|  |
| --- |
|  |
| 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-20-22 | Шумахер М.Е. |
| Принял преподаватель кафедры ИиППО | Волков М.Ю. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | «04» сентября 2024 г. |  |
| «Зачтено» | «04» сентября 2024 г. |  |

Москва 2024

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

**Теоретическое введение:**

**RMI** — это объектно-ориентированный эквивалент RPC (удаленного вызова процедур). Он позволяет объектам Java вызывать методы объектов на другом компьютере. Главная цель разработчиков RMI — предоставить возможность программистам разрабатывать распределенные Java-программы, используя синтаксис и семантику обычных программ.

**Архитектура RMI**

1. **Клиент-серверная модель**: Клиент и сервер обмениваются сообщениями через сеть.
2. **Интерфейсы**: Определение поведения и его реализация разделены.
3. **Заглушка и скелет**: Заглушка на стороне клиента и скелет на стороне сервера обрабатывают вызовы методов.

**Работа приложения RMI**

1. Клиент вызывает метод удаленного объекта.
2. Заглушка перехватывает вызов и передает его в RRL (Remote Reference Layer).
3. RRL на стороне сервера вызывает метод на стороне сервера.
4. Результат передается клиенту.

**Уровни архитектуры RMI**

1. **Уровень заглушки и скелета**: Интерпретирует вызовы методов и переадресовывает их.
2. **Уровень удаленной ссылки (RRL)**: Поддерживает семантику вызовов удаленных служб.
3. **Транспортный уровень**: Обеспечивает соединение между клиентскими и серверными JVM.

**Транспортный уровень**

* Соединения основаны на потоках TCP/IP.
* Даже если JVM работают на одном компьютере, они соединяются через TCP/IP.

**Основные шаги работы с RMI:**

* Определить удаленный интерфейс, согласованный с сервером.
* Написать код сервера.
* Запустить программу rmic (Java RMI stub compiler – компилятор заглушек RMI) для генерации связующего кода.
* Напишисать код клиента.
* Убедиться, что на сервере запущен RMI реестр (программа rmiregistry).
* Запустить сервер.
* Запустить один или несколько клиентов.

**Постановка задачи:** используя 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:**

import java.rmi.Naming;

public class Client {

    public static void main(String[] args) {

        try {

            IQuadraticSolver solver = (IQuadraticSolver) Naming.lookup("rmi://localhost/QuadraticSolver");

            double a = 2;

            double b = -9;

            double c = 4;

            Roots roots = solver.solve(a, b, c);

            if (roots.hasRealRoots()) {

                System.out.println("Корни уравнения:");

                System.out.println("x1 = " + roots.getRoot1());

                System.out.println("x2 = " + roots.getRoot2());

            } else {

                System.out.println("Уравнение не имеет действительных корней.");

            }

        } catch (Exception e) {

            System.err.println("Ошибка: " + e);

            e.printStackTrace();

        }

    }

}

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface IQuadraticSolver extends Remote {

    Roots solve(double a, double b, double c) throws RemoteException;

}

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class QuadraticSolver extends UnicastRemoteObject implements IQuadraticSolver {

    protected QuadraticSolver() throws RemoteException {

        super();

    }

    @Override

    public Roots solve(double a, double b, double c) throws RemoteException {

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

        boolean hasRealRoots = discriminant >= 0;

        double root1 = 0;

        double root2 = 0;

        if (hasRealRoots) {

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

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

        }

        return new Roots(root1, root2, hasRealRoots);

    }

}

import java.io.Serializable;

public class Roots implements Serializable {

    private final double root1;

    private final double root2;

    private final boolean hasRealRoots;

    public Roots(double root1, double root2, boolean hasRealRoots) {

        this.root1 = root1;

        this.root2 = root2;

        this.hasRealRoots = hasRealRoots;

    }

    public double getRoot1() {

        return root1;

    }

    public double getRoot2() {

        return root2;

    }

    public boolean hasRealRoots() {

        return hasRealRoots;

    }

}

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class Server {

    public static void main(String[] args) {

        try {

            LocateRegistry.createRegistry(1099);

            IQuadraticSolver solver = new QuadraticSolver();

            Naming.rebind("rmi://localhost/QuadraticSolver", solver);

            System.out.println("Сервер готов.");

        } catch (Exception e) {

            System.err.println("Ошибка на сервере: " + e);

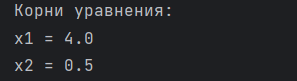
            e.printStackTrace();

        }

    }

}

Вывод программы:



**Вывод**: в результате выполнения практической работы выполнено знакомство с механизмами удаленного вызова процедур в JAVA.

**БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Многопоточность в Java / [Электронный ресурс] // Habr.com : [сайт]. — URL: https://habr.com/ru/articles/164487/ (дата обращения: 04.09.2024).
2. Wait and notify Methods in Java / [Электронный ресурс] // Baeldung.com : [сайт]. — URL: https://www.baeldung.com/java-wait-notify (дата обращения: 04.09.2024).
3. В чем разница между мьютексом, монитором и семафором / [Электронный ресурс] // Javarush.com : [сайт]. — URL: https://javarush.com/groups/posts/2174-v-chem-raznica-mezhdu-mjhjuteksom-monitorom-i-semaforom (дата обращения: 04.09.2024).