**📘 RMI-Based Calculator Application**

**1. 📌 Project Description**

This project is a simple **Java RMI (Remote Method Invocation)** based application that allows a client to perform basic arithmetic operations—**addition, subtraction, multiplication, and division**—by invoking methods on a remote server object.

The purpose of this project is to demonstrate how distributed applications can be built using Java's RMI, enabling remote communication between objects across different JVMs.

**2. 🛠️ Technologies and Concepts Used**

**🔹 Java RMI (Remote Method Invocation)**

* Allows objects to invoke methods across a network.
* Used to simulate distributed computing.

**🔹 Core Java Packages Used:**

* java.rmi.\* — Core RMI classes and interfaces.
* java.rmi.server.UnicastRemoteObject — Used to export remote objects to receive incoming calls.
* java.rmi.registry.LocateRegistry — Used to create and locate the registry that holds remote objects.

**3. 📂 Project Structure**

CalculatorProject/

│

├── Calculator.java // Remote Interface

├── CalculatorImpl.java // Remote Interface Implementation

├── Server.java // RMI Server: Registers the remote object

├── Client.java // RMI Client: Accesses remote methods

* **Calculator.java**  
  Defines the remote interface with method declarations: add(), subtract(), multiply(), and divide().
* **CalculatorImpl.java**  
  Implements the Calculator interface and provides the actual logic for the arithmetic operations.
* **Server.java**  
  Creates and registers the remote object in the RMI registry.
* **Client.java**  
  Looks up the remote object and allows the user to perform remote operations.

**4. ▶️ How to Run the Project**

**✅ Prerequisites:**

* Java JDK installed and environment variables set.
* Command-line interface (cmd/terminal).

**🧭 Steps to Run:**

1. **Open CMD** and navigate to the project folder:

cd "Dir/remoteCalculator"

1. **Compile all .java files**:

javac \*.java

1. **Start the RMI registry** in the background:

start rmiregistry

1. **Run the Server**:

java Server

1. **In a new CMD window, run the Client**:

java Client

**📷 Sample Output:**

Your application supports both automated and user-defined inputs. The user can enter two numbers and an operator, and the result is displayed by invoking the method on the server remotely.

**✅ Output Example**

Addition: 15.0

Subtraction: 5.0

Multiplication: 50.0

Division: 2.0

Do this Yourself

Enter the numbers:

23

87

Enter the operator (+,-,\*,/):

\*

The final result:

23.0 \* 87.0 = 2001.0

## 📘 Terminologies Used in the RMI Calculator Project

### 1. ****RMI (Remote Method Invocation)****

A Java API that allows methods of remote Java objects to be invoked from other Java virtual machines (JVMs), possibly located on different machines.

### 2. ****Remote Interface****

An interface that extends java.rmi.Remote. It declares methods that can be called remotely by a client.

### 3. ****Remote Object****

An object of a class that implements a remote interface and extends UnicastRemoteObject. It can be accessed from a remote JVM.

### 4. ****UnicastRemoteObject****

A superclass provided by Java RMI used to export remote objects so they can accept incoming calls from clients.

### 5. ****Stub (Client-Side Proxy)****

Automatically generated proxy class that resides on the client side. It forwards the client’s method calls to the actual remote object on the server.

Note: In modern Java (since JDK 5.0+), dynamic stubs are created at runtime—no need to manually generate them with rmic.

### 6. ****Skeleton (Server-Side Proxy)****

Used in older versions of Java (before JDK 1.2). In modern RMI, skeletons are not required anymore.

### 7. ****RMI Registry****

A simple naming service that allows clients to obtain a reference to a remote object by name. It must be started before binding any remote object.

📄 Command used:

start rmiregistry

### 8. ****Binding****

Registering a remote object with a name in the RMI registry using Naming.rebind() or Naming.bind().

### 9. ****Lookup****

A client-side operation to find a remote object from the RMI registry using its name.

### 10. ****Serialization****

The process of converting an object into a byte stream to send it over the network. All remote method arguments and return values must be serializable.

### 11. ****Exception Handling in RMI****

Remote methods must handle exceptions like RemoteException, which signals issues in the remote method call (e.g., connection failure, server down).

**📚 References / Bibliography**

1. **Oracle Java RMI Documentation**  
   *Java™ Remote Method Invocation (RMI)*  
   <https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/index.html>
2. **Oracle Java SE API Documentation**  
   *Package: java.rmi*  
   <https://docs.oracle.com/javase/8/docs/api/java/rmi/package-summary.html>
3. **GeeksforGeeks**  
   *Java RMI (Remote Method Invocation)*  
   https://www.geeksforgeeks.org/java-rmi-remote-method-invocation/
4. **TutorialsPoint**  
   *Java RMI - Quick Guide*  
   https://www.tutorialspoint.com/java\_rmi/index.htm
5. **JournalDev**  
   *Java RMI Tutorial*  
   https://www.journaldev.com/2858/java-rmi-example
6. **Javatpoint**  
   *Java RMI Tutorial*  
   https://www.javatpoint.com/RMI