|  |
| --- |
|  |
| 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.

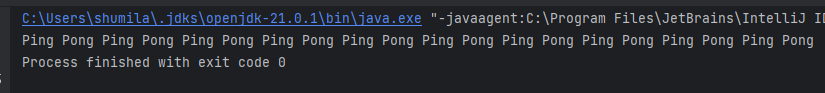
**Постановка задачи:** используя информацию из описания данной практической работы, необходимо реализовать удалённый метод решения квадратных уравнений общего вида ax2 + bx + c = 0. При этом, условие уравнения передавать на сервер, а клиентская часть должна получать результат в виде объектов пользовательского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

**Программный код:**

Main.java:

package prac1;  
  
import java.util.concurrent.locks.Condition;  
import java.util.concurrent.locks.ReentrantLock;  
public class Main {  
 static final ReentrantLock *lock* = new ReentrantLock();  
 static final Condition *pingCondition* = *lock*.newCondition();  
 static final Condition *pongCondition* = *lock*.newCondition();  
 static boolean *pingTurn* = true; // флаг  
 public static void main(String[] args) {  
  
 Thread pingThread = new Thread(() -> {  
 for (int i = 0; i < 10; i++) {  
 *lock*.lock();  
 try {  
 while (!*pingTurn*) {  
 *pingCondition*.await();  
 }  
 System.*out*.print("Ping ");  
 *pingTurn* = false;  
 *pongCondition*.signal();  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 } finally {  
 *lock*.unlock();  
 }  
 }  
 });  
  
 Thread pongThread = new Thread(() -> {  
 for (int i = 0; i < 10; i++) {  
 *lock*.lock();  
 try {  
 while (*pingTurn*) {  
 *pongCondition*.await();  
 }  
 System.*out*.print("Pong ");  
 *pingTurn* = true;  
 *pingCondition*.signal();  
 } catch (InterruptedException e) {  
 Thread.*currentThread*().interrupt();  
 } finally {  
 *lock*.unlock();  
 }  
 }  
 });  
  
 pingThread.start();  
 pongThread.start();  
 }  
}

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).