

TP 1: Java RMI utilisant Eclipse

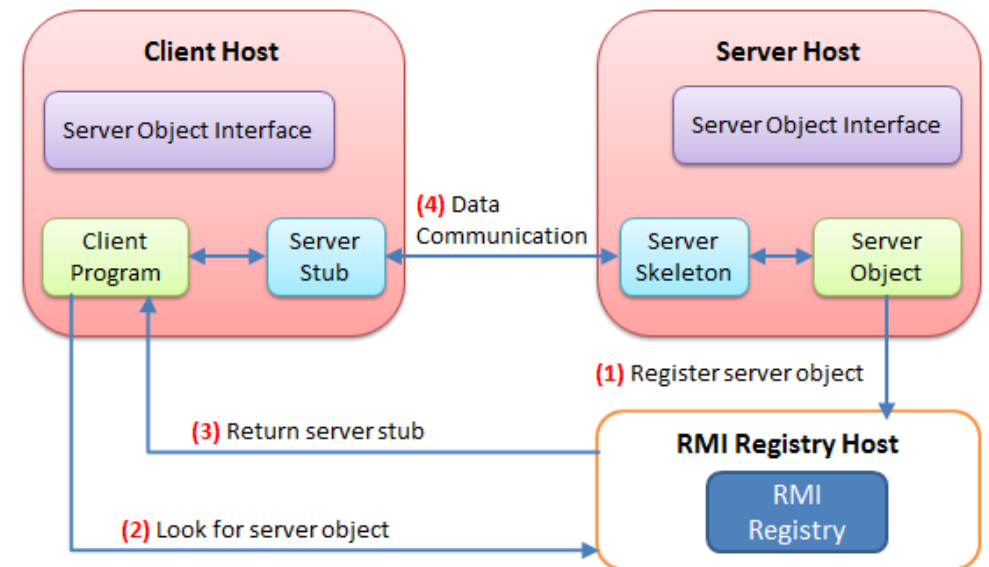
Master 2 STIC

Module : Les applications réparties

Principe du RMI

Un système basé sur Java RMI consiste principalement en trois éléments:

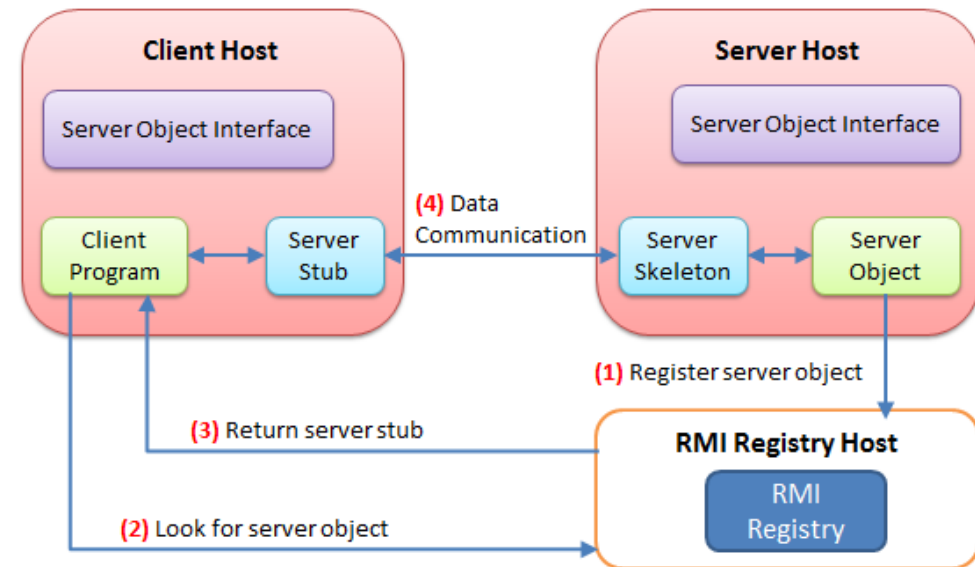
- Client - Utilise une méthode distante
- Server - Processus qui possède l'objet appelé
- Registry - Serveur de nom qui gère l'association nom-objet



Principe du RMI

La figure montre les 4 étapes nécessaires pour appeler un objet distant par Java RMI:

1. Instancier le serveur et l'enregistrer auprès du RMI Registry avec un nom unique
2. Le client recupere les information du serveur en utilisant le nom unique aurpes du RMI Registry
3. Le RMI Registry en utilisant sa base envoie au client le stub qui permettra au client de communiquer avec le server
4. Le stub prends en charge la communication avec le skeleton qui est cote serveur.



Objectifs

Dans ce TP1 de programmation RMI, vous utiliserez l'EDI Eclipse pour apprendre à créer :

- **Objet distant simple.**
- **Server pour instancier (créer) et lier un objet distant.**
- **Client pour invoquer à distance un objet**

Comme RMI est un protocole de communication Java vers Java uniquement, vous devez installer le Java Development Kit ("JDK") ainsi qu'un éditeur de programmation ("IDE") tel que Eclipse.

Les différentes étapes pour créer un objet distant et l'appeler avec RMI

Le développement coté serveur se compose de :

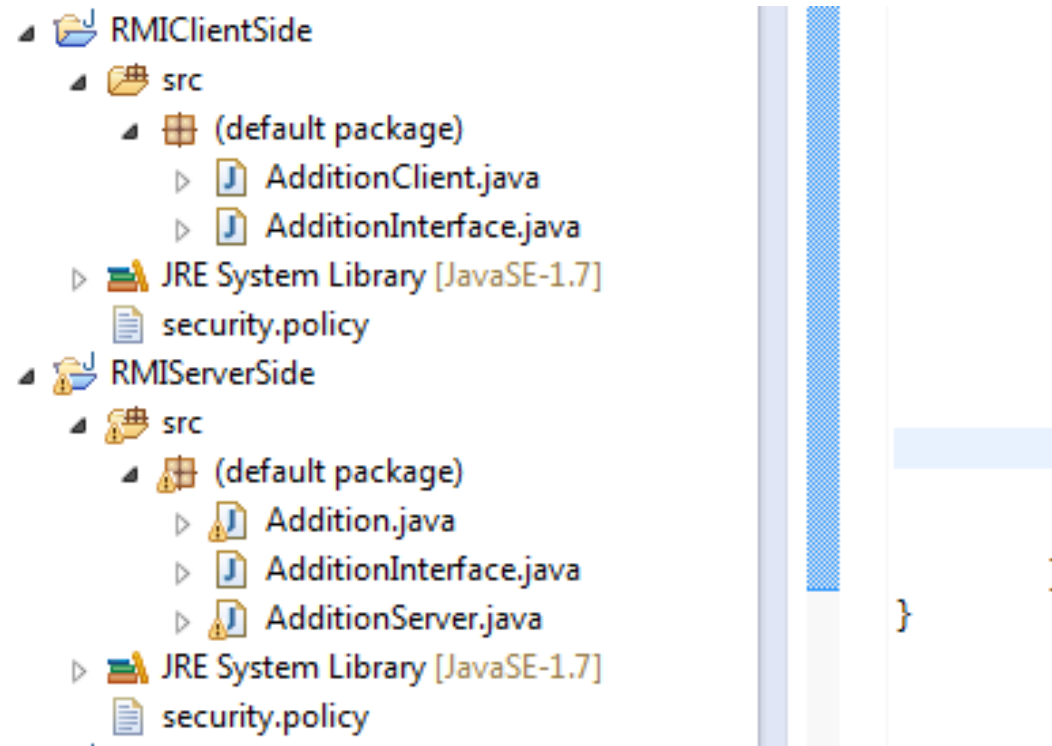
- La définition d'une interface qui contient les méthodes qui peuvent être appelées à distance
- L'écriture d'une classe qui implémente cette interface
- L'écriture d'une classe quiinstanciera l'objet et l'enregistrera en lui affectant un nom dans le registre de noms RMI (RMI Registry)

Le développement côté client se compose de :

- L'obtention d'une référence sur l'objet distant à partir de son nom
- L'appel à la méthode à partir de cette référence

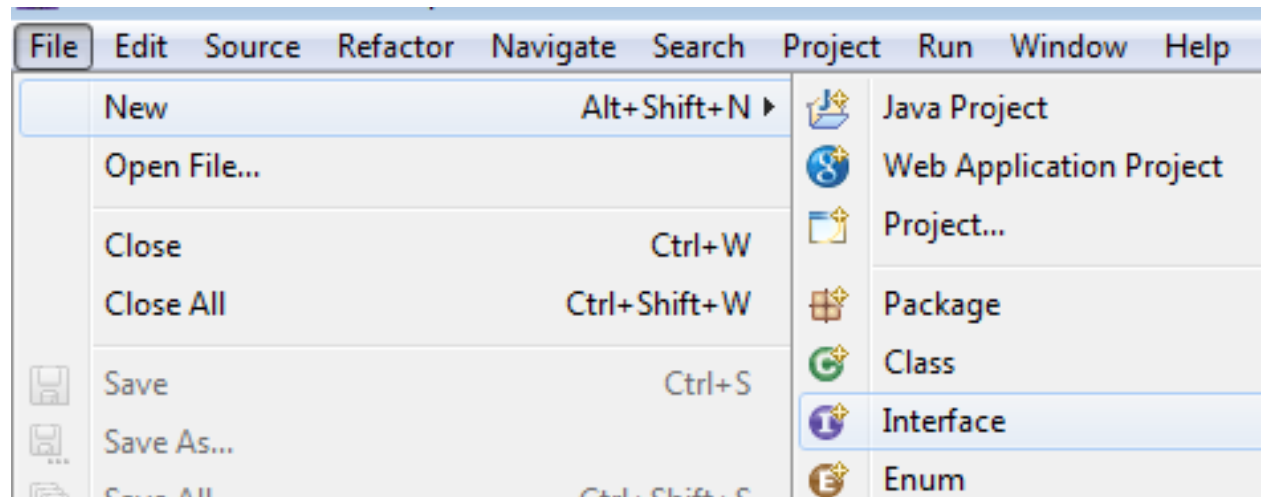
Enfin, il faut générer les classes stub et skeleton en exécutant le programme rmic avec le fichier source de l'objet distant.

La structure des fichiers des projets



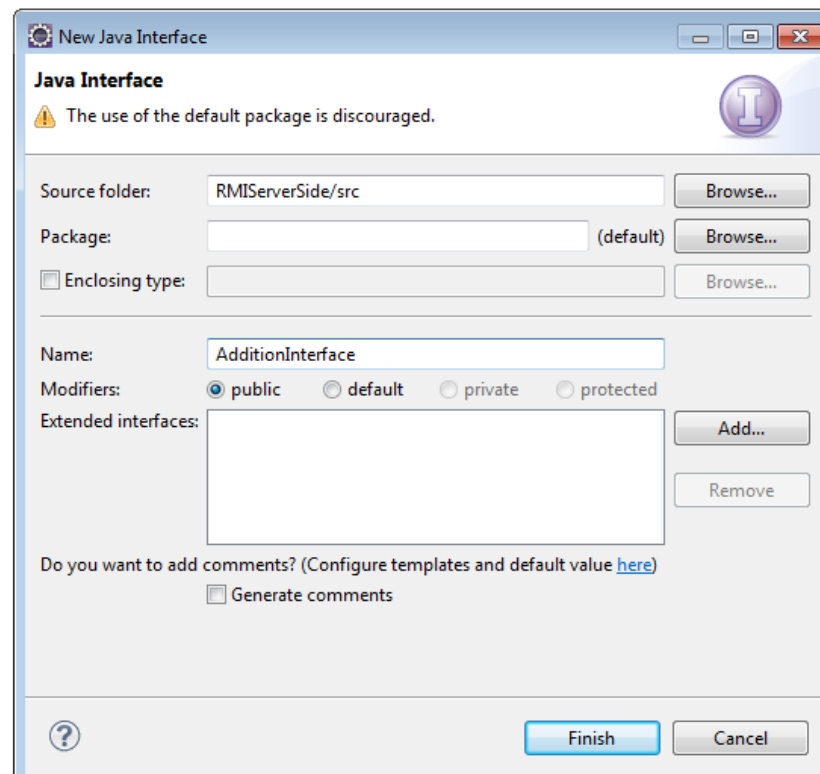
Côté serveur

- 1- Créer un nouveau projet Java en utilisant Eclipse (ou NetBeans ou un autre éditeur de votre choix), et appelons-le : **RMI ServerSide**
- 2- Sous le projet RMI ServerSide, sélectionnez Nouveau -> Interface.



Côté serveur

3- Définissez le nom de l'interface comme suit: AdditionInterface -> Cliquez sur Terminer.



Côté serveur

4- Programmer l'interface **AdditionInterface**

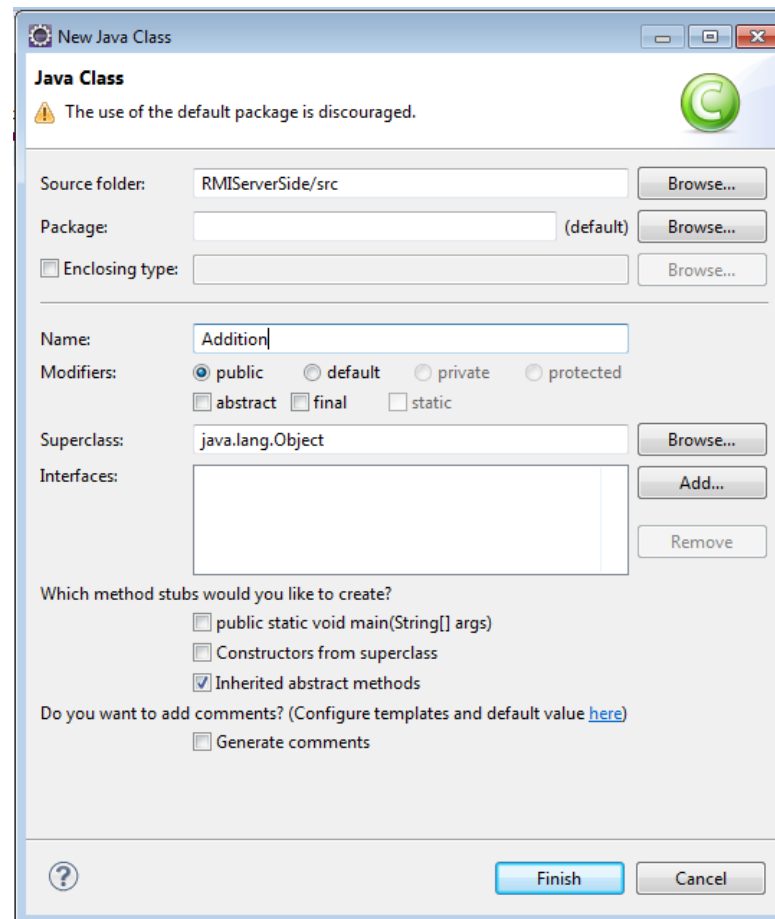
```
1.import java.rmi.*;  
2.  
3.public interface AdditionInterface extends Remote {  
4.public int add(int a,int b) throws RemoteException;  
5.}
```

En étendant l'interface `java.rmi.Remote`, l'interface **AdditionInterface** s'identifie comme une interface dont les méthodes peuvent être appelées à partir d'une autre machine virtuelle Java. Tout objet qui implémente cette interface peut être un objet distant.

Une `RemoteException` est la superclasse commune pour un certain nombre d'exceptions liées à la communication pouvant survenir pendant l'exécution d'un appel de méthode à distance. Chaque méthode d'une interface distante, une interface qui étend `java.rmi.Remote`, doit lister `RemoteException` dans sa clause `throws`.

Côté serveur

5- Sélectionnez le projet RMIServerSide, cliquez sur Nouveau -> Classe, définissez le nom de la classe de la manière suivante: Addition



Côté serveur

6- Programmer la classe Addition

```
1.import java.rmi.*;
2.import java.rmi.server.*;
3.
4.public class Addition extends UnicastRemoteObject
5.implements AdditionInterface {
6.
7.public Addition () throws RemoteException { }
8.
9.public int add(int a, int b) throws RemoteException {
10.int result=a+b;
11.return result;
12.}
13.}
```

`UnicastRemoteObject` est utilisé pour exporter un objet distant avec JRMP et obtenir un stub qui communique avec l'objet distant. Les stubs sont soit générés au moment de l'exécution à l'aide d'objets proxy dynamiques, soit de manière statique lors de la construction, généralement à l'aide de l'outil `rmic`.

Côté serveur

7- Sélectionnez le projet RMI Server Side, cliquez sur Nouveau -> Classe, définissez le nom de la classe comme suit: AdditionServer.

```
1.import java.rmi.*;
2.import java.rmi.server.*;
3.
4.public class AdditionServer {
5.public static void main (String[] argv) {
6.try {
7.Addition Hello = new Addition();
8.Naming.rebind("rmi://localhost/ABC", Hello);
9.
10.System.out.println("Addition Server is ready.");
11.}catch (Exception e) {
12.System.out.println("Addition Server failed: " + e);
13.}
14.}
15.}
```

La classe **Naming** fournit des méthodes pour stocker et obtenir des références à des objets distants dans un registre d'objets distant. Chaque méthode de la classe Naming prend comme l'un de ses arguments un nom java.lang.String au format URL de la forme:

```
//host:port/name
```

La liaison d'un nom pour un objet distant consiste à associer ou à enregistrer un nom pour un objet distant pouvant être utilisé ultérieurement pour rechercher cet objet distant. Un objet distant peut être associé à un nom en utilisant les méthodes bind ou rebind de la classe Naming.

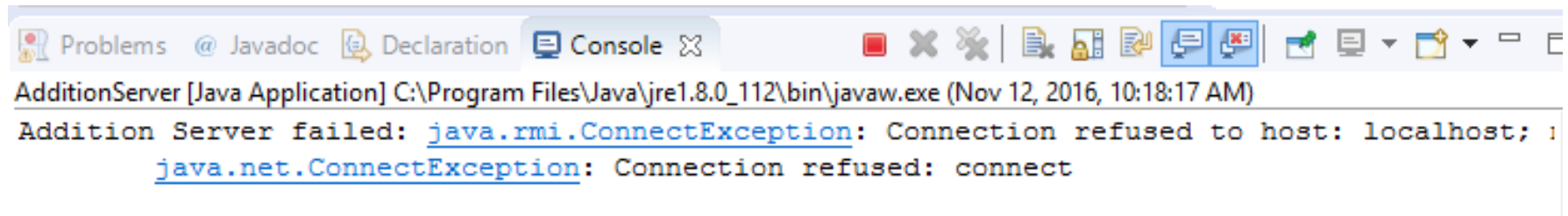
Côté serveur

8- Compilez votre projet en cliquant sur le bouton Lecture verte.



Côté serveur

Ne vous inquiétez pas si vous obtenez l'erreur suivante, car **rmiregistry** n'est pas encore en cours d'exécution.



The screenshot shows a console window from an IDE. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console text reads: 'AdditionServer [Java Application] C:\Program Files\Java\jre1.8.0_112\bin\javaw.exe (Nov 12, 2016, 10:18:17 AM)' followed by the error message: 'Addition Server failed: java.rmi.ConnectException: Connection refused to host: localhost; ; java.net.ConnectException: Connection refused: connect'.

```
AdditionServer [Java Application] C:\Program Files\Java\jre1.8.0_112\bin\javaw.exe (Nov 12, 2016, 10:18:17 AM)
Addition Server failed: java.rmi.ConnectException: Connection refused to host: localhost; ;
java.net.ConnectException: Connection refused: connect
```

Côté serveur

```
cd C:\Users\Amine\workspace\RMISServerSide\bin
```

Exécutez le **rmic** pour générer le stub pour l'addition d'objet distant.
Exécutez la commande suivante:

```
> rmic Addition
```

Si vous obtenez le message: 'rmic' n'est pas reconnu.... Vous devez configurer le chemin pour ajouter le dossier bin du JDK.

Côté serveur

- Dans la même fenêtre CMD, démarrez le registre RMI. Tapez directement.

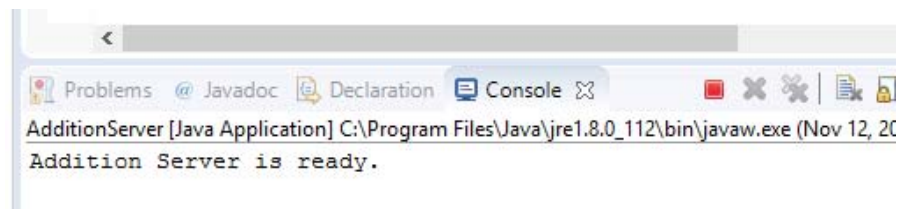
> `start rmiregistry`

Côté serveur

- Maintenant, il est temps de compiler et d'exécuter le **AdditionServer**!
Cliquez sur le bouton vert:



- Le serveur d'addition est prêt maintenant!



Côté Client

- Créez un nouveau projet Java avec Eclipse (ou NetBeans ou un autre éditeur de votre choix) et appelez-le: RMIClientSide-> Cliquez sur Terminer une fois que vous avez terminé.
- Sélectionnez le projet RMIClientSide, cliquez sur Nouveau -> Interface, définissez le nom de la classe comme suit: AdditionInterface, cliquez sur Terminer.

Côté Client

```
1.import java.rmi.*;  
2.  
3.public interface AdditionInterface extends Remote {  
4.public int add(int a,int b) throws RemoteException;  
5.}
```

Côté Client

```
1.import java.rmi.*;
2.
3.public class AdditionClient {
4.public static void main (String[] args) {
5.AdditionInterface hello;
6.try {
7.hello = (AdditionInterface)Naming.lookup("rmi://localhost/ABC");
8.int result=hello.add(9,10);
9.System.out.println("Result is :"+result);
10.
11.}catch (Exception e) {
12.System.out.println("HelloClient exception: " + e);
13.}
14.}
15.}
```

Cet appel **Naming.lookup ()** examine le registre RMI qui s'exécute sur l'hôte local pour rechercher une liaison sous le nom "Hello".

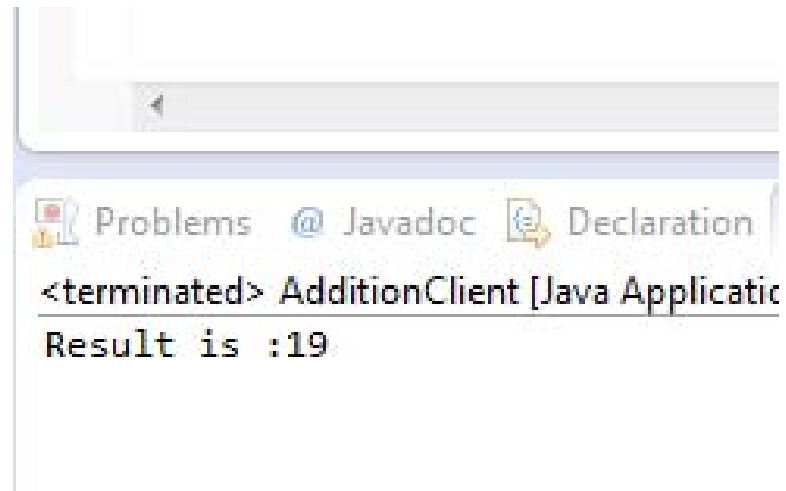
Côté Client

- Cliquez à nouveau sur le bouton vert pour exécuter. C'est le côté client? vous obtiendrez l'erreur suivante!

```
HelloClient exception: java.rmi.UnmarshalException: error  
unmarshalling return; nested exception is:  
java.lang.ClassNotFoundException: Addition_Stub (no security  
manager: RMI class loader disabled)
```

- En effet, il n'y a pas de fichier stub dans le côté client.

Côté Client



Reference

- Imed Bouchrika, New Easy Tutorial for Java RMI using Eclipse

Website: <http://www.ejbtutorial.com/java-rmi/new-easy-tutorial-for-java-rmi-using-eclipse>