UNIVERSITÉ DE JENDOUBA
INSTITUT SUPÉRIEUR DE L'INFORMATIQUE DU KEF

TP 6 : " Introduction à Java RMI (Remote Method Invocation) Communication à Distance"

Matière : Développement d'applications réparties **Niveau :** 3eme LSI
Responsable : Hanen Samaali **A U :** 2023 / 2024

Objectif du TP : Ce TP vise à initier les étudiants aux concepts et à la mise en œuvre de la communication à distance en utilisant Java RMI. Les participants exploreront la création d'un serveur exposant un service distant, l'implémentation du service, et la mise en place d'un client utilisant ce service à distance. L'accent sera mis sur la compréhension des principes fondamentaux de Java RMI, la création d'interfaces distantes, et la communication entre objets situés sur des machines distantes.

Résumé du TP :

le serveur (RpcServer) expose un service distant, le client (RpcClient) utilise ce service à distance en le recherchant dans le registre, l'implémentation du service (CalculatorServiceImpl) effectue les opérations demandées et l'interface (CalculatorService) définit les méthodes que le client peut appeler à distance. La communication entre le client et le serveur se fait via RMI, et les objets sont enregistrés et localisés à l'aide du registre RMI.

Les principaux objectifs sont les suivants :

Comprendre les concepts de base de Java RMI :

Compréhension du mécanisme de communication entre des objets situés sur des machines distantes.

Utilisation d'une interface RMI pour déclarer les méthodes qui peuvent être appelées à distance.

Implémentation d'un service distant :

Création d'une interface (CalculatorService) définissant les méthodes à exposer à distance.

Implémentation concrète du service distant (CalculatorServiceImpl) avec les opérations à effectuer.

Création du serveur RMI :

Mise en place d'un serveur RMI qui crée une instance du service distant et l'enregistre dans le registre RMI.

Création du client RMI :

Écriture d'un client RMI qui localise le registre RMI, recherche l'objet distant et invoque ses méthodes à distance.

Relation entre les objets :

RpcClient communique avec l'objet distant CalculatorService du serveur.

L'objet distant CalculatorService est une instance de CalculatorServiceImpl, qui implémente les méthodes définies dans l'interface CalculatorService.

Les méthodes de CalculatorServiceImpl sont exécutées côté serveur en réponse aux appels distants de RpcClient.

1. **CalculatorService (Interface) :**

Rôle : La définition des méthodes distantes.

Déclare les méthodes que le client peut appeler à distance.

Hérite de l'interface Remote, qui est une interface de marquage pour identifier les interfaces distantes.

Chaque méthode déclarée dans cette interface doit lancer RemoteException, car les appels distants peuvent générer des exceptions.

1. Interface `CalculatorService`

```
// Importe les classes nécessaires pour la définition d'une interface RMI
import java.rmi.Remote;
import java.rmi.RemoteException;

// Définit l'interface RMI CalculatorService, qui étend l'interface Remote
public interface CalculatorService extends Remote {

    // Déclare une méthode distante "add" qui prend deux entiers en paramètres

    int add(int a, int b) throws RemoteException;

    // Déclare une autre méthode distante "subtract" qui prend deux entiers en
paramètres
    int subtract(int a, int b) throws RemoteException; }
```

2. **CalculatorServiceImpl :**

Rôle : L'implémentation du service RMI.

Implémente l'interface distante (CalculatorService) qui déclare les méthodes à exécuter à distance.

Étend UnicastRemoteObject pour faciliter l'exportation de l'objet distant.

Lorsqu'une méthode est appelée, effectue les opérations correspondantes et renvoie les résultats.

```
   // Importe les classes nécessaires pour gérer les exceptions RMI
```

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

// Définit la classe CalculatorServiceImpl qui implémente l'interface distante
CalculatorService
public class CalculatorServiceImpl extends UnicastRemoteObject implements
CalculatorService {

    // Constructeur de la classe, qui doit lancer une RemoteException
    public CalculatorServiceImpl() throws RemoteException {

    }

    // Implémente la méthode distante "add" déclarée dans l'interface
CalculatorService
    @Override
    public int add(int a, int b) throws RemoteException {

        return a + b;
    }

    // Implémente la méthode distante "subtract" déclarée dans l'interface
CalculatorService
    @Override
    public int subtract(int a, int b) throws RemoteException {

        return a - b;
    }

}
```

3. RpcServer :

Crée une instance de l'objet distant (CalculatorServiceImpl), qui expose les méthodes que le client peut appeler à distance.

Crée un registre RMI, un répertoire où les objets distants sont enregistrés pour que les clients puissent les trouver. Enregistre l'objet distant (CalculatorServiceImpl) dans le registre RMI sous un nom spécifique ("CalculatorService").

Reste en attente des appels distants des clients.

```java
// Importe les classes nécessaires pour la gestion des exceptions RMI et la création
du registre RMI
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class RpcServer {
        public static void main(String[] args) {
        try {
            // Crée une instance de l'objet distant (implémentation du service)
            CalculatorService calculatorService = new CalculatorServiceImpl();

            // Crée un registre RMI sur le port 1099 (port par défaut pour le
registre RMI)
            Registry registry = LocateRegistry.createRegistry(1099);

            // Enregistre l'objet distant dans le registre sous le nom
"CalculatorService"
            registry.rebind("CalculatorService", calculatorService);
```

```
            System.out.println("Server is ready.");
        } catch (Exception e) {
            // En cas d'exception, imprime la trace de la pile (stack trace)
            e.printStackTrace();
        }
    }
}
```

4. RpcClient :

Localise le registre RMI (le même registre utilisé par le serveur) en spécifiant l'adresse du serveur et le numéro de port.

Recherche l'objet distant (CalculatorService) dans le registre en utilisant un nom spécifique.

Une fois l'objet distant obtenu, le client peut appeler ses méthodes à distance pour effectuer des opérations distantes.

```
// Importe les classes nécessaires pour la gestion des exceptions RMI et la
localisation du registre RMI
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class RpcClient {

    public static void main(String[] args) {
        try {
            // Obtient le registre RMI local sur le port 1099 (le port par défaut
pour le registre RMI)
            Registry registry = LocateRegistry.getRegistry("localhost", 1099);

            // Recherche l'objet distant enregistré dans le registre sous le nom
"CalculatorService"
            CalculatorService calculatorService = (CalculatorService)
registry.lookup("CalculatorService");

            // Utilise les méthodes du service RPC (objet distant)
            int result = calculatorService.add(10, 5);
            System.out.println("Result of addition: " + result);
        } catch (Exception e) {

            e.printStackTrace();
        }
    }
}
```