|  |
| --- |
|  |
| 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 инструментального и прикладного программного обеспечения |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2** | |
| **по дисциплине** |  |
| **«Архитектура клиент-серверных приложений»**  **Тема: Концепция удаленного вызова метода (RMI) в JAVA** | |
| Выполнил студент группы ИКБО-20-21 | Сидоров С.Д. |
| Принял преподаватель кафедры ИиППО | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | « » 2023 г. |  |
| «Зачтено» | «  » 2023 г. |  |

Москва 2023

**Теоретическое введение**

RMI - это объектно-ориентированный эквивалент RPC (удаленного вызова процедур). Это называется вызовом удаленного метода. Вызов удаленного метода (RMI) позволяет объекту Java вызывать метод объекта, запущенного на другом компьютере. RMI обеспечивает удаленную связь между Java-программой.

Главной целью разработчиков RMI было предоставление возможности программистам разрабатывать распределенные Java программы, используя такие же синтаксис и семантику, как и при разработке обычных нераспределенных программ. Для этого они должны были преобразовать модель работы классов и объектов в одной виртуальной машине Java (JVM) в новую модель работы классов и объектов в распределенной (несколько JVM) вычислительной среде.

Приложение RMI можно разделить на две части. Одним из них является программой клиентом и другая программа сервер. Сервер программа создает некоторые удаленный объект, сделать их ссылки доступны для клиента для вызова метода на нем. Клиент программа делает запрос для удаленных объектов на сервере и вызвать метод на них. Заглушка и скелет - два важных объекта, используемых для связи с удаленным объектом.

Таким образом, запустив открытый сервер RMI в системе, можно разрешить внешним субъектам взаимодействовать с ним и, возможно, выполнять методы на сервере RMI. Эти методы должны быть определены в реализации Сервера. Как только они вызываются клиентом, они будут выполняться на сервере, а возвращаемые значения будут возвращены клиенту. Еще одна интересная часть заключается в том, что собственный RMI (опять же, я НЕ говорю о JMXRMI) не поддерживает большую часть безопасности, кроме шифрования соединения с использованием SSL

**Постановка задачи**

**Цель:** 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**

В ходе выполнения данной практической работы были созданы файлы Client.java, Server.java, SquareRootSolver.java, SquareEquationSolverImpl.java содержимое которых представлено в листингах 1 - 4.

Листинг 1 - содержимое файла client.java

public class Client {

public static void main(String[] args) {

try {

Registry registry = LocateRegistry.getRegistry(2099);

SquareRootSolver solver = (SquareRootSolver) registry.lookup("Solver");

double[] a = solver.solveSquareEquation(1,3, 2);

System.out.println(a.length);

for (int i = 0; i<a.length; i++){

System.out.println(a[i]+" ");

}

} catch (Exception e) {

System.err.println("Client exeption" + e.toString());

e.printStackTrace();

}

}

}

Листинг 2 - содержимое файла server.java

public class Server extends SquareEquationSolverImpl {

public Server() throws RemoteException {

}

public static void main(String[] args) {

try {

SquareEquationSolverImpl im = new SquareEquationSolverImpl();

//SquareRootSolver solver = (SquareRootSolver) UnicastRemoteObject.exportObject(im, 0);

LocateRegistry.createRegistry(2099);

Registry registry = LocateRegistry.getRegistry(2099);

registry.bind("Solver", im);

System.out.println("Server ready");

} catch (Exception e) {

System.err.println("Server exeption"+e.toString());

}

}

}

Листинг 3 - содержимое файла SquareRootSolver.java

public interface SquareRootSolver extends Remote {

double[] solveSquareEquation(double a, double b, double c) throws RemoteException;

}

Листинг 4 - содержимое файла SquareEquationSolverImpl.java

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class SquareEquationSolverImpl extends UnicastRemoteObject implements SquareRootSolver {

private static final long serialVersionUID = 1L;

protected SquareEquationSolverImpl() throws RemoteException {

super();

}

@Override

public double[] solveSquareEquation(double a, double b, double c) throws RemoteException {

double discriminant = b \* b - 4 \* a \* c;

double[] roots;

if (discriminant > 0) {

double root1 = (-b + Math.sqrt(discriminant)) / (2 \* a);

double root2 = (-b - Math.sqrt(discriminant)) / (2 \* a);

roots = new double[]{root1, root2};

} else if (discriminant == 0) {

double root = -b / (2 \* a);

roots = new double[]{root};

} else {

roots = new double[0];

}

return roots;

}

}

**Вывод программы**

Результат работы программы Server представлен на рисунке 1.

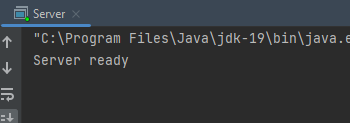
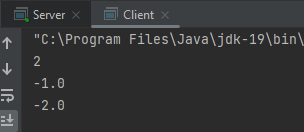


Рисунок 1 - Результат работы программы Server

Результат работы программы Client представлен на рисунке 2.

  
Рисунок 2 - Результат работы программы Client

**Вывод**

В результате выполнения данной работы был получен опыт работы с механизмами удалённого вызова процедур в JAVA.