RPC
Définitions

- Remote Procedure Call
- Protocole réseau permettant à un ordinateur (client) de faire des appels de procédures sur un ordinateur distant (serveur)
- Mécanisme de communication interprocessus qui permet l'invocation de fonctionnalités d'un processus différent. Ce dernier peut être sur la même machine, sur le réseau local ou sur Internet

RPC
IDL

- Interface Definition Language
- Langage permettant de définir une interface qui permet de faire communiquer des processus implémentés dans des langages différents, ou déployés à travers un réseau sur des systèmes hétérogènes (Windows, Linux, Mac OS, etc.)

RPC
Stub

- Programme qui convertit les données transmises entre le client et le serveur lors d'un appel de procédure à distance (RPC)
- Stub client
- Stub serveur
- Génération des stubs :
  - Manuellement : le programmeur écrit ses propres stubs
  - Automatiquement : le programmeur utilise IDL pour définir l'interface entre le client et le serveur et c'est le compilateur IDL qui se charge de créer les stubs

RPC
Etapes Client

- Appel de la procédure du stub client
- Conversion des paramètres (arguments) dans un format de transmission réseau
- Envoi des paramètres (arguments) au serveur
- Attente de la réponse du serveur
- Réception de la réponse du serveur
- Conversion des données reçues du format de transmission réseau au format utilisé par le client
- Ecriture des données reçues dans la mémoire du client
- Exploitation des données reçues par le client

RPC
Etapes Serveur

- Acceptation de la demande du client
- Conversion des paramètres (arguments) reçus du format de transmission réseau au format utilisé par le serveur
- Ecriture des paramètres (arguments) reçus dans la mémoire du serveur
- Appel de la procédure du stub serveur
- Appel de la procédure réelle sur le serveur
- Envoi du résultat au stub serveur
- Conversion du résultat dans un format de transmission réseau
- Envoi du résultat au client

RPC
Etapes Programmation

- Définition de l'interface en utilisant IDL
- Ecriture du stub client et du stub serveur / Génération du stub client et du stub serveur en compilant l'interface
- Compilation du stub client et du stub serveur
- Ecriture du programme client et du programme serveur / Modification du squelette du programme client et du squelette du programme serveur générés par la compilation de l'interface
- Compilation du programme client et du programme serveur en liant chaque programme à son stub

RMI
Définitions

- Remote Method Invocation
- API (Application Programming Interface) Java qui permet d'appeler des méthodes distantes
- Mécanisme permettant l’appel de méthodes entre objets Java s’exécutant sur le même ordinateur ou sur des ordinateurs distants reliés par un réseau

RMI
Etapes Programmation

- Définition de l'interface distante
- Définition d'une classe qui implémente l'interface distante
- Compilation de la classe qui implémente l'interface distante (javac)
- [Génération du Stub et du Skeleton en compilant la classe qui implémente l'interface distante (rmic)]
- Ecriture du programme serveur et du programme client
- Compilation du programme serveur et du programme client (javac)

- Remarque :
  - Il faut démarrer le service de registres RMI (rmiregistry) avant d'exécuter le programme serveur

RMI
Exemple Interface Distante

```
import    java.rmi.Remote;
import    java.rmi.RemoteException;

public    interface    Calculatrice    extends    Remote
{
            public    float    addition(float    n1, float    n2)    throws    RemoteException;
            public    float    soustraction(float    n1, float    n2)    throws    RemoteException;
            public    float    multiplication(float    n1, float    n2)    throws    RemoteException;
            public    float    division(float    n1, float    n2)    throws    RemoteException;
}
```

RMI
Exemple Classe Implémentation

```
import   java.rmi.server.UnicastRemoteObject;
import   java.rmi.RemoteException;

public   class   CalculatriceImpl   extends   UnicastRemoteObject   implements   Calculatrice
{
            public   CalculatriceImpl()   throws   RemoteException
            {
                        super();
            }
            public   float   addition(float   n1, float   n2)   throws   RemoteException
            {
                        return   n1 + n2;
            }
            public   float   soustraction(float   n1, float   n2)   throws   RemoteException
            {
                        return   n1 - n2;
            }
            public   float   multiplication(float   n1, float   n2)   throws   RemoteException
            {
                        return   n1 * n2;
            }
            public   float   division(float   n1, float   n2)   throws   RemoteException
            {
                        return   n1 / n2;
            }
}
```

RMI
Exemple Programme Serveur

```
import   java.rmi.Naming;

public   class   CalculatriceServ
{
        public   static   void   main(String   args[])
        {
                try
                {
                        CalculatriceImpl   c   =   new   CalculatriceImpl();
                        Naming.rebind("rmi://127.0.0.1/CalculatriceService", c);
                }
                catch (Exception   e)
                {
                        System.out.println("Erreur Serveur : "   +   e);
                }
        }
}
```

RMI
Exemple Programme Client

```
import   java.rmi.Remote;
import   java.rmi.Naming;

public   class   CalculatriceCli
{
            public   static   void   main(String[]   args)
            {
                        try
                        {
                           Remote   r   =   Naming.lookup("rmi://127.0.0.1/CalculatriceService");
                           System.out.println(((Calculatrice)   r).addition(7, 7));
                           System.out.println(((Calculatrice)   r).soustraction(7, 7));
                           System.out.println(((Calculatrice)   r).multiplication(7, 7));
                           System.out.println(((Calculatrice)   r).division(7, 7));
                        }
                        catch (Exception   e)
                        {
                                    System.out.println("Erreur Client : "   +   e);
                        }
            }
}
```