|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 №1** | |
| **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 №1 Многопоточные приложения, как основа распределённых клиент-серверных вычислений**

**Цель:**

Ознакомиться с моделью многопоточного программирования в Java.

**Задание:**

Используя материалы данной практической работы необходимо написать многопоточную программу, в которой два потока записывают строку в стандартный вывод, по образцу PING PONG PING PONG PING PONG. Программа должна работать следующим образом:

* 1-й поток печатает «Ping» и переходит в состояние ожидания.
* 2-й поток выходит из состояния ожидания, печатает «Pоng», уведомляет 1-й поток, возвращается в состояние ожидания.
* 1-й поток выходит из состояния ожидания, печатает «Pшng», уведомляет 2-й поток, возвращается в состояние ожидания.
* Шаги 2 и 3 повторяются и печатают «Ping Pong».

Программа должна быть реализована только с использованием Wait Notify, либо ReentrantLock.

**Решение:**

public class Main {  
  
 public static void main(String[] args) {  
 Mess mess = new Mess();  
  
 Thread thread1 = new Thread(mess);  
 thread1.setName("Ping");  
 thread1.start();  
  
 Thread thread2 = new Thread(mess);  
 thread2.setName("Pong");  
 thread2.start();  
 }  
  
 static class Mess implements Runnable {  
 @Override  
 public void run() {  
 synchronized(this) {  
 for(int i = 0; i < 10; i++) {  
 try {  
 notify();  
 System.*out*.println( Thread.*currentThread*().getName());  
 wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 }  
}

Мы создаем два потока с именами “Ping” и “Pong” – передаем им объект, имплементирующий метод Runnable. В методе run(), который определяет точку входа в поток определим блок synchronized, в котором будем поочередно выводить имена потоков. Управление очередностью осуществляется с помощью методов notify() и wait().

**Вывод:**

В результате практической работы мы научились управлять работой нескольких потоков.