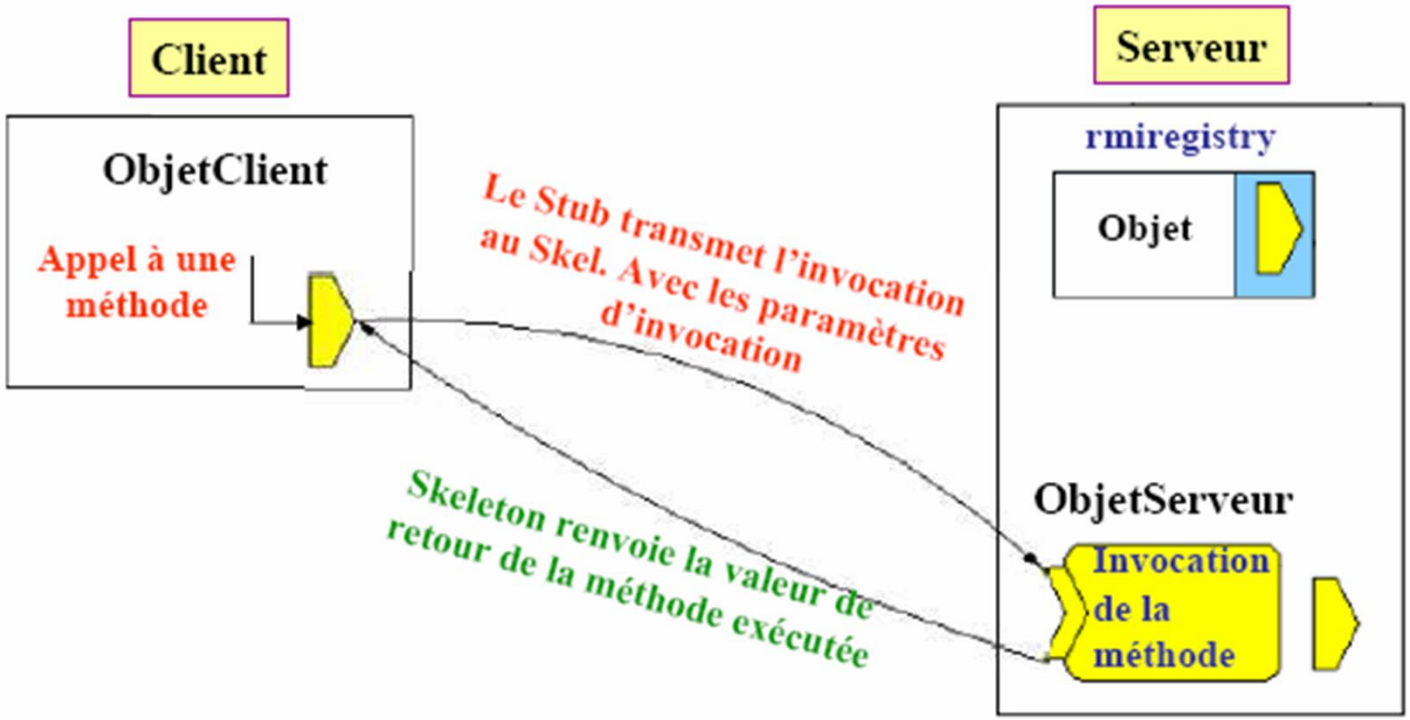

Figure10 : Invocation d'une méthode distante et récupération des résultats de retour via le Stub

Java RMI

Création et manipulation d'objets distants

- Cinq packages sont disponibles:
  - **Java.rmi** pour accéder à des objets distants;
  - **Java.rmi.server** pour créer des objets distants;
  - **Java.rmi.registry** lié à la localisation et au nommage d'objets distants;
  - **Java.rmi.dgc** ramasse-miettes pour les objets distants;
  - **Java.rmi.activation** support pour l'activation d'objets distants.

Java RMI

Étapes du développement

1. Spécifier et écrire l'interface de l'objet distant;
2. Ecrire l'implementation de cette interface;
3. Générer les stub/skeleton correspondant (commande rmic).

Étapes de l'exécution

4. Écrire le serveur qui instancier l'objet implémentant l'interface, exporte son **Stub** puis attend les requêtttes via le **Skeleton (n'est plus indispensable depuis JDK2)**;
5. Écrire le client qui réclame l'objet distant, importe le stub et invoque une méthode de l'objet distant via le **Stub**.

Java RMI

Exemple 1

Interface étend java.rmi.Remote;
Les méthodes doivent pouvoir lever java.rmi.RemoteException

Définition de l'interface d'accès aux objets distants

```
1 import java.rmi.Remote ;
2 import java.rmi.RemoteException ;
3 public interface Message extends Remote
4 {
5     public String messageDistant() throws RemoteException ;
6 }
```

Étendre une des sous-classes de java.rmi.server.RemoteServer comme java.rmi.server.UnicastRemoteObject (souvent utilisée - *offre toutes les fonctionnalités des classes distantes*).

```
1 import java.rmi.server.UnicastRemoteObject;
2 import java.rmi.RemoteException ;
3 public class MessageImpl extends UnicastRemoteObject implements Message
4 {
5     public MessageImpl () throws RemoteException {super() ;} ;
6     public String messageDistant() throws RemoteException
7     {
8         return("Message : GSTR-ENSA de Tétouan") ;
9     }
10 }
```

Doit implémenter l'interface distante Message

Java RMI

Exemple 1 (suite)

```
1  import java.net.MalformedURLException  ;
2  import java.rmi.* ;
3  public class Serveur
4  {
5      public static void main(String [] args)
6      {
7          try {
8                  MessageImpl objLocal = new MessageImpl ();
9                  Naming.rebind("rmi://localhost:1099/Message",objLocal);
10                 System.out.println("Le serveur est pret");
11         }
12         catch (RemoteException re) {System.out.println(re);}
13         catch (MalformedURLException e) { System.out.println(e) ; }
14     }
15 }
```

Le serveur crée un objet qui sera accessible à distance. Ensuite, il inscrit cet objet dans le serveur de nom en l'associant à une chaîne de caractère spécifique.

Java RMI

Exemple 1 (suite)

```
1  import java.rmi.* ;
2  import java.net.MalformedURLExcepti
3  public class Client
4  {
5      public static void main(String [] args)
6      {
7          try {
8                  Message b =(Message) Naming.lookup("//"+args[0]+"/Message");
9                  System.out.println("Le client reçoit : "+b.messageDistant());
10             }
11             catch (NotBoundException re) { System.out.println(re); }
12             catch (RemoteException re) { System.out.println(re); }
13             catch (MalformedURLException e) { System.out.println(e); }
14     }
15 }
```

Une référence sur l'objet est récupérée grâce aux services offerts par le service de nom **rmiregistry**.

Java RMI

Compilation et déploiement de l'applicatio RMI

1. javac *.java
2. rmic MessageImpl
3. rmiregistry (sur la machine serveur)
4. java Serveur (sur la machine serveur)
5. java Client <nom_machine_serveur> (sur la machine client)