|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2** | |
| **по дисциплине** | |
| **«Архитектура клиент-серверных приложений»** | |
|  | |
| Выполнили студенты группы ИКБО-10-19 | Старовойтов Ф.Д. |
| Принял преподаватель | Степанов П.. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_201\_\_ г. | *(подпись руководителя)* |

Москва 2021

**Практическая работа №2 Концепция удаленного вызова метода (RMI) в JAVA**

**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.

**Решение:**

1) Реализация сервера

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

2) Реализация клиента

public class Client {  
 public static final String *UNIQUE\_BINDING\_NAME* = "server.equation";  
  
 public static void main(String[] args) throws RemoteException, NotBoundException {  
  
 final Registry registry = LocateRegistry.*getRegistry*(2732);  
  
 Equation equation = (Equation) registry.lookup(*UNIQUE\_BINDING\_NAME*);  
 String result = equation.solution(3, 4, 5);  
  
 System.*out*.println(result);  
 }  
}

3) Реализация интерфейса Equation

public interface Equation extends Remote {  
 String solution(double a, double b, double c) throws RemoteException;  
}

4) Реализация класса, который имплементирует Equation

public class EquationImpl implements Equation{  
  
 @Override  
 public String solution(double a, double b, double c) throws RemoteException {  
 double 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 "Корни уравнения: x1 = " + x1 + ", x2 = " + x2;  
 }  
 else if (D == 0) {  
 double x;  
 x = -b / (2 \* a);  
 return "Уравнение имеет единственный корень: x = " + x;  
 }  
 else {  
 return "Уравнение не имеет действительных корней!";  
 }  
 }  
}

При запуске сервера и клиента вместе – мы можем запускать удаленно методы объекта из клиентской части, но сама работа метода будет происходит на серверной части.

**Вывод:**

В результате практической работы мы научились работать с RMI в Java.