BÁO CÁO THỰC HÀNH

**Môn học: Lập trình ứng dụng mạng**

**Buổi báo cáo: Lab 05**

**Tên chủ đề:** Remote Method Invocation

*GVHD: Trần Mạnh Hùng*

*Ngày thực hiện: 26/5/2024*

**THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT109.O21.MMCL

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Văn Anh Tuấn | 21522757 | 21522757@gm.uit.edu.vn |

1. **ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| **Nội dung** | **Kết quả** |
| Tổng thời gian thực hiện bài thực hành trung bình | 1 ngày |
| Link Video thực hiện  *(nếu có)* |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất … |  |
| Điểm tự đánh giá | 10/10 |

BÁO CÁO CHI TIẾT

**1. RMI là gì ?**

**RMI** là viết tắt của **Remote Method Invocation**. Đó là một cơ chế cho phép một đối tượng cư trú trong một hệ thống (JVM) truy cập/gọi một đối tượng chạy trên một JVM khác.

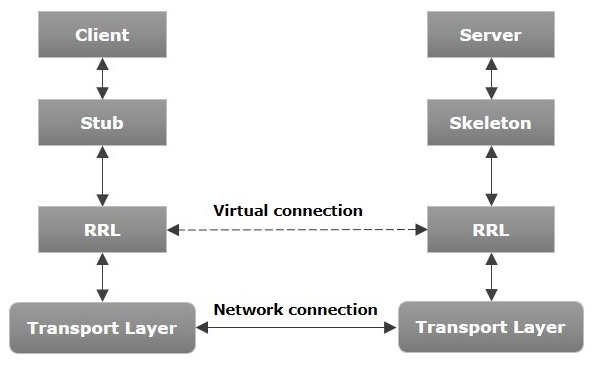
**RMI** được sử dụng để xây dựng các ứng dụng phân tán, nó cung cấp khả năng liên lạc từ xa giữa các chương trình Java. **RMI** được cung cấp trong gói **java.rmi**.

**Kiến trúc của một ứng dụng RMI:**

Trong ứng dụng **RMI**, cần có tối thiểu 2 chương trình, **Server và Client**:

* Bên trong **Server**, một **Remote Object** được tạo và tham chiếu của đối tượng đó được cung cấp cho **Client** sử dụng **RMI registry**.
* **Client** yêu cầu các **Remote Object** trên máy chủ và cố gắng gọi các phương thức của nó.

Sơ đồ dưới đây thể hiện kiến ​​trúc của một ứng dụng **RMI**:

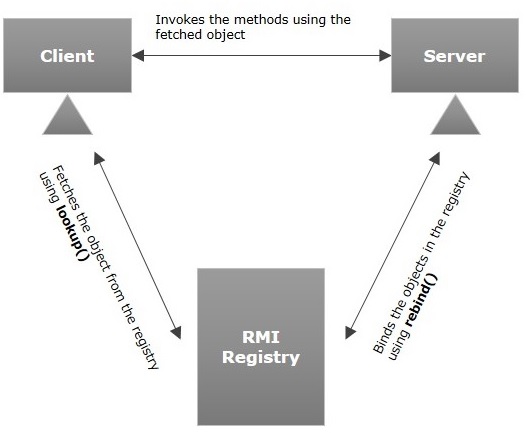


* **Transport Layer** - Lớp này kết nối **Client** và **Server**. Nó quản lý kết nối hiện có và cũng thiết lập các kết nối mới.
* **Stub** – stub là một đại diện (proxy) của đối tượng từ xa tại **Client**. Nó hoạt động như một gateway cho **Client**.
* **Skeleton** − Đây là đối tượng nằm ở phía **Server**. **Stub** giao tiếp với **Skeleton** để chuyển yêu cầu đến đối tượng ở xa.
* **RRL**- Đây là lớp quản lý các tham chiếu do **Client** tạo ra tới **Remote Object**.

**RMI Registry:**

**RMI Registry** là một không gian tên nơi đặt tất cả các **server object**. Mỗi lần Server tạo một Object, nó sẽ đăng ký đối tượng này với **RMI Registry** (sử dụng các phương thức **bind()** hoặc **reBind()**). Chúng được đăng ký bằng một tên duy nhất được gọi là **bind name**.

Để gọi một **Remote Object**, **Client** cần có một tham chiếu đến đối tượng đó. Vào thời điểm đó, Client tìm và **fetches** đối tượng từ **RMI Registry** bằng **bind name** của nó (sử dụng phương thức **lookup()**).

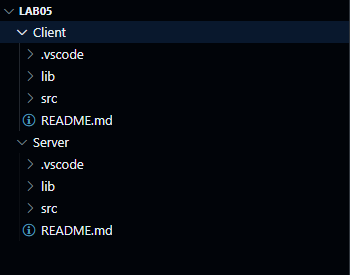


**2. Ứng dụng RMI cơ bản:**

**Mô tả về ứng dụng:** Client nhập văn bản từ console, Server nhận được và echo ra màn hình. Dừng khi client nhập “quit”.

**Chi tiết:**

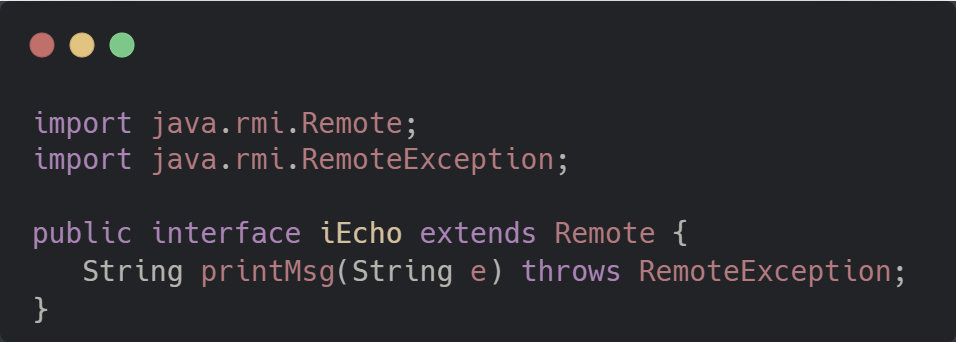
Tạo 2 project Java, Client và Server:



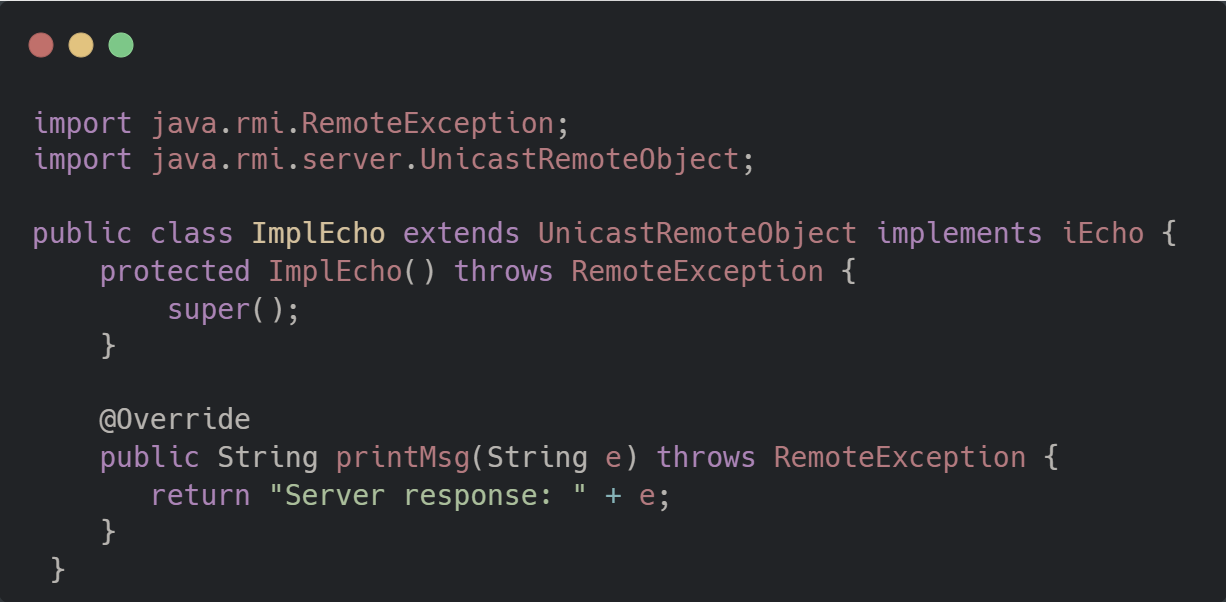
* **Server:**

**Remote Interface**: cung cấp mô tả về tất cả các phương thức của một đối tượng từ xa cụ thể. **Client** giao tiếp với **Remote Interface** này.

Dưới đây là một ví dụ về **Remote Interface**. Ở đây chúng ta đã định nghĩa một giao diện có tên **iEcho** và nó có một phương thức gọi là **printMsg().**



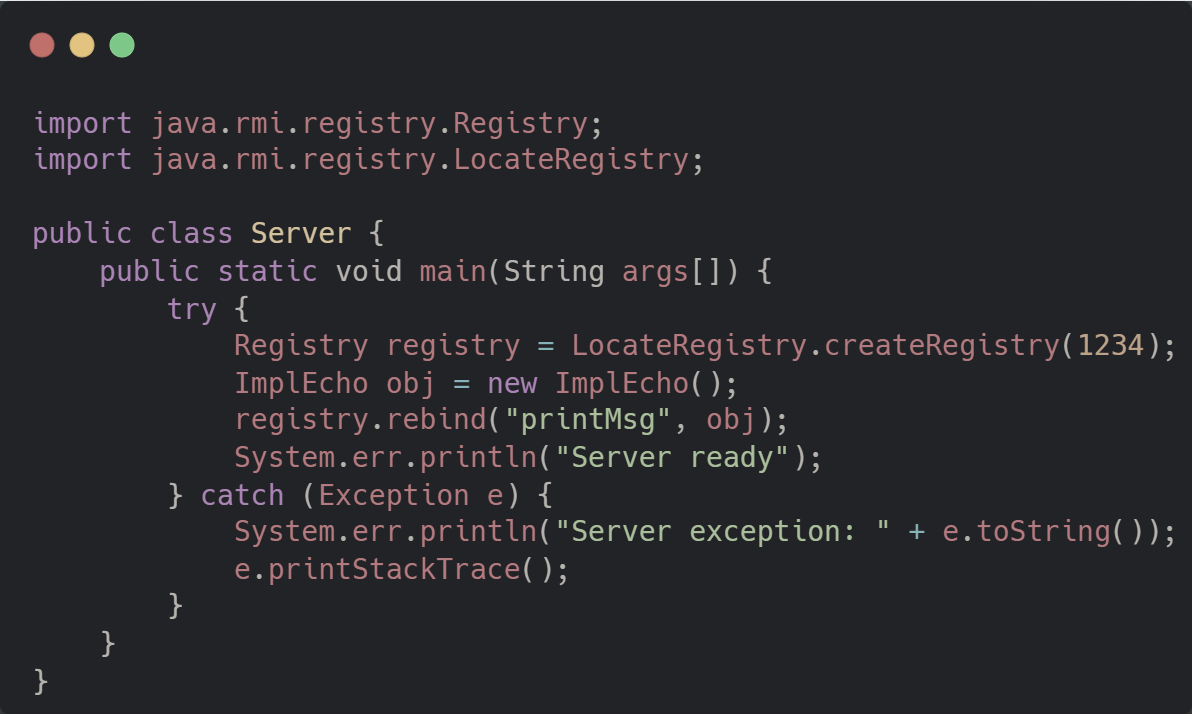
**Implementation Class:** Sau khi đã tạo **Remote Interface** chúng ta sẽ **implements** nó vào class **ImplEcho.** Sau đó overide phương thức đã khai báo ở **iEcho** và định nghĩa cho phương thức đó. Ở đây chúng ta sẽ in ra những gì nhận được từ **Client**.



**Server Class:** Chương trình máy chủ **RMI** sẽ triển khai **Remote Interface** hoặc mở rộng lớp triển khai. Ở đây, chúng ta tạo một **Remote Object** và liên kết nó với **RMIRegistry**.

RMIRegistry sẽ đăng ký lắng nghe ở port được chỉ định là 1234. Sau đó, khởi tạo

**Remote Object** (**ImplEcho**) và đăng ký vào **RMI Registry.**

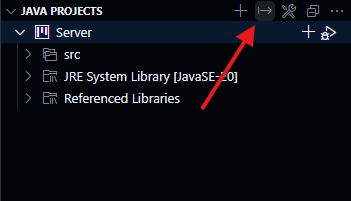


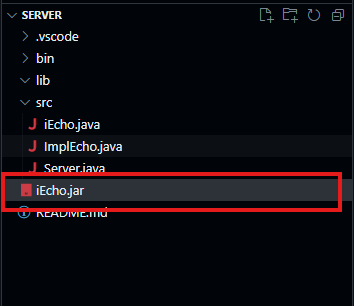
**Sự khác nhau giữa bind() và rebind():** về cơ bản cả 2 đều có chức năng như nhau là sử dụng để đăng ký một đối tượng từ xa vào registry. **bind(String name)** sẽ liên kết tên đã được chỉ định vào một đối tượng từ xa trong khi **rebind(String name)** liên kết tên đã được chỉ định vào một đối tượng từ xa mới. Điều này tạo nên sự khác biệt giữa bind() và rebind(), nếu tên đã tồn tại trong registry phương thức bind() sẽ gây ra lỗi ngược lại vì rebind() tạo ra đối tượng từ xa mới nó sẽ cho phép thay đổi các đối tượng từ xa mà không cần xóa liên kết hiện có.

* **Client:**

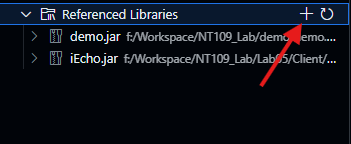
Đầu tiên, export file **iEcho.jar** ở project **Server**. Sau đó, **add file jar** này vào **Client** project.

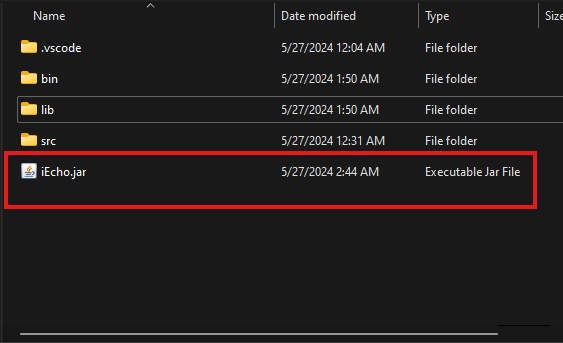
Bước 1: Export iEcho.jar.





Bước 2: add file iEcho.jar vào Client project



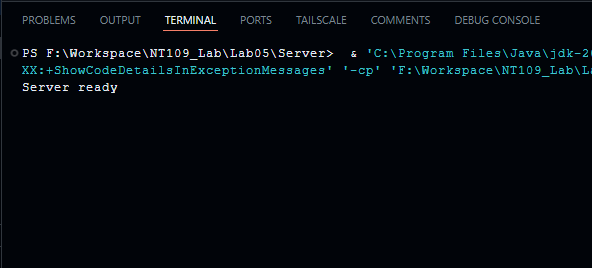


**Client Class:** ở phía **client** sẽ tìm nạp đối tượng từ xa và gọi phương thức được yêu cầu bằng đối tượng này. Đầu tiên, gọi đến **server** thông qua **localhost** ở port **1234**. Sau đó, sử dụng **registry lookup()** để nạp đối tượng từ xa vào **stub**. Sử dụng **stub** để gọi đối tượng từ xa. Cuối cùng, in ra console kết quả trả về và dừng lại khi **client** nhập vào **“quit”**.

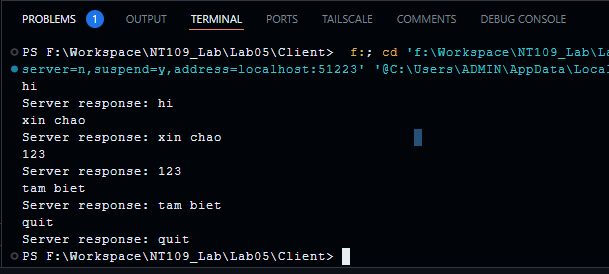


**Kết quả:**

Ở phía Server sau khi khởi chạy thành công sẽ in ra dòng chữ “Server ready”



Phía client, sau khi chạy sẽ đợi người dùng nhập vào và server sẽ trả về kết quả. Dừng lại khi người dùng nhập vào quit.



**3. Ứng dụng CORBA:**

**a. CORBA là gì ?**

**HẾT**