|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №2**

по дисциплине «Архитектура клиент-серверных приложений»

**Тема практической работы: Концепция удаленного вызова метода (RMI) в JAVA**

**Студент группы** ИКБО-10-19 Харитонов Дмитрий Михайлович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** преподаватель Степанов П. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Москва 2021

**Цель:**

Знакомство с механизмами удаленного вызова процедур в JAVA.

**Задание:**

Использую информацию из описания данной практической работы, необходимо реализовать удалённый метод решения квадратных уравнений общего вида ax2 + bx + c = 0. При этом, условие уравнения передавать на сервер, а клиентская часть должна получать результат в виде объектов пользовательского класса. Клиент и сервер должны работать на одном хосте.

**Ход работы:**

В ходе выполнения практической работы были созданы необходимые классы. Ниже, на рисунках, представлена реализация программы и результат её работы.

Исходный код программы:

CalcServer.java

import java.rmi.RemoteException;  
  
public class CalcServer implements Calculator{  
 @Override  
 public RootList calculate(double a, double b, double c) throws RemoteException {  
 Double D;  
 D = b \* b - 4 \* a \* c;  
 if (D > 0) {  
 Double x1, x2;  
 x1 = (-b - Math.*sqrt*(D)) / (2 \* a);  
 x2 = (-b + Math.*sqrt*(D)) / (2 \* a);  
 return new RootList(x1,x2,true);  
 }  
 else if (D == 0) {  
 Double x;  
 x = -b / (2 \* a);  
 return new RootList(x,x,true);  
 }  
 else {  
 return new RootList(null,null,false);  
 }  
 }  
}

Calculator.java

import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
public interface Calculator extends Remote {  
 RootList calculate(double a, double b, double c) throws RemoteException;  
}

ClientMain.java

import java.rmi.NotBoundException;  
import java.rmi.RemoteException;  
import java.rmi.registry.LocateRegistry;  
import java.rmi.registry.Registry;  
  
public class ClientMain {  
  
 public static final String *UNIQUE\_BINDING\_NAME* = "server.calculator";  
  
 public static void main(String[] args) throws RemoteException, NotBoundException {  
  
 final Registry registry = LocateRegistry.*getRegistry*(2732);  
  
 Calculator calculator = (Calculator) registry.lookup(*UNIQUE\_BINDING\_NAME*);  
  
 RootList result = calculator.calculate(1,-8,15);  
  
 System.*out*.println(result);  
 }  
}

RootList.java

import java.io.Serializable;  
  
public class RootList implements Serializable {  
 Double x1;  
 Double x2;  
 boolean isHas;  
  
 public RootList(Double x1, Double x2, boolean isHas) {  
 this.x1 = x1;  
 this.x2 = x2;  
 this.isHas = isHas;  
 }  
  
 @Override  
 public String toString() {  
 if(isHas) {  
 if (x1 != x2){  
 return "x1= "+x1+" x2= "+x2;  
 }  
 else{  
 return "x= "+x1;  
 }  
 }  
 return "Уравнение не имеет действительных корней!";  
 }  
}

ServerMain.java

import java.rmi.AlreadyBoundException;  
import java.rmi.Remote;  
import java.rmi.RemoteException;  
import java.rmi.registry.LocateRegistry;  
import java.rmi.registry.Registry;  
import java.rmi.server.UnicastRemoteObject;  
  
public class ServerMain {  
  
 public static final String *UNIQUE\_BINDING\_NAME* = "server.calculator";  
  
 public static void main(String[] args) throws RemoteException, AlreadyBoundException, InterruptedException {  
  
 final CalcServer server = new CalcServer();  
  
 final Registry registry = LocateRegistry.*createRegistry*(2732);  
  
 Remote stub = UnicastRemoteObject.*exportObject*(server, 0);  
 registry.bind(*UNIQUE\_BINDING\_NAME*, stub);  
  
 Thread.*sleep*(Integer.*MAX\_VALUE*);  
  
 }  
}

**Вывод программы:**

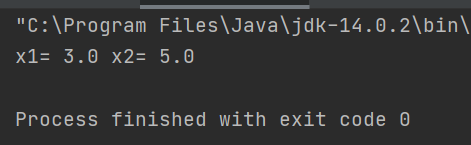


Рисунок 1 – результат выполнения программы

**Вывод:**

В результате выполнения практической работы были получены знания и навыки работы с механизмами удаленного вызова процедур в JAVA.

**GitHub:**

Github - https://github.com/Unripper1/Architecture