|  |
| --- |
|  |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |
| Институт информационных технологий (ИТ) |
| Кафедра … |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1** | |
| **по дисциплине** |  |
| **«Программирование на языке Java»**  **Тема: Многопоточные приложения, как основа распределённых клиент-серверных вычислений** | |
| Выполнил студент группы ИКБО-20-19 Анваржонов Ж Т |  |
| Принял ассистент кафедры ИиППО | Русляков А.А. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « 28 » 09 2021г. |  |
| «Зачтено» | « » 2021 г. |  |

Москва 2020

**Теоретическое введение**

Цель работы: Знакомство с механизмами удаленного вызова процедур в JAVA.

**Постановка задачи**

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.

**Программный код**

**Интерфейс Calculator**

*import* java.rmi.*Remote*;  
*import* java.rmi.RemoteException;  
*import* java.util.ArrayList;  
  
*public interface Calculator extends Remote* {  
 String quadraticEquation(*float* a, *float* b, *float* c) *throws* RemoteException;  
}

**Реализация интерфейса:**

*import* java.rmi.RemoteException;  
  
*/\*\*  
 \* ЭТО КЛАСС-СЕРВЕР, который будет реализовывать наш интерфейс Calculator  
 \*/  
  
public class* RemoteCalculationServer *implements Calculator* {  
  
 @Override  
 *public* String quadraticEquation(*float* a, *float* b, *float* c) *throws* RemoteException {  
  
 *float* discriminant = b \* b - 4 \* a \* c;  
  
 *if* (discriminant < 0)  
 *return* "При a = " + a + ", b= " + b + " и с= " + c + ": Уравнение не имеет действительных корней";  
 *else if* (discriminant == 0) {  
 *float* x = (*float*) (-b / (2 \* a));  
 *return* "При a = " + a + ", b= " + b + " и с= " + c + ": Уравнение имеет единственный корень: x = " + x;  
 } *else* {  
 *float* x1, x2;  
 x1 = (*float*) ((-b - Math.*sqrt*(discriminant)) / (2 \* a));  
 x2 = (*float*) ((-b + Math.*sqrt*(discriminant)) / (2 \* a));  
 *return* "При a = " + a + ", b= " + b + " и с= " + c + ": Корни уравнения: x1 = " + x1 + ", x2 = " + x2;  
 }  
 }  
}

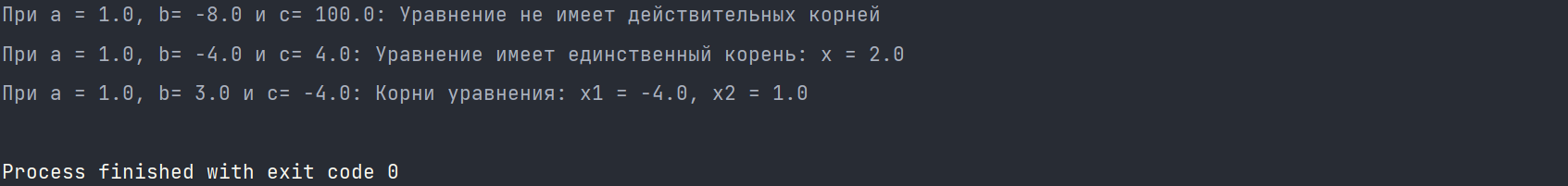
**Класс ServerMain:**

*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;  
  
*/\*\*  
 \* ЭТО ПРОГРАММА-СЕРВЕР, которая будет настраивать и запускать наш серверный класс-калькулятор.  
 \*/  
public class* ServerMain {  
 *public static final* String UNIQUE\_BINDING\_NAME = "server.calculator";  
  
 *public static void* main(String[] args) *throws* RemoteException, AlreadyBoundException, InterruptedException {  
 *final* RemoteCalculationServer server = *new* RemoteCalculationServer();  
  
 *// Cоздали реестр удаленных объектов  
 final Registry* registry = LocateRegistry.*createRegistry*(1234);  
  
 *// Создали заглушку stub, которая инкапсулирует внутри себя весь процесс удаленного вызова  
 Remote* stub = UnicastRemoteObject.*exportObject*(server, 0);  
 *// зарегистрировали нашу заглушку в реестре удаленных объектов* registry.bind(UNIQUE\_BINDING\_NAME, stub);  
  
 Thread.*sleep*(Integer.MAX\_VALUE);  
 }  
}

**Класс СlientMain:**

*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 {  
  
 *//создание реестра расшареных объектов  
 final Registry* registry = LocateRegistry.*getRegistry*(1234);  
  
  
 *//получаем объект (на самом деле это proxy-объект)  
 Calculator* calculator = (*Calculator*) registry.lookup(UNIQUE\_BINDING\_NAME);  
  
 *// Вызываем удаленные методы* String result = calculator.quadraticEquation(1, -8, 100);  
 String result1 = calculator.quadraticEquation(1, -4, 4);  
 String result2 = calculator.quadraticEquation(1, 3, -4);  
  
 System.out.println(result);  
 System.out.println(result1);  
 System.out.println(result2);  
  
 }  
}

**Вывод программы**

****

**Вывод**

В результате выполнения реализовали удалённый метод решения квадратных уравнений общего вида ax2 + bx + c = 0.