­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­



Лабораторна робота №3

з курсу «**Технології розподілених систем та паралельних обчислень**»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 11

Виконав студент гр. КНз-41

Чалий Михaйло

Перевірив

Фармага І.В.

­­

Львів 2016

## Мета роботи

Метою роботи є отримання навиків роботи з програмним інтерфейсом виклику віддалених методів у мові Java – Java RMI.

## Теоретична частина

RMI (англ. Remote Method Invocation) - програмний інтерфейс виклику віддалених методів у мові Java; це розподілена об'єктна модель, що описує, яким чином здійснюється виклик віддалених методів, що працюють на іншій віртуальній машині Java. При доступі до об'єктів на іншому комп'ютері, можна викликати методи цього об'єкту. Необхідно тільки доставити параметри методу на інший комп'ютер, повідомити об'єкт про необхідність виконання методу, а потім передати назад значення (якщо метод не void). Механізм RMI дає можливість організувати виконання всіх цих операцій.

RMI є реалізацією RPC на Java для розподілених комунікаційних взаємодій Java-об'єкт - Java-об'єкт. Об'єкт Java реєструється для віддаленого доступу, що дає можливість клієнтам отримувати віддалену посилання на цей об'єкт, що дозволяє використовувати цей об'єкт дистанційно. Синтаксис виклику методу ідентичний синтаксису виклику методів інших об'єктів у тій же програмі. RMI обслуговує маршалінг даних через мережу і дає можливість програмам на Java передавати закінчені об'єкти Java за допомогою механізму серіалізації об'єктів Java. В складі J2SE є інструментальні засоби створення необхідного коду для мережевих взаємодій з певних інтерфейсів програми, це означає, що RMI не вимагає від програміста знання мови IDL. Крім того, ніякого нейтрального до мови IDL інтерфейсу не потрібно - так як RMI підтримує тільки Java; достатньо власних інтерфейсів Java.

## Завдання

1. Скласти програму на мові Java, що реалізує RMI.
2. Виконати індивідуальне завдання (ваш варіант) до лабораторної роботи №3 «**Використання функціональної декомпозиції для розв’язку обчислювальних задач»,** використовуючи засоби RMI для розпаралелювання алгоритму на 2-х комп‘ютерах.
3. Проведіть дослідження коду та результатів виконання програми. Порівняйте результати розпаралелювання розв‘язку задачі на одному комп‘ютері із результатами паралельного виконання на 2-х комп‘ютерах.

Приклад результатів виконання п. 1. індивідуального завдання.

import java.rmi.\*;

import java.rmi.registry.\*;

import java.io.\*;

import java.rmi.server.\*;

public class test

{

public interface Brain extends Remote //Це віддалений інтерфейс

{

public void Ping() throws Exception;

public void message(String str) throws Exception; //Відправка текстового повідомлення на сервер

}

public class BrainImpl extends UnicastRemoteObject implements Brain

//Це реалізація віддаленого інтерфейсу

{

public BrainImpl() throws Exception

{

}

public void Ping() throws Exception

{

System.out.println("Pong! :)");

}

public void message(String str) throws Exception

{

System.out.println("Client: "+str);

}

}

public class Host //Це сервер

{

public Registry reg;

public Host(String HostName,int port) throws Exception

{

Registry reg = LocateRegistry.createRegistry(port); //Порт задеє користувач

reg.rebind(HostName, new BrainImpl()); //Прив'язую сервер до реєстру RMI

System.out.println("Host "+HostName+" started successfully");

System.out.println("Press [Enter] to close the Host");

BufferedReader keyboard = new BufferedReader(new InputStreamReader(System.in));

keyboard.readLine(); //Сервер завершить роботу при натиску клавіші Enter

}

}

public class Client //Це клієнт

{

public Brain remotingBrain; //Це інтерфейс що буде посилатися на віддалений клас

public Client(String address,String HostName,int port) throws Exception

{

remotingBrain = (Brain) Naming.lookup("//"+address+":"+port+"/"+HostName);

//З'єднуюся з реєстром RMI і отримую посилання на віддалений серверний об'єкт

System.out.println("Connection to "+address+":"+port+"/"+HostName+" is succeed");

}

}

public test() throws Exception //Це реалізація інтерфейсу користувача

{

System.out.println("Type [1] to start Host");

System.out.println("Type [2] to start Client");

BufferedReader keyboard = new BufferedReader(new InputStreamReader(System.in));

int c = keyboard.readLine().charAt(0); //Вибір запуску клієнта або сервера

if (c == '1') //Якщо обрано запуск сервера

{

String address = null;

String name = null;

int port;

System.out.print("\t-host name:\t");

name = keyboard.readLine(); //Зчитую ім'я сервера

System.out.print("\t-port:\t\t");

port = Integer.parseInt(keyboard.readLine()); //І номер порта де він буде розміщений

System.out.println("\tHost is starting ...");

Host myHost = new Host(name,port); //Запускаю сервер

}

else if (c == '2') //Якщо обрано запуск клієнта

{

String address = null;

String name = null;

int port;

System.out.print("\t-adress:\t");

address = keyboard.readLine(); //Зчитую адрес де має бути розташований сервер

System.out.print("\t-host name:\t");

name = keyboard.readLine(); //Зчитую його ім'я

System.out.print("\t-port:\t\t");

port = Integer.parseInt(keyboard.readLine()); //І його порт

System.out.println("\tClient is starting...");

Client myClient = new Client(address,name,port); //Запускаю клієнт

System.out.print("\tPing! - ");

myClient.remotingBrain.Ping(); //Викликаю просту віддалену процедуру

System.out.println("Pong!");

System.out.println("Type <something data> to send that");

System.out.println("Type [%exit] to exit");

String str = null;

while(!(str = keyboard.readLine()).equalsIgnoreCase("%exit"))

myClient.remotingBrain.message(str); //Зациклюю програму для можливості передачі

//повідомлень на серверний комп'ютер

}

}

public static void main(String[] args) throws Exception

{

new test();

}

}

Приклад виконання програми:

|  |  |  |
| --- | --- | --- |
| Сервер | Клієнт | Примітка |
| **Type [1] to start Host**  **Type [2] to start Client**  **1**  **-host name: lab3**  **-port: 1099**  **Host is starting ...**  **Host lab3 started successfully**  **Press [Enter] to close the Host**  **Pong! :)**  **Client: Hello world!** | **Type [1] to start Host**  **Type [2] to start Client**  **2**  **-adress: localhost**  **-host name: lab3**  **-port: 1099**  **Client is starting...**  **Connection to localhost:1099/lab3 is succeed**    **Ping! - Pong!**  **Type <something data> to send that**  **Type [%exit] to exit**  **Hello world!**  **%exit** | З'єднання успішне  Передача повідомлення  Клієнт завершив роботу |
|  |  |  |

## Висновки

Отримав навики роботи з програмним інтерфейсом виклику віддалених методів у мові Java – Java RMI.