Lab3: Intersystem communication protocols

Contents

TCP/IP sockets 2
InetAddress Class 2
URL Class 4
URL Connection Class 5
Lập trình socket 8
RMI 10
Đặc tính của RMI 10
RMI Architecture 10
Chương trình duyệt các đối tượng có trong RMIRegistry 11
Tạo bộ đăng ký cục bộ 12
Remote Method Invocation 13
Ứng dụng RMI 17
Bài tập thực hành 20
WSDL 22
Tài liệu tham khảo 23

TCP/IP sockets

InetAddress Class

Java.net.InetAddress biểu diễn một địa chỉ Internet. Bao gồm 2 trường thông tin: hostName (kiểu string) và address (một số kiểu int). Các trường này không phải là trường public, vì thế ta không thể truy xuất chúng trực tiếp

Java.net.InetAddress được sử dụng bởi hầu hết các lớp mạng, bao gồm: Socket, ServerSocket, URL, DatagramSocket, DatagramPacket,…

Với lập trình mạng thì dùng gói: java.net.*

Trong bài này thì chúng ta quan tâm tới lớp InetAddress và nó cũng có 2 lớp con cho IPv4 và IPv6.

Biểu diễn 1 địa chỉ IP trên mạng với 2 trường là: HostName kiểu string và Address kiểu int.

Không có constructor nên muốn tạo đối tượng thì chúng ta thông qua phương thức static là: InetAddress.getByName(). Tham số truyền vào là 1 tên trang web hoặc 1 địa chỉ IP gì cũng đc.

Ví dụ 1: Viết chương trình nhận hostname từ đối dòng lệnh và in ra địa chỉ IP tương ứng với hostname đó.

```
package IPAddress;

import java.net.InetAddress;
import java.net.UnknownHostException;

public class TimDCIP {
	public static void main(String[] args)
	{
		try{
			InetAddress host = InetAddress.getByName(args[0]);
			String hostName = host.getHostName();
			System.out.println("Host name:"+hostName);
			System.out.println("Dia chi IP:"+host.getHostAddress());
		}
		catch(UnknownHostException e)
		{
			System.out.println("Khong tim thay dia chi");
			return;
		}

	}

}
```

Ví dụ 2: Viết chương trình nhập một hostName từ đối dòng lệnh và in ra dòng thông báo cho biết địa chỉ IP tương ứng với địa chỉ IP đó thuộc lớp nào.

```
import java.net.*;
public class  PhanLoaiDCIP
{
	public static void main(String[] args)
	{
		try{
			if(args.length!=1)
			{
				System.out.println("Cach su dung: java TimDCIP <Hostname>");
			}
			InetAddress host = InetAddress.getByName(args[0]);
			String hostName = host.getHostName();
			System.out.println("Host name:"+hostName);
			System.out.println("Dia chi IP:"+host.getHostAddress());
			byte[] b=host.getAddress();
			int i=b[0]>=0?b[0]:256+b[0];

			if((i>=1)&(i<=126)) System.out.println(host+" thuoc dia chi lop A");
			if((i<=191)&(i>=128)) System.out.println(host+" thuoc dia chi lop B");
			if((i<=223)&(i>=192)) System.out.println(host+" thuoc dia chi lop C");

		}
		catch(UnknownHostException e)
		{
			System.out.println("Khong tim thay dia chi");
			return;
		}

	}
}
```

URL Class

Vd: Viết chương trình nhập vào một URL từ đối dòng lệnh và hiển thị từng thành phần tạo nên URL lên màn hình

```
import java.net.MalformedURLException;
import java.net.URL;

public class getURLParts {
	public static void main(String[] args)
	{
		try
		{
			URL u = new URL(args[0]);
			System.out.println("URL is "+u);
			System.out.println("The protocol part is "+u.getProtocol());
			System.out.println("The host part is "+u.getHost());
			System.out.println("The file part is "+u.getFile());
			System.out.println("The reference part is "+u.getRef());
		}
		catch(MalformedURLException e)
		{
			System.err.println(e);
		}
	}

}
```

Vd: Viết chương trình nhập một URL từ bàn phím, kết nối với Internet và hiển thị mã nguồn của trang Web đó lên màn hình.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;

public class viewSource {

	public static void main(String[] args)
	{
		URL u;
		String thisLine;

		if(args.length>0){
			try{
				u = new URL(args[0]);
				try{
					BufferedReader br = new BufferedReader(new
InputStreamReader(u.openStream())); //thay the cho DataInputStream
```

```
                                        while((thisLine = br.readLine())!= null)
                                                System.out.println(thisLine);

                                }
                                catch(IOException e)
                                {
                                        System.err.println(e);
                                }
                        }
                        catch(MalformedURLException e){
                                System.err.println(e);
                        }
                }

        }

}
```

URL Connection Class

Là một lớp trừu tượng biểu diễn 1 liên kết tích cực tới 1 tài nguyên xác định bởi 1 URL Nó có 2 mục đích như thế này :

1) Với nó thì cung cấp cho chúng ta nhiều khả năng điều khiển hơn URL , qua việc tương tác với server. Có thể kiểm tra các headerMINE, download các file nhị phân, gởi dữ liệu trở lại server qua POST

2) Nó là 1 phần của cơ chế quản lý giao thức, cớ chế này còn bao gôm cả lớp URLStreamHandler.Ý tưởng đằng sau các trình quản trị giao thức rất đơn giản: chúng cho phép bạn phân tách các chi tiết xữ lý một giao thức với việc xử lý các kiểu dữ liệu cụ thể, cung cấp các giao diện người dùng và thực hiện các công việc khác mà một trình duyệt thường làm. Lớp cơ sở URLConnection là 1 lớp trường tượng, để cài đặt một giao thức cụ thể bạn cần phải viết một lớp con, các lớp con này có thể đc tải bởi các ứng dụng của riêng bạn hay bởi các trình duyệt Hotjava; trong tương lai, các ứng dụng java có thể tải về các trình quản lý giao thức khi cần

Mở 1 URLConnection

```
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

public class getUrlConnection {
    public static void main(String[] args) {
        URL u;
        URLConnection uc;
        try {
            u = new URL("http://iuh.edu.vn");
            uc=u.openConnection();
        } catch (MalformedURLException e) {
```

```
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Đọc dữ liệu từ server

Vd1: download mã nguồn của một trang web

```
import java.net.*;
import java.io.*;

public class SourceViewer2 {

  public static void main (String[] args) {
    if  (args.length > 0) {
      try {
        //Open the URLConnection for reading
        URL u = new URL(args[0]);
        URLConnection uc = u.openConnection( );
        InputStream raw = uc.getInputStream( );
        InputStream buffer = new BufferedInputStream(raw);

        // chain the InputStream to a Reader
        Reader r = new InputStreamReader(buffer);
        int c;
        while ((c = r.read( )) != -1) {
          System.out.print((char) c);
        }
      }

      catch (MalformedURLException ex) {
        System.err.println(args[0] + " is not a parseable URL");
      }
      catch (IOException ex) {
        System.err.println(ex);
      }
    }
  }
}
```

Vd2: Đọc các trường header của trang web

```
import java.net.*;
import java.io.*;
import java.util.*;

public class HeaderViewer {

  public static void main(String args[]) {

    for (int i=0; i < args.length; i++) {
      try {
        URL u = new URL(args[i]);
        URLConnection uc = u.openConnection( );
```

```
        System.out.println("Content-type: " + uc.getContentType( ));
        System.out.println("Content-
encoding: " + uc.getContentEncoding( ));
        System.out.println("Date: " + new Date(uc.getDate( )));
        System.out.println("Last modified: " + new Date(uc.getLastModified(
 )));
        System.out.println("Expiration date: " + new Date(uc.getExpiration(
 )));
        System.out.println("Content-length: " + uc.getContentLength( ));
      }

      catch (MalformedURLException ex) {
        System.err.println(args[i] + " is not a URL I understand");
      }

      catch (IOException ex) {
        System.err.println(ex);
      }
      System.out.println( );
    }
  }
}
```

Vd3: Lấy tất cả các header

```
import java.net.*;
import java.io.*;

public class AllHeaders {

  public static void main(String args[]) {

    for (int i=0; i < args.length; i++) {
      try {
        URL u = new URL(args[i]);
        URLConnection uc = u.openConnection( );
        for (int j = 1; ; j++) {
          String header = uc.getHeaderField(j);
          if (header == null) break;
          System.out.println(uc.getHeaderFieldKey(j) + ": " + header);
        }
      }

      catch (MalformedURLException ex) {
        System.err.println(args[i] + " is not a URL I understand.");
      }
      catch (IOException ex) {
        System.err.println(ex);
      }
      System.out.println( );
    }
  }
}
```

Lập trình socket

MyClient

```
import java.io.IOException;
import java.io.PrintStream;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;

public class MyClient {
	public static void main(String[] args) {
		try {
			Socket client = new Socket("LocalHost", 9540);
			System.out.println("Client da duoc tao");

			Scanner inFromServer = new Scanner(client.getInputStream());
			PrintStream outToServer = new PrintStream(client.getOutputStream());

			System.out.println("server: " + inFromServer.nextLine());

			Scanner scan = new Scanner(System.in);
			String ten = scan.nextLine();

			outToServer.println(ten);

			System.out.println("server: " + inFromServer.nextLine());

		} catch (UnknownHostException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}
}
```

My Server

```
import java.io.IOException;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class MyServer {
	public static void main(String[] args) {
```

```
		try {
			ServerSocket server = new ServerSocket(9540);
			System.out.println("Server da duoc tao");

			Socket client = server.accept();
			System.out.println("Client da ket noi den server");

			Scanner inFromClient = new Scanner(client.getInputStream());
			PrintStream outToClient = new PrintStream(client.getOutputStream());

			outToClient.println("Xin chao, ban ten gi?");

			String ten = inFromClient.nextLine();
			System.out.println("clien: " + ten);

			outToClient.println("Rat vui duoc tro chuyen voi " + ten);

		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
}
```

RMI

Remote Method Invoke – Triệu gọi phương thức từ xa

RMI - Remote Method Invocation là một kĩ thuật cài đặt các đối tượng phân tán trong Java. RMI là một phần của bộ J2SDK và là hàm thư viện hỗ trợ các lời gọi phương thức từ xa và trả về giá trị cho các ứng dụng tính toán phân tán. Chúng ta giả sử rằng ngôn ngữ Java được sử dụng ở cả hai phía gọi và phía bên phương thức được gọi.

Để giải quyết một số vấn đề trong việc truyền thông giữa Client/Server. RMI không gọi trực tiếp mà thông qua lớp trung gian. Lớp này tồn tại ở cả hai phía Client và Server. Lớp ở máy Client gọi là Stub, lớp ở máy Server gọi là Skel (Skeletion)

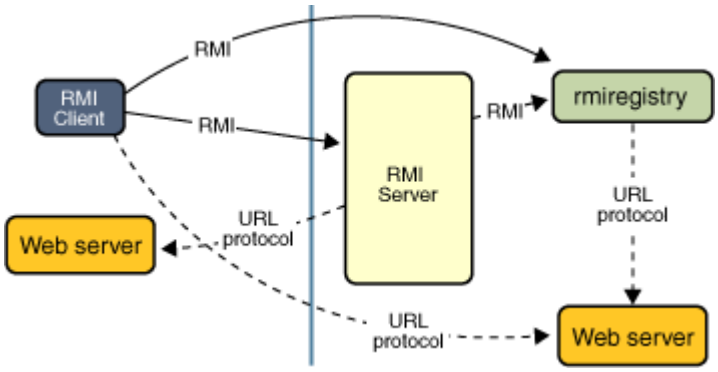
Đặc tính của RMI

- RMI là mô hình đối tượng phân tán của Java, RMI giup cho việc giao tiếp giuỗ các đối tượng phân tán trong môi trường internet trể nên dễ dàng hơn.
- RMI là API bậc cao được xây dựng dựa trên lập trình Socket.
- RMI không những cho phép chúng ta truyền dữ liệu giữa các đối tượng trên các hệ thống máy tính khác nhau, mà còn triệu gọi các phương thức trong các đối tượng ở xa (Remote Object).
- Việc truyền dữ liệu giữa các máy khác nhau được xử lý một cách trong suốt bởi máy ảo Java (Java virtual machine).
- Tương tự như mô hình Client/Server, RMI vẫn lấy/duy trì khái niệm của Client và Server, tuy nhiên cách tiếp cận (approach) của RMI linh hoạt hơn, mềm dẻo hơn so với môt hình Client/Server.
- Một điều thuận lợi quan trong nhất của RMI là nó cung cấp cớ chế callbacks, nó cho phép Server triệu gọi các phương thức ở Client.

RMI Architecture

- **Remote interface**: Nên extend từ java.rmi.remote. Nó khai báo tất cả các phương thức mà Client có thể triệu gọi. Tất cả các method trong interface này nên throw RemoteException
- **Remote implementation**: Được thực thi từ Remote interface và mở rộng từ UnicastRemoteObject. Triển khai các method được khai báo trong Interface tại đây. Nó là một Remote Object thực sự. Phát sinh hai lớp trung gian Stub và Skel.
- **Server class bao gồm**:
  - RMI registry: Bộ đăng kí này sẽ đăng kí một Remote object với Naming Registry. Giúp các Remote object được chấp nhận khi gọi các method từ xa.
  - Các class thực thi trên server.

- **Client class**: Truy vấn trên tên Remote object trên RMI registry, thông qua stub để gọi các phương thức trên server.



RMI Architecture (Ảnh: docs.oracle.com)

Chương trình duyệt các đối tượng có trong RMIRegistry

Bước 1: Xây dựng chương trình duyệt rmiregistry.

ví dụ: Traverse.java

import java.rmi.registry.*;

Pubic calss Traverse{

Public static void main (String[] args) throws Exception{

String hostAddr=”127.0.0.1”;

System.out.println(“connecting registry…”);

Registry registru\y=LocateRegistry.getregistry(hostAddr);

String objectAvailable[]=registry.list();

System.out.println(“Registry object:”);

For(int i=0;i<objectAvailable.lengh;i++){

System.out.println(“objectAvailable[i]”);

} } }

Biên dịch chương trình : Javac Traverse.java

Bước 2: Khởi động và đăng ký một số đối tượng rmiregistry.

Khởi động bộ đăng ký: **Start rmiregistry**

Đăng ký đối tượng Calculator với bọ đăng ký: **C:\rmi\Calculator>java Setup**

Đăng ký đối tượng Hello với bộ đăng ký : **C:\rmi\ByValue>java Setup**

Bước 3: Chạy chương trình Traverse:

c:\rmi\Traverse>java Traverse

myCalculator

myhello

pingobject

Bạn sẽ liệt kê được tổng cộng 3 đối tượng đã đăng ký với rmiregistry. Chương trình của tái sử dụng lớp LocateRegistry. Chứa trong gói java.rmi.registry. Bạn cung cấp địa chỉ Ip của máy ảo nơi rmiregistry đang chạy, phương thức tĩnh GetRegistry() của lớp này sẽ trả về đối tượng registry.

Registry registry=LocateRegistry.getRegistry(hostAddr);

Tạo bộ đăng ký cục bộ

Java cho phép bạn tự tạo bộ đăng cho riêng mình mà không cần dùng đến chương trình rmiregistry.exe. Để tạo bộ đăng ký và tự đăng ký đối tượng bạn gọi phương thức tĩnh createRegistry() của lớp LocateRegistry. Chẳng hạn chương trình setup.java ở ví dụ trên có thể tự đăng ký đố tượng hello theo cách sau:

Ví dụ: Setup.java

```
Import java.rmi.*;

Import java.rmi.server.*;

Import java.rmi.registry.*;

Public class setup{

Public static void main(String[]args) throws Exception{

LocateRegistry.createRegistry(1099);

helloImpl hello=new HelloImpl();

UnicastRemoteObject.exportObject(hello);

System.out.println("Registering object…");

Naming.bind("rmi:/localhost/myhello",hello);

System.out.println("Waiting for client request…");

} }
```

Bạn chỉ cần biên dịch và gọi chương trình java setup của ví dụ trên từ máy chủ. Setup sẽ tự động tạo bộ đăng ký rmiregistry ở cổng mặc định 1099 sau đó tự đăng ký đối tượng với rmiregistry do mình

tạo ra. Tuy nhiên cách này chỉ có thể áp dụng một đối tượng. Nếu một đối tượng nào đó đã tạo ra bộ đăng ký rồi thì bạn không được goi LocateRegistry.createRegistry() lần thứ hai trên cùng một cổng của máy chủ. Bạn phảI tạo một cổng khác cho bộ đăng ký

Remote Method Invocation

BT1: Write a program depicting remote interface

```
package myrem.host;
import java.rmi.*;
public interface MyRem extends Remote
{
int gnum( ) throws RemoteException;
String gchar(int n) throws RemoteException;
}
```

BT2: Write a program that displays random number

```
package ju.exam.random.server;
import java.rmi.*;
public interface RandomServer extends Remote {
double getRandom() throws RemoteException;
}
package ju.exam.random.client;
import ju.exam.random.server.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.rmi.*;
import java.rmi.server.*;
public class RandomClient extends Frame implements Runnable {
String host="192.160.90.233";
TextField text = new TextField(50);
Button newRandom = new Button("Random");
RandomServer server;
int screenWidth = 500;
int screenHeight = 100;
public void run(){
}
public RandomClient() {
super("RandomClient");
setup();
setSize(screenWidth,screenHeight);
addWindowListener(new WindowEventHandler());
show();
}
void setup() {
setLayout(new FlowLayout(FlowLayout.LEFT));
newRandom.addActionListener(new ButtonHandler());
```

```
add(newRandom);
add(text);
connectToServer();
}
void connectToServer() {
try {
server = (RandomServer) Naming.lookup("//"+host+"/RandomServer");
} catch (Exception ex) {
text.setText(ex.toString());
}
}
class ButtonHandler implements ActionListener {
public void actionPerformed(ActionEvent ev){
String s=ev.getActionCommand();
if("Random".equals(s)){
try {
double rand=server.getRandom();
text.setText(new Double(rand).toString());
} catch (Exception ex){
text.setText(ex.toString());
}
}
}
}
}
package ju.exam.random.server;
import java.rmi.*;
import java.rmi.server.*;
public class RandomServerImpl extends UnicastRemoteObject
implements RandomServer {
public RandomServerImpl() throws RemoteException {
super();
}
public double getRandom() throws RemoteException {
return Math.random();
}
public static void main(String args[]){
System.setSecurityManager(new RMISecurityManager());
try {
RandomServerImpl instance = new RandomServerImpl();
Naming.rebind("///RandomServer", instance);
System.out.println("RandomServer is registered.");
} catch (Exception ex) {
System.out.println(ex.toString());
}
}
}
```

BT3: Write an applet that accesses the remote server. This applet is used as a local client.

```
package ju.ab.info.client;
```

```
import ju.ab.info.server.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.rmi.*;
public class eg1 extends Applet {
String text = "Click here to Update the server.";
TextArea textArea = new TextArea(15,50);
Button update = new Button("Update");
InfoServer server;
public void init() {
setLayout(new BorderLayout());
add("Center",textArea);
update.addActionListener(new ButtonHandler());
add("South",update);
try {
URL hostURL = getCodeBase();
String host = hostURL.getHost();
server = (InfoServer) Naming.lookup("//"+host+"/InfoServer");
textArea.setText(text);
} catch (Exception ex) {
textArea.setText(ex.toString());
}
}
class ButtonHandler implements ActionListener {
public void actionPerformed(ActionEvent ev){
String s=ev.getActionCommand();
if("Update".equals(s)){
try {
String newText=server.getInfo();
text=newText+"\n"+text;
textArea.setText(text);
} catch (Exception ex){
textArea.setText(ex.toString());
}
}
}
}
}
```

The file to display the above applet.

```
<HTML>
<HEAD>
<TITLE>The applet representing the client</TITLE>
</HEAD>
<BODY>
<APPLET CODEBASE="http://204.115.182.233/codebase/"
CODE=" ju.ab.info.client.eg1.class"
WIDTH=800 HEIGHT=400>
</APPLET>
</BODY>
```

```
</HTML>
```

BT4: Write an applet that helps in developing a client application. This client application enables to browse through a host's registry so as to see which remote objects it supports.

```
import java.awt.*;
import java.awt.event.*;
import java.rmi.registry.*;
public class clnteg extends Frame {
TextField host = new TextField(30);
TextArea objectNames = new TextArea(15,50);
Button browse = new Button("Browse host");
int screenWidth = 500;
int screenHeight = 400;
public static void main(String args[]){
Browser app = new clnteg();
}
public clnteg() {
super("Browser");
setup();
setSize(screenWidth,screenHeight);
addWindowListener(new WindowEventHandler());
show();
}
void setup() {
browse.addActionListener(new ButtonHandler());
Panel panel = new Panel();
panel.add(new Label("Host name: "));
panel.add(host);
panel.add(browse);
add("North",panel);
add("Center",objectNames);
}
class ButtonHandler implements ActionListener {
public void actionPerformed(ActionEvent ev){
String s=ev.getActionCommand();
if("Browse host".equals(s)){
try {
Registry registry = LocateRegistry.getRegistry(host.getText());
String objectList[] = registry.list();
String objects="";
for(int i=0;i<objectList.length;++i){
objects+=objectList[i]+"\n";
}
objectNames.setText(objects);
} catch (Exception ex){
objectNames.setText(ex.toString());
}
}
} } }
```

Ứng dụng RMI

B1: Tạo giao diện (Remote interface)

```
public interface interface_name extends Remote {
 //Khai báo những phương thức được gọi từ xa
 public datatype|void method_name([parameter list]) throws RemoteException;
 //...
}
```

B2: Tạo lớp cài đặt giao diện (Remote interface)

```
public class class_name extends UnicastRemoteObject implements interface_name {
 //Phương thức khởi tạo
 public class_name() throws RemoteException {

 }
 //Cài đặt xử lý cho tất cả các phương thức có trong giao diện
 public datatype|void method_name([parameter list]) throws RemoteException {
   //Viết xử lý cho phương thức
 }
 //...
}
```

B3: Viết xử lý phía client

```
//Tạo đối tượng Registry
Registry reg = LocateRegistry.getRegistry(servername, portnumber);

//Truy xuất remoteObject
interface_name remoteObject = (interface_name)reg.lookup(registername);

//Gọi phương thức thông qua remoteObject
remoteObject.method_name([parameter list])
//...
```

B4: Viết xử lý phía server

```
//Tạo đối tượng Registry
Registry reg = LocateRegistry.createRegistry(portnumber);

//Đăng ký Remote Object
class_name remoteObject = new class_name();
reg.bind(registername, remoteObject);
```

Chương trình mẫu: Xây dựng ứng dụng tính bình phương của một số

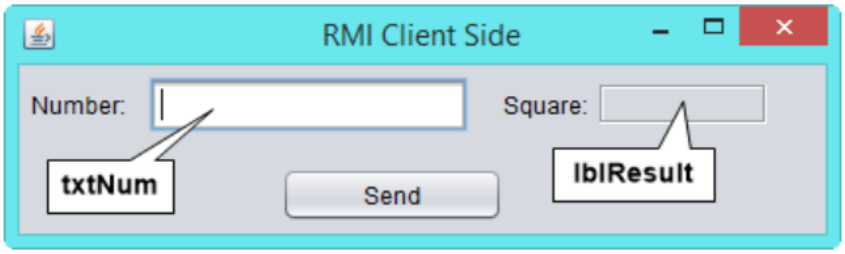
B1:

```
package swing_pkg.rmi;

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

/**
 *
 * @author giasutinhoc.vn
 */
public interface ICalculator extends Remote {

 // Khai báo phương thức tính bình phương
 public double square(double a) throws RemoteException;
}
```

B2:

```
package swing_pkg.rmi;

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

/**
 *
 * @author giasutinhoc.vn
 */
public class CalculatorImpl extends UnicastRemoteObject implements ICalculator{

 // Khai báo phương thức khởi tạo
 public CalculatorImpl() throws RemoteException{

 }

 // Viết xử lý cho phương thức tính bình phương
 public double square(double a) throws RemoteException {
   return a*a;
 }
}
```

B3:



Xử lý khi người dùng click nút Send

```
try {
 // Gọi server đang lắng nghe tại cổng 7777
 Registry reg = LocateRegistry.getRegistry("localhost", 7777);

 // Lấy đối tượng từ xa
 ICalculator cal = (ICalculator) reg.lookup("RMICalSer");

 // Gọi phương thức từ xa
 double result = cal.square(Double.parseDouble(txtNum.getText()));

 // Hiển thị kết quả
 lblResult.setText(String.valueOf(result));
} catch (Exception e) {
  e.printStackTrace();
}
```

B4:

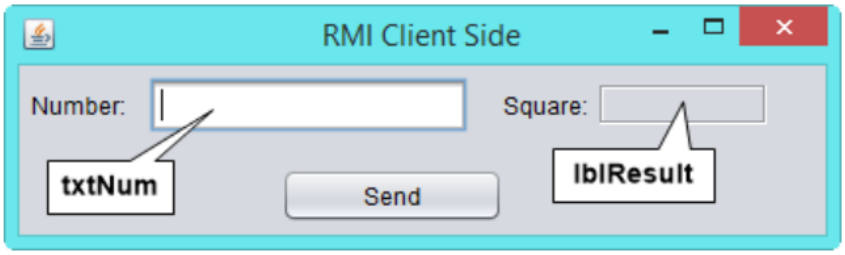
Xử lý khi người dung click nút Start

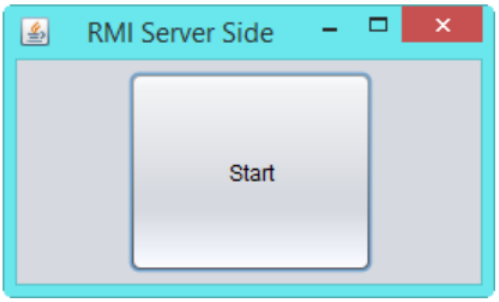
```
try {
 //Tạo đối tượng Registry
 Registry reg = LocateRegistry.createRegistry(7777);

 //Đăng ký Remote Object
 CalculatorImpl ci = new CalculatorImpl();
 reg.bind("RMICalSer", ci);
} catch (Exception e) {
  e.printStackTrace();
}
```

Bài tập thực hành

Câu 1: Xây dựng ứng dụng tính bình phương của một số

a. Thiết kế giao diện ứng dụng



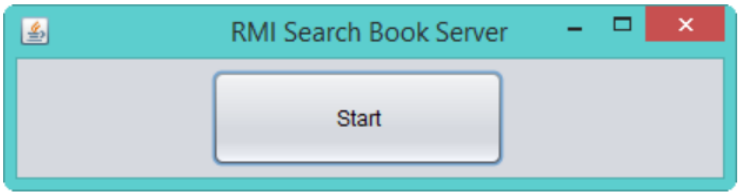


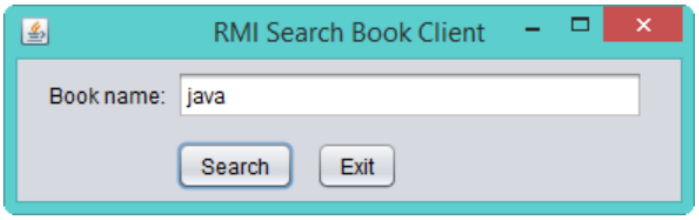
b. Viết xử lý

- Phía server: Nhận thông tin (số) mà client gởi về. Tính bình phương. Trả kết quả lại cho client
- Phía client: Gởi 1 số đến cho server. Hiển thị kết quả trả về từ server

Câu 2: Xây dựng ứng dụng tìm kiếm thông tin sách trong tập tin text

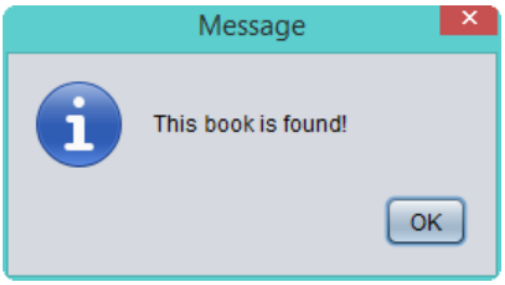
c. Thiết kế giao diện ứng dụng

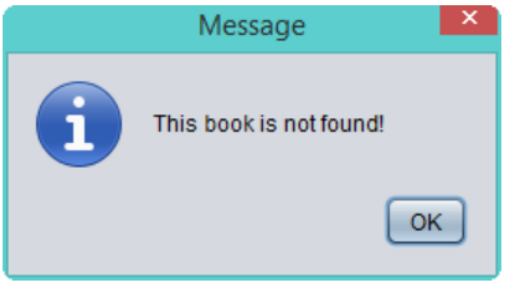




d. Viết xử lý

- Phía server: tìm thông tin sách mà client gửi về. Thông tin sách sẽ được tìm trong tập tin text. Nếu tìm thấy trả về kết quả là true, ngược lại là false.
- Phía client: sau khi người dùng nhập thông tin cần tìm và nhấn nút lệnh "**Search**", thực hiện gửi thông tin cần tìm cho server và hiển thị thông báo phù hợp dựa trên kết quả trả về từ server.





Câu 3: Write the Mail Server application

Câu 4: Write the Tictactoe Game that two person can play over network

Câu 5: Write the Applet that perform the tasks

a. Client sent to server string

b. Server rerutn to client Reverse String

Câu 6: Write the Program that performing Add, sub, Multi, division by RMI

Câu 7: Write the program that Matrix Add calculating by RMI

Câu 8: Write the Program that performs Reverse Number by clients

WSDL

Tài liệu tham khảo

https://congdongjava.com/forum/threads/t%E1%BB%95ng-h%E1%BB%A3p-java-l%E1%BA%ADp-tr%C3%ACnh-m%E1%BA%A1ng-ph%E1%BA%A7n-1.2681/

https://viblo.asia/p/java-rmi-va-ung-dung-phan-tan-don-gian-OEqGj50KM9bL

http://giasutinhoc.vn/lap-trinh-giao-dien/lap-trinh-phan-tan-voi-java-rmi-bai-7/

http://voer.edu.vn/c/phat-trien-he-thong-phan-tan/8fddcc3d

http://voer.edu.vn/c/lap-trinh-phan-tan-voi-web-service/8fddcc3d/79c2292d

https://congdongjava.com/forum/threads/t%E1%BB%95ng-h%E1%BB%A3p-java-l%E1%BA%ADp-tr%C3%ACnh-m%E1%BA%A1ng-ph%E1%BA%A7n-3.2694/