



LẬP TRÌNH MẠNG

RMI

hungdn@ptit.edu.vn

Nội dung



- Lập trình phân tán và RMI
- Xây dựng ứng dụng phân tán RMI
- Demo

- *Một số kỹ thuật với RMI*
 - *Lớp Stub ở máy khách*
 - *Vấn đề truyền dữ liệu trong RMI*
 - *Factory Object*



1. Lập trình phân tán và RMI

Giới thiệu



- Kỹ thuật lập trình phân tán thực chất là kỹ thuật lập trình phân tán mã lệnh và đối tượng
- Một số kỹ thuật lập trình phân tán
 - Kỹ thuật gọi thủ tục từ xa RPC(Remote Procedure Call)
 - Kỹ thuật gọi phương thức từ xa **RMI** (Remote Method Invocation)
 - Kỹ thuật mô hình đối tượng thành phần phân tán DCOM
 - Kỹ thuật kiến trúc môi giới trung gian **CORBA**
 - Kỹ thuật EJB, WebService, RPC-XML...

Sơ lược về RPC

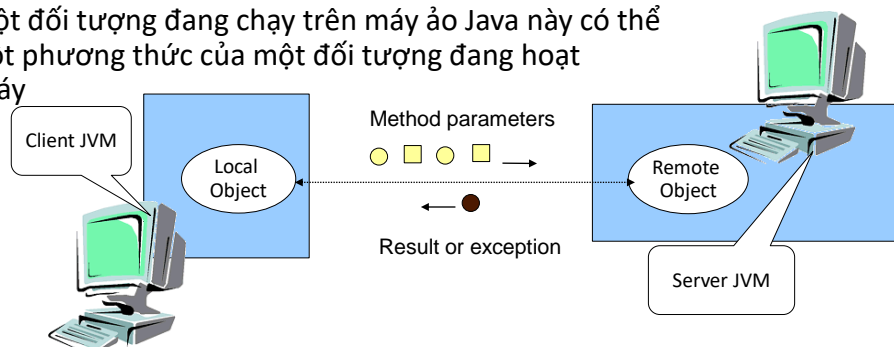


- RPC – Remote Procedure Call (gọi thủ tục từ xa) là một cơ chế cho phép một chương trình có thể gọi một thủ tục hay hàm trên một máy tính khác
 - Local Procedure: được định nghĩa, cài đặt và thực thi tại máy cục bộ
 - Remote Procedure: được định nghĩa, cài đặt và thực thi trên một máy tính khác
- Trong một hệ thống RPC
 - Server cung cấp các thủ tục ở xa, cho phép các chương trình trên máy khác triệu gọi
 - Client là các chương trình có thể triệu gọi các thủ tục ở xa trong quá trình tính toán của mình

Giới thiệu RMI



- RMI - Remote method Invocation (Gọi phương thức từ xa): là một sự cài đặt cơ chế RPC trong ngôn ngữ lập trình Java
 - Local method Invocation
 - Remote method Invocation
- Hệ thống RMI
 - Cho phép một đối tượng đang chạy trên máy ảo Java này có thể kích hoạt một phương thức của một đối tượng đang hoạt động trên máy ảo Java khác





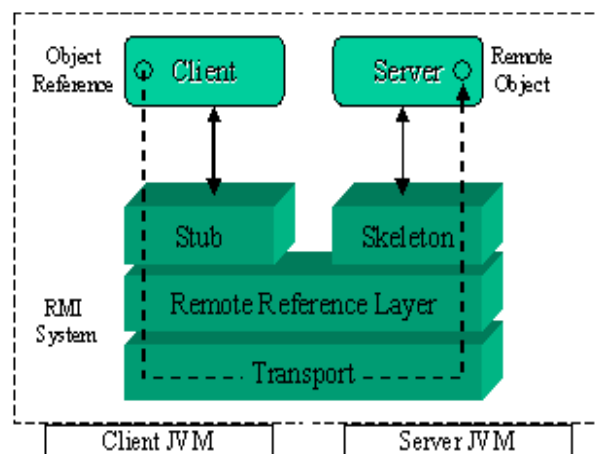
Đặc điểm của RMI

- RMI là một cơ chế truyền thông và là kỹ thuật thuần Java. Điều đó nghĩa là, RMI chỉ cho phép các đối tượng thuần Java mới gọi từ xa phương thức của nhau được
- Ứng dụng phân tán RMI cũng được tổ chức theo mô hình client/server
 - Phía Server chứa các đối tượng có phương thức cho phép triệu gọi từ xa
 - Phía Client chứa các đối tượng phát sinh lời gọi phương thức từ xa
- Một client có thể kích hoạt các phương thức từ xa trên 1 hay nhiều Server -> Sự thực thi của chương trình được trải rộng trên nhiều máy tính
 - Đặc điểm của các ứng dụng phân tán

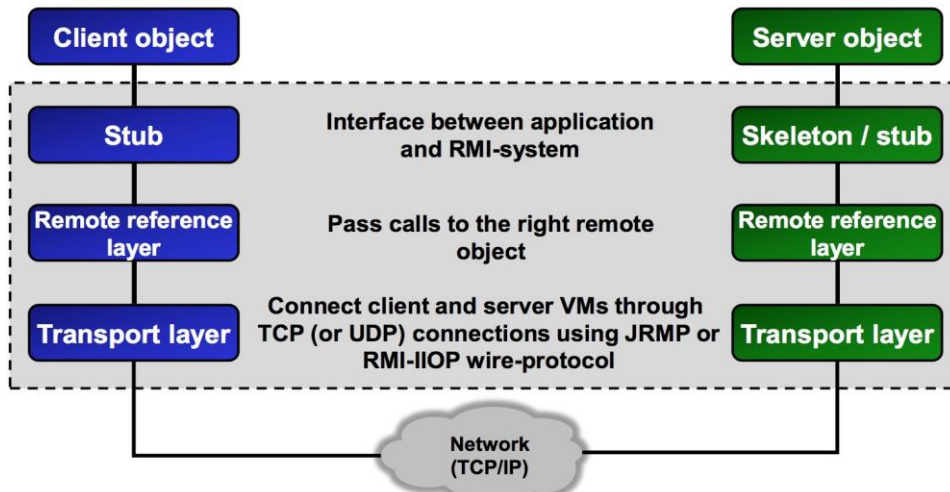
Kiến trúc RMI



- Không gọi trực tiếp mà thông qua lớp trung gian
- Kiến trúc một chương trình client/server theo cơ chế RMI gồm
 - Stub là đối tượng trung gian phía Client
 - Skeleton là đối tượng trung gian phía Server
 - Remote Reference Layer là lớp tham chiếu từ xa của RMI
- Lớp trung gian hỗ trợ thông báo khi có các sự cố về mạng
 - Một vấn đề của triệu gọi phương thức từ xa



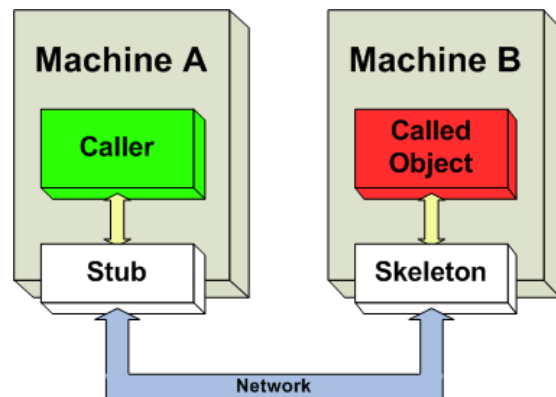
Kiến trúc RMI (Cont.)



1. *Locate remote objects*
2. *Communicate with remote objects*
3. *Load class bytecodes for objects that are passed around*

Stub và Skeleton

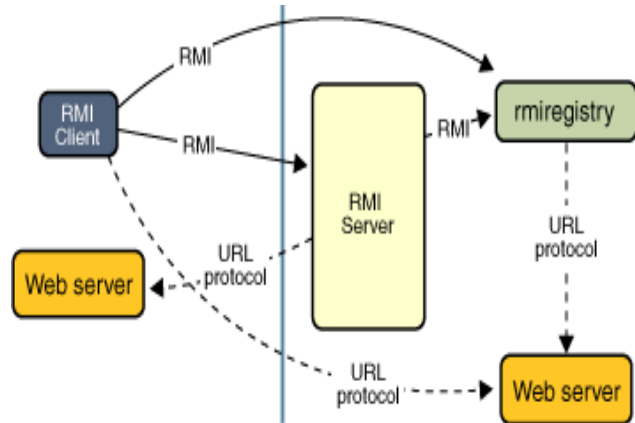
- Trình biên dịch `rmic.exe` tạo lớp trung gian Stub và Skeleton
- Caller thông qua Stub đóng gói lời gọi *procedure* và tham số đầu vào (*marshalls*) và truyền đi
- Skeleton nhận và mở gói lời gọi (*unmarshalls*) và tìm kiếm *procedure* ở Called Object và thực hiện trả về kết quả



1. *Locate remote objects*

Mô tả sử dụng dịch vụ danh bạ

- **Server:** Đăng ký tên cho đối tượng có thể được gọi từ xa của mình với dịch vụ danh bạ (registry server)
 - rmiregistry là một dịch vụ chạy ngầm, mặc định lắng nghe các yêu cầu ở cổng 1099
 - Đóng vai trò như một DNS nhỏ cung cấp dịch vụ tìm kiếm
- **Client:** Tìm đối tượng ở xa thông qua tên đã được đăng ký trên Registry Server (lookup) và tiếp đó gọi các phương thức ở xa

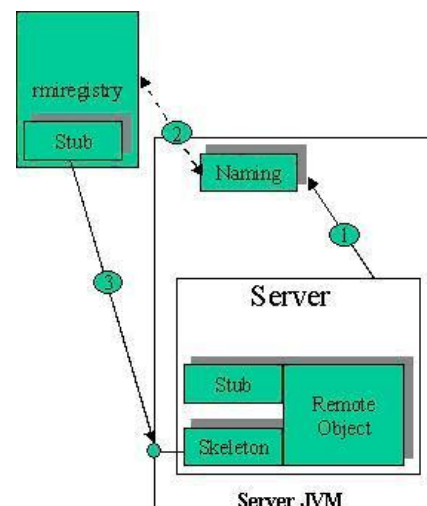


2. Communicate with remote objects

Tiến trình vận hành của một ứng dụng client/server theo RMI

Server cho phép tạo ra các đối tượng cho phép gọi từ xa cùng với các Stub và Skeleton của chúng

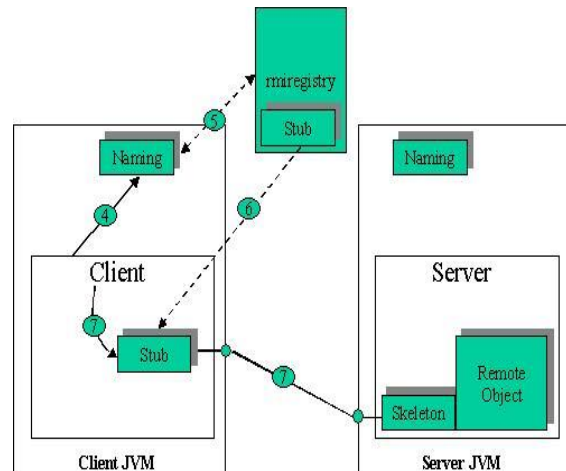
- **Bước 1:** Server sử dụng lớp Naming để đăng ký tên cho đối tượng từ xa (1)
- **Bước 2:** Naming đăng ký Stub của đối tượng từ xa với Registry Server (2)
- **Bước 3:** Registry Server sẵn sàng cung cấp tham khảo đến đối tượng từ xa khi có yêu cầu (3)



2. Communicate with remote objects

Tiến trình vận hành của một ứng dụng client/server theo RMI (Cont.)

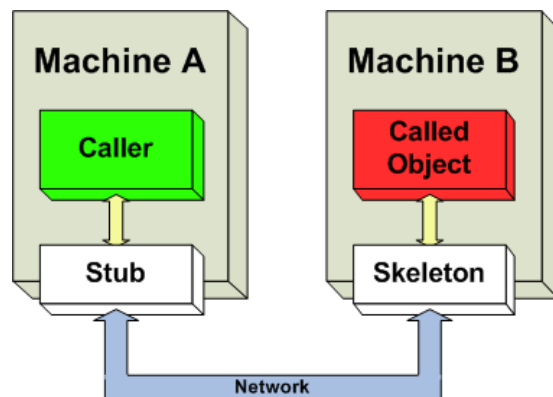
- **Bước 4:** Client gửi yêu cầu Naming định vị đối tượng xa qua tên đã được đăng ký (phương thức lookup) với dịch vụ tên (4)
- **Bước 5:** Naming tải Stub của đối tượng xa từ dịch vụ tên mà đối tượng xa đã đăng ký về Client (5)
- **Bước 6:** Cài đặt đối tượng Stub và trả về tham chiếu đối tượng xa cho Client (6)
- **Bước 7:** Client thực thi một lời gọi phương thức từ xa thông qua đối tượng Stub (7)



2. Communicate with remote objects

Hoạt động của Stub và Skeleton

- Khi Client gọi một phương thức từ xa, lời gọi này được chuyển tiếp đến Stub.
- Stub có nhiệm vụ gửi tiếp lời yêu cầu này đến Skeleton phía server bằng cách
 - Stub mở một socket đến server
 - Đóng gói các tham số: Gói nhận dạng đối tượng từ xa; Gói cách thức nhận dạng
 - Truyền luồng dữ liệu này đến Skeleton.
- Skeleton chứa đựng một phương thức nhận các lời yêu cầu từ xa,
 - Mở gói tham số
 - Gọi hàm thực sự trên server để tính toán
 - Trả kết quả về Stub phía client





2. Xây dựng ứng dụng phân tán RMI

Gói java.rmi



Package	Description
java.rmi.*	Core RMI package with classes and interfaces used by both client and server. Contains interface Remote, classes Naming and RMISecurityManager and some basic exception classes.
java.rmi.activation.*	Classes and interfaces for dynamic activation of remote objects together with RMI daemon (rmid). More information on dynamic invocation see below.
java.rmi.dgc.*	Classes and interfaces for distributed garbage collection (DGC).
java.rmi.registry.*	Registry and LocateRegistry classes for directly interacting with a (remote or local) registry. Registry class provides lookup(), rebind(), list() and other methods.
java.rmi.server.*	Classes for use on the server side like class loader (RMIClassLoader) and UnicastRemoteObject (base class for remote objects).
javax.rmi.*	APIs for RMI-IIOP (interoperability between RMI and CORBA).

Giao tiếp Remote và lớp Naming



- **Remote** Interface: Giao tiếp này không khai báo bất kỳ một phương thức nào. Các phương thức được khai báo trong giao tiếp này là các phương thức có thể gọi từ xa
- Lớp **java.rmi.Naming** trao đổi trực tiếp với một trình đăng ký đang chạy trên server để ánh xạ các URL [rmi://hostname:port/Objectname](#) thành các đối tượng từ xa xác định
 - rmi: là giao thức
 - hostname và port là địa chỉ và port của Server nơi trình đăng ký chạy
 - Objectname là tên đối tượng (tự đặt) – phía Client sẽ dựa vào tên này để truy tìm tham chiếu tới đối tượng cần dùng

Lớp Naming



- Một số phương thức
 - `public static String[] list(String url) throws RemoteException`
 - `public static Remote lookup(String url) throws RemoteException, NotBoundException, AccessException, MalformedURLException`
 - `Public static void bind(String url, Remote object) throws RemoteException, AlreadyBoundException, MalformedURLException, AccessException`
 - `public static void rebind(String url, Remote obj) throws RemoteException, AccessException, MalformedURLException`
- Một số ngoại lệ
 - `MalformedURLException`: url không đúng cú pháp.
 - `RemoteException`: không thể liên lạc được với trình đ.ký
 - `AccessException`: client bị từ chối hoặc không được phép thực hiện.
 - `AlreadyBoundException`: nếu đối tượng URL đã gắn với một đối tượng cục bộ

Registry Interface



- Interface Registry cho phép các máy khách tìm kiếm các đối tượng ở xa trên máy chủ
- Một số phương thức hữu ích
 - **Bind()** để gán một tên với một đối tượng từ xa cụ thể
 - **List()** liệt kê tất cả các tên đã được đăng ký với trình đăng ký
 - **Lookup()** tìm một đối tượng từ xa cụ thể với một URL cho trước gắn với nó
 - **Rebind()** gán một tên với một đối tượng ở xa khác
 - **Unbind()** loại bỏ một tên đã được gán cho một đối tượng ở xa trong trình đăng ký
 - **Registry.REGISTRY_PORT** là cổng mặc định để lắng nghe các yêu cầu. Giá trị mặc định là 1099

Cài đặt ứng dụng theo RMI



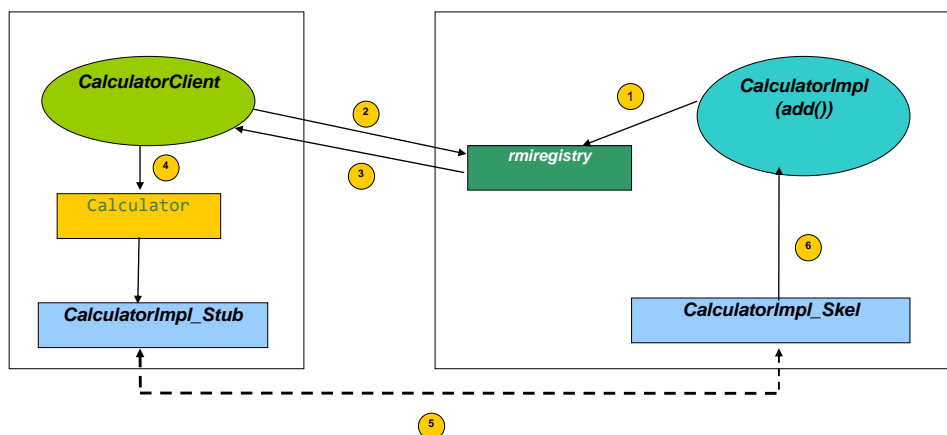
- Server
 - Đặc tả giao tiếp từ xa
 - Cài đặt giao tiếp từ xa
 - Tạo Stub và Skeleton
 - Xây dựng chương trình phía Server
- Client
 - Xây dựng chương trình phía Client
 - Tải Stub hoặc các lớp hỗ trợ cần thiết
- Thực thi
 - rmiregistry server
 - Server
 - Client



Ex:

- Chương trình máy tính đơn giản sử dụng RMI theo mô hình client/server
 - Server cung cấp phương thức gọi từ xa int **add(int a,int b)** để tính tổng 2 số
 - Client cho phép nhập và gọi phương thức từ xa **add** để tính tổng 2 số và hiển thị kết quả

Ex: máy tính đơn giản



Remote Interface



```
// Calculator.java
import java.rmi.*;
public interface Calculator extends Remote
{
    public int addNum(int a,int b) throws RemoteException;
}
```

Remote Class



1

```
/* CalculatorImpl.java*/
import java.rmi.*;
public class CalculatorImpl implements Calculator {
    public int add(int a,int b) throws RemoteException {
        System.out.println("Client request to calculate");
        return (a+b);
    }
}
```

2

```
public class CalculatorImpl extends UnicastRemoteObject implements Calculator {
```

Java 1.8?

Server



```

/* CalculatorServer.java */
public class CalculatorServer{
public static void main(String[] args) throws AlreadyBoundException {
    try{
        //tao doi tuong CalculatorImpl
        CalculatorImpl cal= new CalculatorImpl();
        System.out.println("Exporting Calculator ! ");
        //thong bao su hien dien của cal là doi tuong có khả năng remote cho JVM
        UnicastRemoteObject.exportObject (cal); // ???
        //dang ky doi tuong với trình quản lý rmi
        Naming.bind("rmi://localhost/cal", cal);
        System.out.println("Register Calculator!");
    } catch(Exception e)
        { System.out.println(e); }
    }
}

```

Client



```

/* CalculatorClient.java*/
import java.rmi.*;
public class CalculatorClient {
public static void main(String[] args){
    try{
        System.out.println("Finding Object ... ");
        // tìm doi tuong tu xa theo ten dang ky
        Calculator cal= (Calculator)Naming.lookup ("rmi://localhost/cal");
        // triệu gọi phương thức tu xa
        System.out.println(cal.add(10,20));
    } catch(Exception e) { System.out.println(e); }
    }
}

```



Case 1: Biên dịch

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\mam>G:

G:\>cd G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src
G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src>javac *.java
G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src>rmic CalculatorImpl
G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src>_
```

- Calculator.class
- Calculator.java
- CalculatorClient.class
- CalculatorClient.java
- CalculatorImpl.class
- CalculatorImpl.java
- CalculatorImpl_Stub.class
- CalculatorServer.class
- CalculatorServer.java

Thực thi chương trình



- Run rmiregistry
 - rmiregistry
 - start rmiregistry
- Run calculator server

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\mam>G:

G:\>cd G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src
G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src>start rmiregistry
G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src>_
```



Thực thi chương trình

```
C:\WINDOWS\system32\cmd.exe - java CalculatorServer
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\mam>G:

G:\>cd G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src
G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src>java CalculatorServer
Exporting Calculator !
Register Calculator!
client request a calculate

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\mam>G:

G:\>cd G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src
G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src>java CalculatorClient
Finding Object cal...
30
G:\TONGHOPDAY\CDIT\Java 01\Workspace\netbean\JNPLesson05RMI\src>
```

Case 2: (Java 1.8)



- Kế thừa `UnicastRemoteObject`
 - Không cần thực hiện `rmic CalculatorImpl` (static stub and skeleton)
 - Sử dụng `UnicastRemoteObject`
- Run
 - `CalculatorServer`
 - `CalculatorClient`
- Lưu ý có thể thay vì thực thi `rmiregistry` bằng
 - `LocateRegistry.createRegistry(1099);`



Case 2: Biên dịch

- Lưu ý phần warning

```

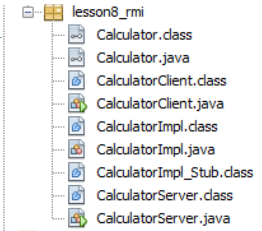
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.16299.125]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\mam>cd C:\jnp\nhom7\jnp7\src

C:\jnp\nhom7\jnp7\src>javac lesson8_rmi\*.java

C:\jnp\nhom7\jnp7\src>rmic lesson8_rmi.CalculatorImpl
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

C:\jnp\nhom7\jnp7\src>
    
```



Case 2: Run



```

C:\WINDOWS\system32\cmd.exe - java -classpath C:\jnp\nhom7\jnp7\src lesson8_rmi.CalculatorServer

C:\Users\mam>java -classpath C:\jnp\nhom7\jnp7\src lesson8_rmi.CalculatorServer
registered calculator!
client request
    
```

- Lưu ý có thể thực thi

- Cmd rmiregistry hoặc
- Mã nguồn
LocateRegistry.createRegistry(1099);

```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.16299.125]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\mam>java -classpath C:\jnp\nhom7\jnp7\src lesson8_rmi.CalculatorClient
30

C:\Users\mam>
    
```


Demo



- **RMI**

- Calculator
- CalculatorImpl
- CalculatorServer
- CalculatorClient

- **Run**

- rmiregistry
- Server
- Client



3. Một số vấn đề RMI



Prob 1: tự tải stub cho máy khách

- **Lớp Stub được sinh ra để phục vụ cho cơ chế của RMI**

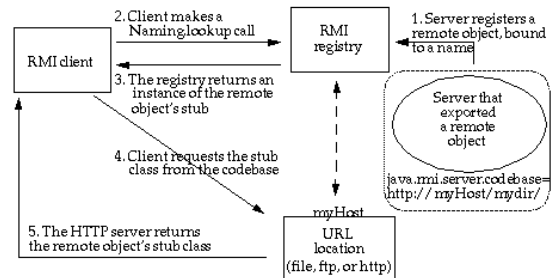
- Khi lớp Stub thay đổi
- Nhà phát triển cơ bản không cần quan tâm đến lớp Stub khi phát triển chương trình Client

- **Ý tưởng**

- Sử dụng codebase để tự động tải lớp Stub
- Yêu cầu phải hỗ trợ file, ftp, http

- **Lưu ý:**

- *codebase có thể được sử dụng để tải bất kỳ một lớp nào, không chỉ riêng lớp Stub*
- *Chính sách bảo mật ở Client, lớp RMISecurityManger*



3. Load class bytecodes for objects that are passed around

Prob 2: Truyền tham số



- **Nguyên tắc**

- Đối với kiểu dữ liệu nguyên thủy int, char ... Truyền theo kiểu tham trị
- Đối với kiểu dữ liệu tham chiếu, đối tượng muốn truyền đi bắt buộc phải cài đặt interface
 - Remote, Truyền theo kiểu tham chiếu
 - Serializable, Truyền theo kiểu tham trị

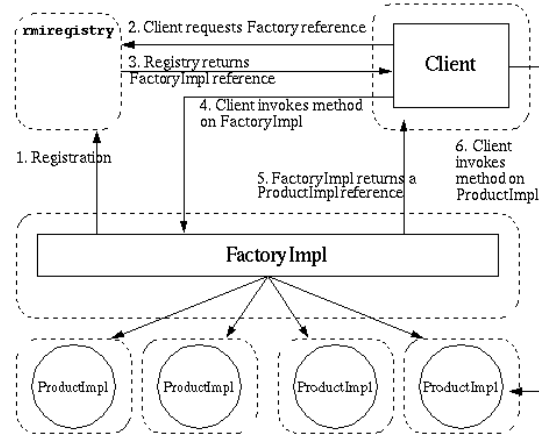
- **Ý tưởng**

- Thực hiện các lời gọi phương thức từ xa có tham số truyền vào là đối tượng cho 02 trường hợp cài đặt Interface
 - Remote
 - Serializable
- **Đánh giá**
 - **Remote** (tham chiếu)
 - **Serializable** (tham trị)



Prob 3: Quá nhiều đối tượng

- Khi xây dựng lớp đối tượng truy cập từ xa -> thực hiện đặt tên, đăng ký với rregistry
- Vấn đề
 - Rmregistry quản lý nhiều đối tượng
 - Client bắt buộc phải truyền đúng tên các đối tượng này
- Ý tưởng
 - Xây dựng một đối tượng duy nhất (**Factory object**) đăng ký với rregistry cho mỗi ứng dụng
 - Đối tượng này có nhiệm vụ tạo ra các đối tượng con khác



Exercise



- Prob 1
- Prob 2
- Prob 3
- RMI
 - Server: cài đặt một chương trình hoàn chỉnh
 - Client: không cần cài đặt, thực hiện lời gọi server
- Ý tưởng RMIClassLoader
 - Cho phép nạp về mã nguồn thực thi của toàn bộ chương trình Client
 - Thực thi
- Lưu ý vấn đề security
 - Java
 - Network

Tổng kết



- RMI trong Java
- Các bài toán phân tán

Case study: Login sử dụng RMI



...



Q & A