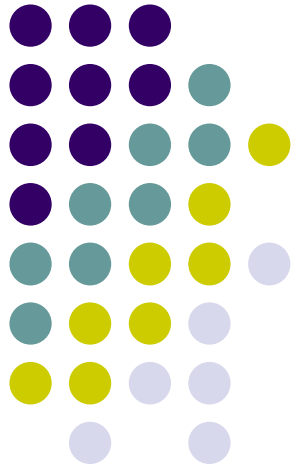
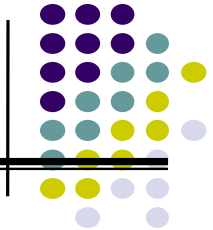
RPC và RMI

- Khái niệm RPC
- Khái niệm RMI
- Các bước cài đặt RMI trong Java
- Ví dụ về RMI



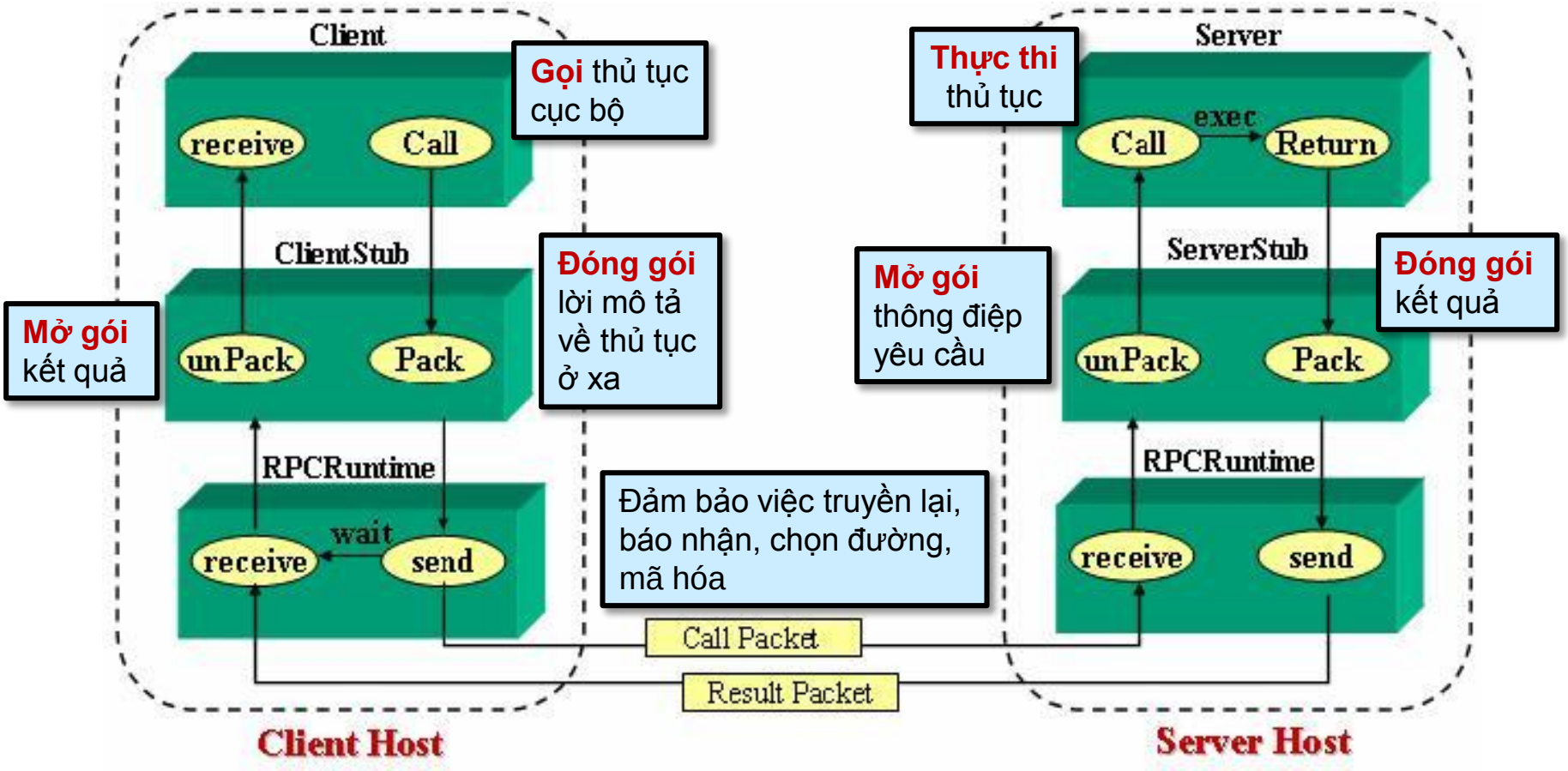
RPC (Remote Procedure Call)



- Khái niệm
  - RPC: gọi thủ tục ở xa.
  - Trong suốt về mặt ngữ nghĩa: gọi thủ tục ở xa cũng có cú pháp tương tự như gọi thủ tục cục bộ.
  - Định hướng lời gọi đến máy tính đích ở xa thông qua khái niệm Stub.
  - Đơn giản hóa việc xây dựng các ứng dụng Client-Server
    - Server : cung cấp các thủ tục ở xa
    - Client : gọi các thủ tục ở xa trong quá trình tính toán của mình.
  - Mô hình của ứng dụng phân tán (Distributed Application):
    - Thực thi của chương trình được trải rộng ra trên nhiều máy tính khác nhau.

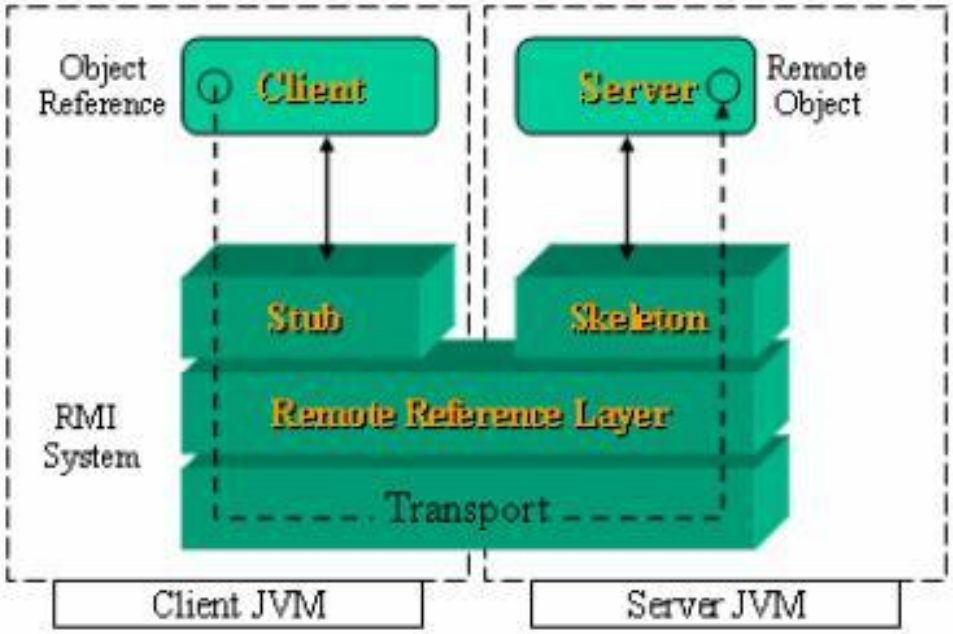
RPC (Remote Procedure Call)

- Kiến trúc chương trình



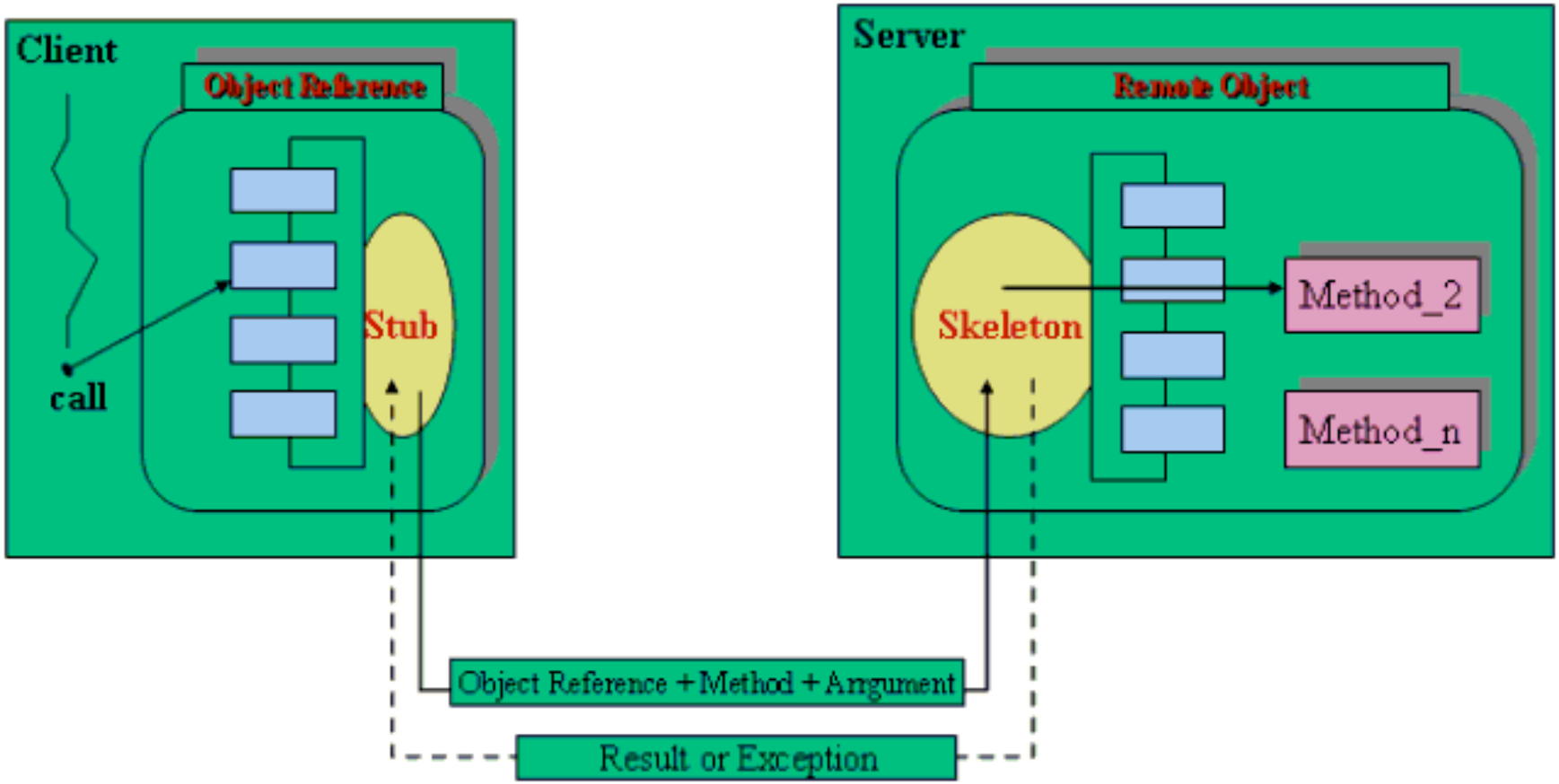
RMI (Remote Method Invocation)

- Khái niệm
  - Cài đặt RPC bằng ngôn ngữ Java.
  - Cho phép 1 phương thức thực thi từ xa trên nhiều máy ảo khác nhau.
- Kiến trúc ứng dụng



RMI (Remote Method Invocation)

- Con đường kích hoạt 1 phương thức ở xa



RMI (Remote Method Invocation)

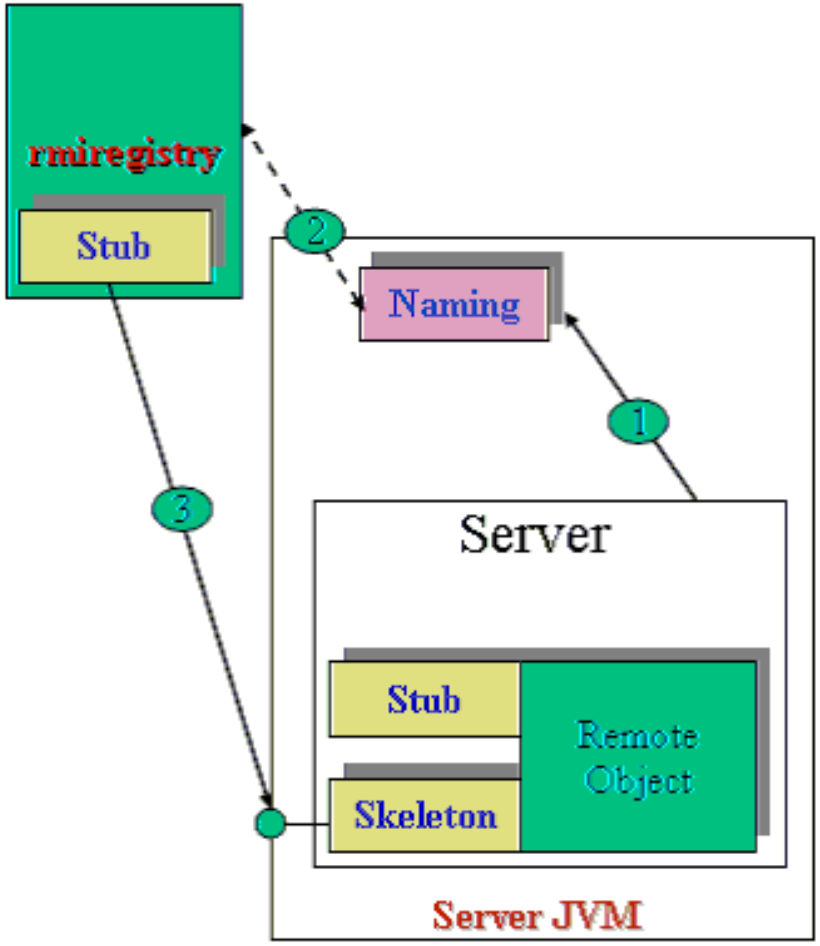
- Cơ chế vận hành của 1 ứng dụng Client-Server theo kiểu RMI

• **Bước 0**: Server **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 một đối tượng từ xa.

• **Bước 2**: Naming **đăng ký Stub** của đối tượng từ xa với Registry Server.

• **Bước 3**: Registry Server **sẵn sàng cung cấp tham thảo** đến đối tượng từ xa khi có yêu cầu.



RMI (Remote Method Invocation)

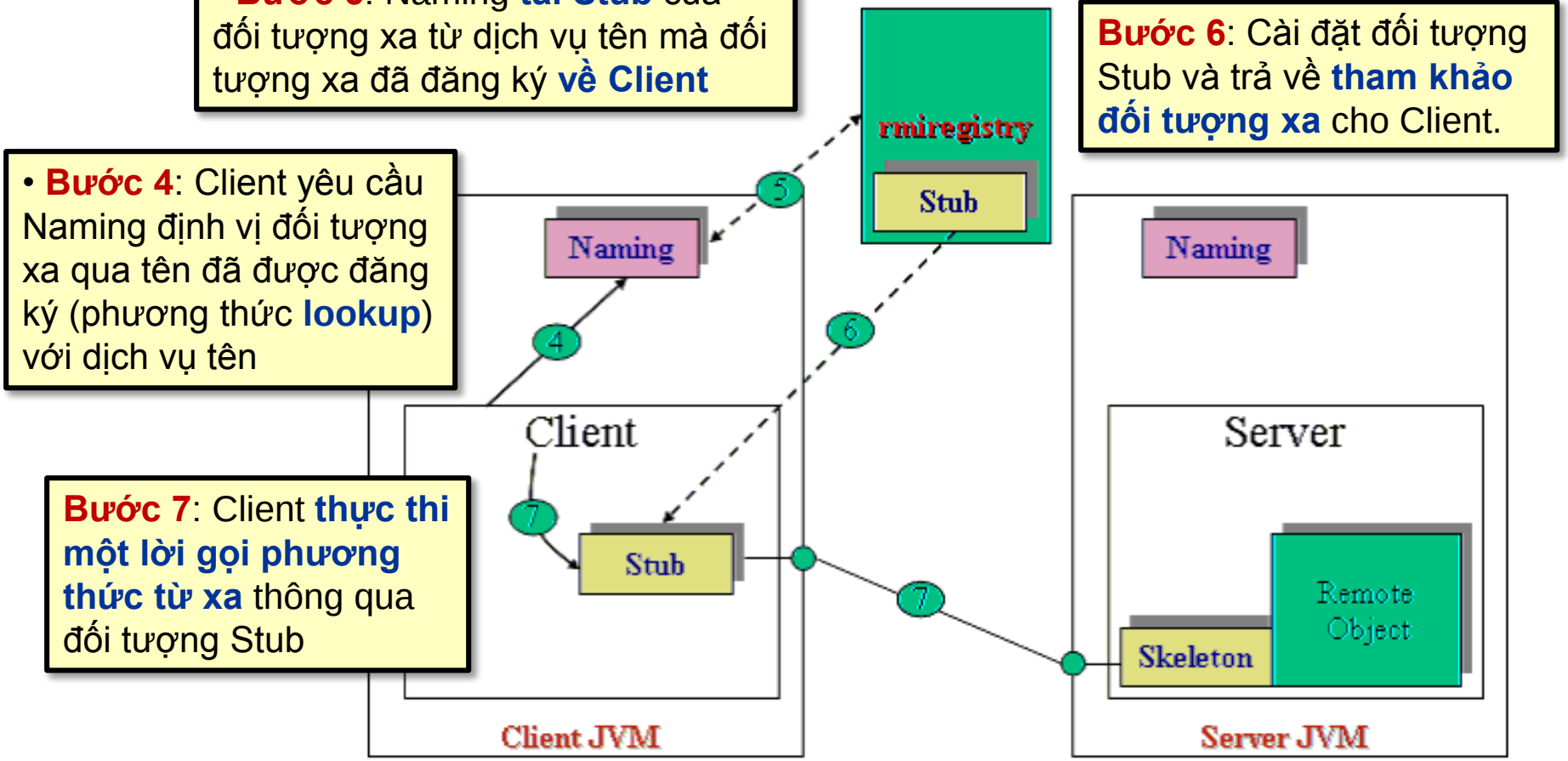
- Cơ chế vận hành của 1 ứng dụng Client-Server theo kiểu RMI

• **Bước 5**: Naming **tải Stub** của đối tượng xa từ dịch vụ tên mà đối tượng xa đã đăng ký **về Client**

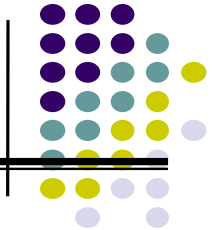
Bước 6: Cài đặt đối tượng Stub và trả về **tham khảo đối tượng xa** cho Client.

• **Bước 4**: Client yêu cầu Naming định vị đối tượng xa qua tên đã được đăng ký (phương thức **lookup**) với dịch vụ tên

Bước 7: Client **thực thi một lời gọi phương thức từ xa** thông qua đối tượng Stub

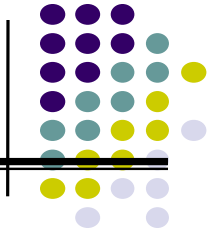


RMI (Remote Method Invocation)



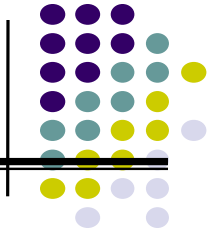
- Các lớp của Java hỗ trợ xây dựng ứng dụng RMI
  - Các gói **java.rmi** và **java.rmi.Server**
  - Các lớp thường dùng là:
    - java.rmi.Naming
    - java.rmi.RMISecurityManager
    - java.rmi.RemoteException
    - java.rmi.Server.RemoteObject
    - java.rmi.Remote
    - java.rmi.Server.UnicastRemoteObject

RMI (Remote Method Invocation)



- Xây dựng ứng dụng phân tán với RMI
  1. Tạo giao diện (interface) khai báo các phương thức được gọi từ xa của đối tượng.
  2. Tạo lớp cài đặt (implements) cho giao diện đã được khai báo.
  3. Viết chương trình Server.
  4. Viết chương trình Client.
  5. Dịch các tập tin nguồn theo dạng RMI (rmic) để tạo ra các lớp tương ứng và stub cho Client, skeleton cho Server.
  6. Khởi động dịch vụ registry (rmiregistry).
  7. Thực hiện chương trình Server.
  8. Thực thi chương trình Client.

RMI (Remote Method Invocation)

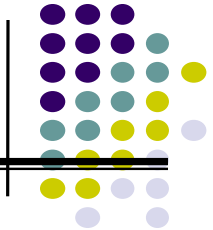


- **Ví dụ minh họa**:

  Xây dựng ứng dụng phân tán theo dạng RMI:

  - Định nghĩa 1 phương thức String sayHello() cho phép gọi từ xa.
  - Khi kích hoạt phương thức đó: sẽ trả về kết quả là 1 chuỗi "*Hello RMI*" cho Client gọi nó.

RMI (Remote Method Invocation)



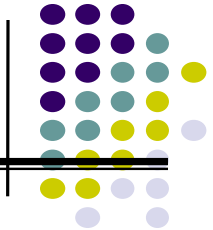
- Bước 1: Tạo **interface** khai báo phương thức từ xa

```
import java.rmi.Remote;
import java.rmi.RemoteException;

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

- Bước 2: Tạo **lớp cài đặt** cho interface khai báo ở trên

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

public class  Hello extends UnicastRemoteObject implements  HelloItf  {
    public Hello() throws RemoteException {
        super();
    }
    // Cai dat ham cho phep goi tu xa
    public String  sayHello() {
        return  "Hello RMI";
    }
}
```

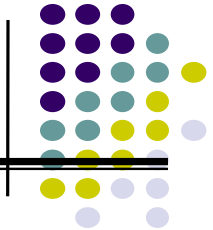
RMI (Remote Method Invocation)



- Bước 3: Chương trình **Server**, tạo đối tượng cho phép gọi hàm từ xa.

```
import java.rmi.*;
import java.net.MalformedURLException;
public class  HelloServer {
    public static void main(String[] args)  {
        if(System.getSecurityManager()==null)   // Cai dat co che bao mat
            System.setSecurityManager(new RMISecurityManager());
        try{
            // Tao ra doi tuong cho phep goi ham tu xa
            Hello  obj = new Hello();
            System.out.println("Tao duoc doi tuong cho phep goi tu xa");
            // Dang ky doi tuong
            Naming.rebind("HelloObject", obj);
            System.out.println("Da dang ky doi tuong de goi tu xa thanh cong !!!");
        }
        catch(RemoteException  e)
            { System.out.println("Loi trong khi khoi tao doi tuong tu xa"); }
        catch(MalformedURLException  e)
            { System.out.println("Loi trong khi dang ky doi tuong tu xa"); }
    }
}
```

RMI (Remote Method Invocation)



- Bước 4: Chương trình **Client**, tạo đối tượng tham chiếu đến đối tượng từ xa, gọi hàm từ xa trên đối tượng đó.

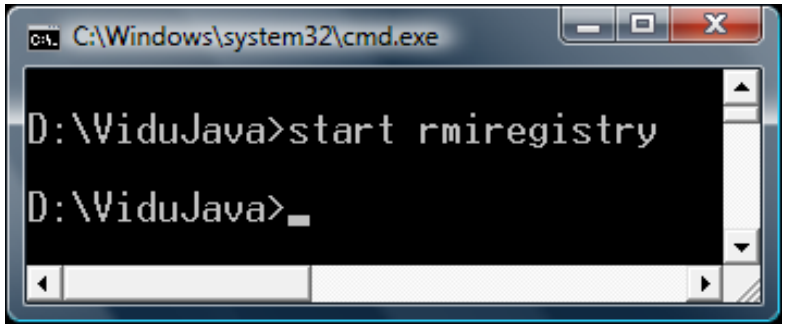
```
import java.rmi.*;
import java.net.MalformedURLException;
public class  HelloClient  {
    public static void main(String[] args)  {
        try{
            // Do tim doi tuong tu xa
            HelloItf  ref = (HelloItf) Naming.lookup("rmi://172.18.211.160/HelloObject");
            // Goi ham tren doi tuong
            String  result = ref.sayHello();
            System.out.println("Ket qua nhan duoc la: " + result);
        }
        catch(NotBoundException  e)
         {  System.out.println("Khong tim thay doi tuong goi tu xa"); }
        catch(MalformedURLException  e)
            { System.out.println("Sai trong dinh dang URL"); }
        catch(RemoteException  e)
         {  System.out.println("Loi trong khi goi ham tu xa") ; }


    }
}
```

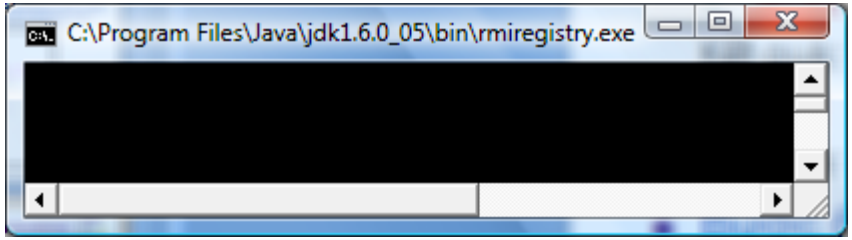
RMI (Remote Method Invocation)

- Bước 5: Dịch các tập tin nguồn theo dạng RMI
  - javac HelloItf.java Hello.java HelloServer.java HelloClient.java

  Kết quả: HelloItf.class, Hello.class, HelloServer.class, HelloClient.class
  - rmic Hello

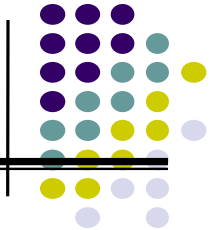
  Kết quả: ~~Hello_Skel.class~~, Hello_Stub.class
- Bước 6: Khởi động dịch vụ rmiregistry
  - start rmiregistry [port]
  - Cổng mặc định là 1099.





Không đóng cửa sổ này lại

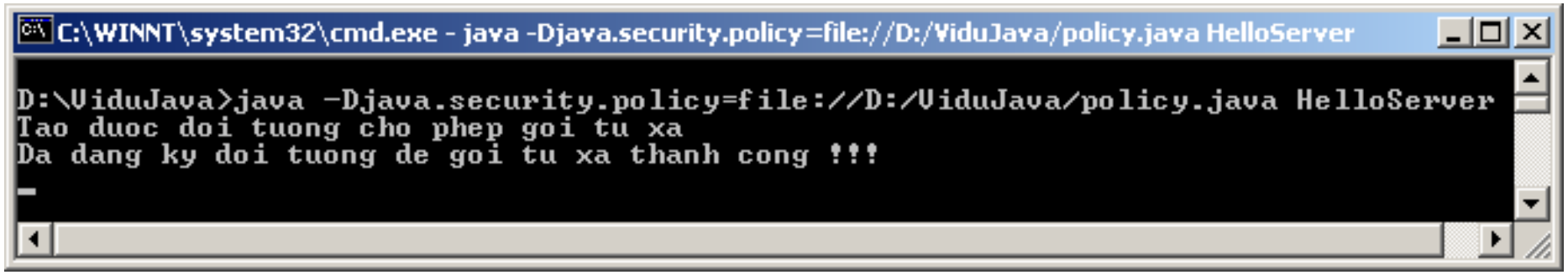
RMI (Remote Method Invocation)



- Bước 7: Thực thi Server
  - java -Djava.rmi.server.hostname=IP_Of_Server -Djava.security.policy=URL_Of_PolicyFile ServerName
  - URL_Of_PolicyFile: địa chỉ theo dạng URL của tập tin mô tả chính sách về bảo mật mã nguồn của Server (policy file).
  - File có dạng sau (*tham khảo thêm về tài liệu Security của Java*):

```
grant {
    // Giả sử cho phép mọi người đều được download tập tin trên Server
    permission java.security.AllPermission;
};
```

Giả sử tập tin policy nằm ở thư mục D:\ViduJava\policy.java



RMI (Remote Method Invocation)

- Bước 8: Thực thi Client