Programmation orientée objet 2 (TP-05)
Remote Method Invocation (RMI)

Method :

Step 1: Define the Interface

Create an interface that extends java.rmi.Remote and declares the methods that you want to make remotely accessible.

import java.rmi.Remote; Hello.java

import java.rmi.RemoteException;

```
public interface Hello extends Remote {
   String sayHello() throws RemoteException;
}
```

Step 2: Implement the Interface HelloImpl.php

Create a class that implements the interface. This class will contain the actual implementation of the methods declared in the interface.

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

```
public class HelloImpl extends UnicastRemoteObject implements Hello {
   public HelloImpl() throws RemoteException {
      super();
   }

   @Override
   public String sayHello() throws RemoteException {
      return "Hello, World!";
   }
}
```

Step 3: Create the Server HelloServer.php

Create a server class that registers the remote object with the RMI registry and waits for client requests.

```
import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;


public class HelloServer {

   public static void main(String[] args) {

      try {

         // Create the remote object

         HelloImpl helloObj = new HelloImpl();


         // Bind the remote object to the registry

         Registry registry = LocateRegistry.createRegistry(1099);

         registry.bind("Hello", helloObj);

/*You can write this way too

        Hello server = new  Hello();

         Naming.rebind("rmi://localhost/Hello", server); */


         System.out.println("Server ready");

      } catch (Exception e) {

         System.err.println("Server exception: " + e.toString());

         e.printStackTrace();

      }

   }

}
```

Step 4: Create the Client

Create a client class that looks up the remote object in the RMI registry and invokes its methods.

```
import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;
```

```
public class HelloClient {
   public static void main(String[] args) {
      try {
         // Look up the remote object from the registry
         Registry registry = LocateRegistry.getRegistry("localhost", 1099);
         Hello helloObj = (Hello) registry.lookup("Hello");
/* you can write it this way too
 Hello helloObj = ( Hello) Naming.lookup("rmi://localhost/Hello"); */

         // Call the remote method
         String message = helloObj.sayHello();
         System.out.println("Message from server: " + message);
      } catch (Exception e) {
         System.err.println("Client exception: " + e.toString());
         e.printStackTrace();
      }
   }
}
```

Step 5: Compile and Run

Exercise 01 :
Implement a basic RMI server and client to perform arithmetic operations :

```
public interface Calculator extends Remote {
   int add(int a, int b) throws RemoteException;
   int subtract(int a, int b) throws RemoteException;
   int multiply(int a, int b) throws RemoteException;
   int divide(int a, int b) throws RemoteException;
}
```

Exercise 02 : Implement an RMI-based chat application.

1. Define the Remote Interface

Define a remote interface called ChatService that will define the methods available for sending and receiving messages in our chat application :

```
// Method for sending a message
void sendMessage(String message) throws RemoteException;
// Method for receiving a message
String receiveMessage() throws RemoteException;
```

2. Implement the Server(Implementation) The messages should be added to the list in sending Messages, and should be removed when sending messages

```
public ChatServer() throws RemoteException {
    super();
    messages = new ArrayList<>();
}
```

3. Start the RMI Registry and Bind the Server (Name your register ChatService)

4. Implement the client :

```
import java.rmi.Naming;


public class ChatClient {
    public static void main(String[] args) {
        try {
            // Look up the remote ChatService object from the RMI registry
            ChatService chatService = (ChatService) Naming.lookup("rmi://localhost/ChatService");


            // Example usage: send a message to the server
            chatService.sendMessage("Hello from client");


            // Example usage: receive a message from the server
            String message = chatService.receiveMessage();
            System.out.println("Received message: " + message);
```

```
        } catch (Exception e) {
            System.err.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```