

16/07/2022

# Conception d'une calculatrice avec les Nombres complexes

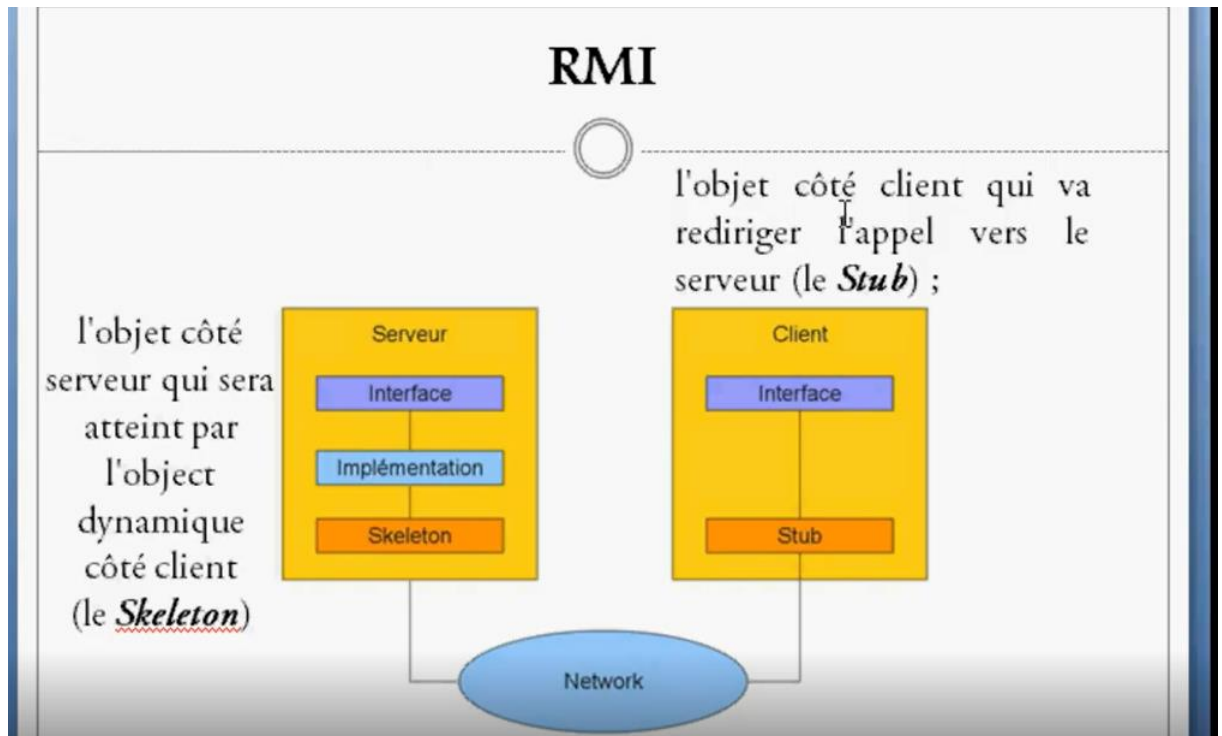
MASTER 1 BDGL

**Etudiant : SACKO ALLOU-BADRA ET KAMATE ISSIAKA**  
**Prof : Dr. ASSOHOUN**

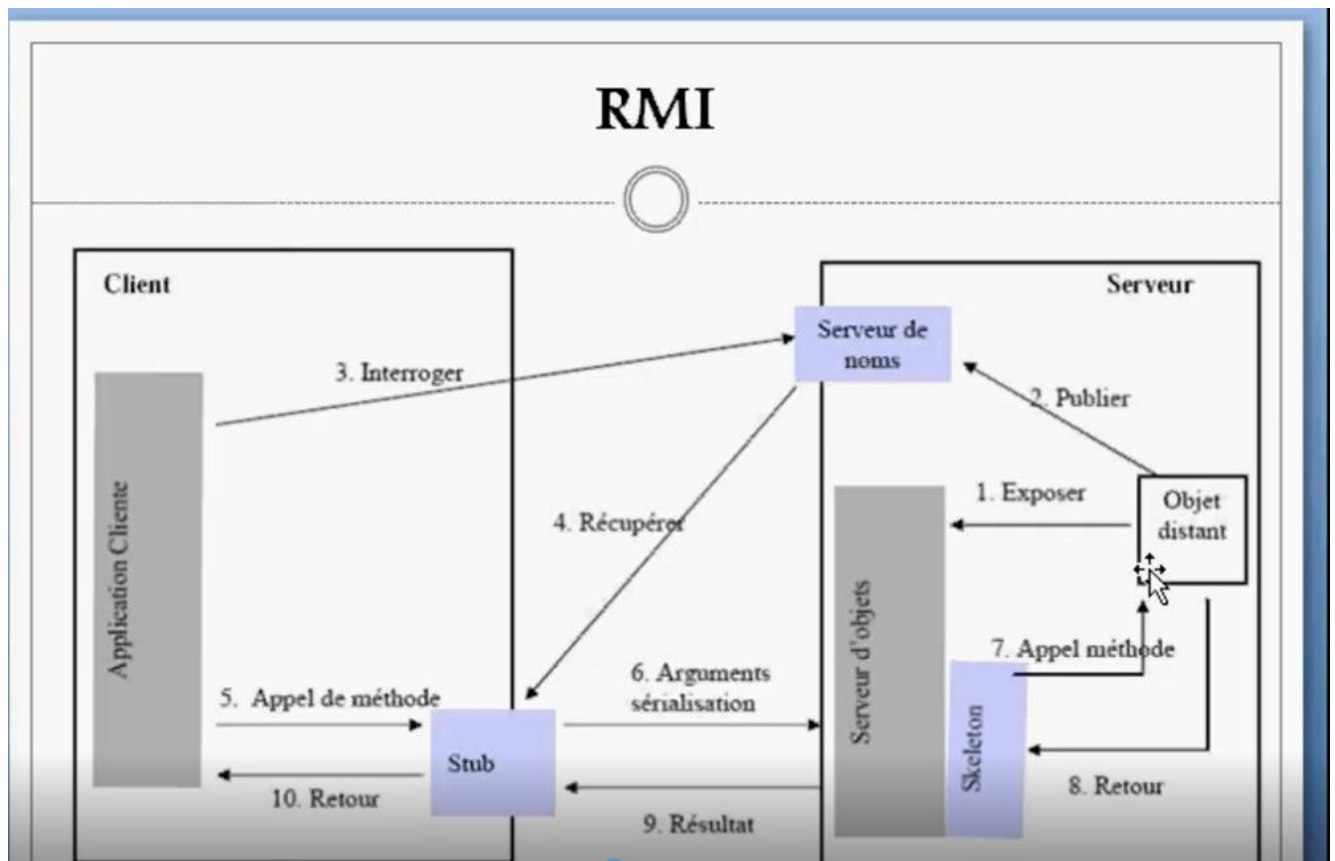
---

## 1. Présentation du Protocol RMI (Remote Method Invocation) :

Le **RMI** est un protocole de communication qui permet d'appeler un code à distance. Une partie du code est exposé sur le client au travers d'interfaces. L'implémentation proprement dite se trouvera du côté du serveur et le rôle sera de faire le nécessaire afin de lier les interfaces aux implémentations distantes.



Le client peut être mis à jour de manière transparente par un simple redéploiement au niveau du serveur pour modifier son comportement. L'exécution distante de la méthode est masquée au code client.



- La première étape consiste à écrire une interface contenant les signatures des méthodes qui seront exposées au travers de RMI.

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer:** Shows a project named **PROJETRMI** with a **src** folder containing **AddRem.class**, **AddRem.java**, **client.class**, **client\$1.class**, and **client\$2.class**.
- Editor:** Displays the **AddRem.java** file with the following code:
 

```

src > AddRem.java > AddRem
1 //define remote interface
2 import java.rmi.*;
3
4 public interface AddRem extends Remote{
5     public String addition (int nbr1,int nbr2, int nbr3, int nbr4) throws RemoteException;
6     public String soustraction (int nbr1, int nbr2, int nbr3, int nbr4) throws RemoteException;
7     public String multiplication(int nbr1, int nbr2, int nbr3, int nbr4) throws RemoteException;
8     public String conjugue(int nbr1, int nbr2) throws RemoteException;
9     public double module(int nbr1, int nbr2) throws RemoteException;
10 }
      
```

- Il faudrait créer une classe qui implémente les différentes méthodes définies dans l'interface.
- Le serveur sera une simple classe contenant une méthode main.

Dans la classe du serveur main, nous instancieront notre implémentation. Ensuite nous créerons notre **skeleton** que nous lierons grâce à notre **registry**.

```
//server programm
import java.rmi.*;
import java.net.*;

public class AddServer{
    public static void main(String args[]){
        try{
            AddRemImpl locobj=new AddRemImpl();
            Naming.rebind("rmi:///AddRem",locobj);
        }catch(RemoteException re){
            re.printStackTrace();
        }catch (MalformedURLException mfe){
            mfe.printStackTrace();
        }
    }
}
```

- Le serveur RMI est dorénavant disponible, nous pouvons créer un client qui se connectera à ce serveur pour invoquer le code présent sur le serveur.

## 2. Les interfaces du Logiciel

- ACCUEIL :

The screenshot shows a Java Swing window titled "ACCUEIL" with a dark blue background. In the top right corner, there is a red "EXIT" button. The window is divided into three main sections: "Partie Réelle", "Partie Imaginaire", and "Resultat". Each section contains two white rectangular input fields. Below these input fields, there are five blue buttons with white text: "ADDITION", "SOUSTRACTION", "MULTIPLICATION", "CONJUGUE", and "MODULE".

- Addition :

EXIT

Partie Réelle	Partie Imaginaire	Resultat
1	2	4+(6)i
3	4	

ADDITION

SOUSTRACTION

MULTIPLICATION

CONJUGUE

MODULE

## Soustraction :

EXIT

Partie Réelle	Partie Imaginaire	Resultat
1	2	-2+(-2)i
3	4	

ADDITION

SOUSTRACTION

MULTIPLICATION

CONJUGUE

MODULE

- Multiplication

EXIT

Partie Réelle	Partie Imaginaire	Resultat
1	2	-5+(12)i
3	4	

ADDITION SOUSTRACTION MULTIPLICATION CONJUGUE MODULE

- Conjugué :

EXIT

Partie Réelle	Partie Imaginaire	Resultat
1	2	1-2i
3	4	

ADDITION SOUSTRACTION MULTIPLICATION CONJUGUE MODULE

- Module :

operation - Tous droits réservés

EXIT

Partie Réelle	Partie Imaginaire	Resultat
1	2	2.23606797749979
3	4	

ADDITION SOUSTRACTION MULTIPLICATION CONJUGUE MODULE