|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №1**

по дисциплине «Архитектура клиент-серверных приложений»

**Тема практической работы: Многопоточные приложения, как основа распределённых клиент-серверных вычислений**

**Студент группы** ИКБО-10-19 Магин Константин Андреевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** преподаватель Степанов В. С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Москва 2021

**Цель:**

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

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

**Ход работы:**

В ходе выполнения практической работы был создан класс, реализующий работу с потоками, согласно заданию. Ниже, на рисунках, представлена реализация программы и результат её работы.

Исходный код программы:

PingPongThread.java

package com.company.pr1;  
  
public class PingPongThread implements Runnable {  
  
 private Object SyncObject;  
 private String name;  
  
 public PingPongThread(Object SyncObject, String name) {  
 this.SyncObject = SyncObject;  
 this.name = name;  
 }  
  
 @Override  
 public void run() {  
 synchronized (SyncObject) {  
 while(true) {  
 System.*out*.println(name);  
  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e1) {  
 e1.printStackTrace();  
 }  
 SyncObject.notify();  
  
 try {  
 SyncObject.wait(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

Main.java

package com.company.pr1;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Object LOCK\_OBJECT = new Object();  
 Thread ping = new Thread(new PingPongThread(LOCK\_OBJECT, "Ping"));  
 Thread pong = new Thread(new PingPongThread(LOCK\_OBJECT, "Pong"));  
 ping.start();  
 pong.start();  
 }  
}

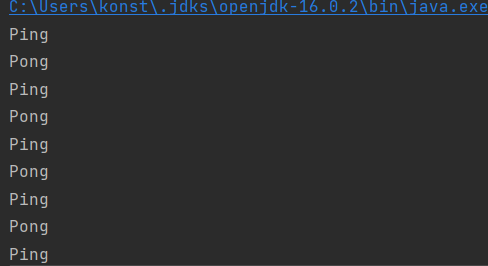


Рисунок 1 – результат выполнения программы

**Вывод:**

В результате выполнения практической работы были получены знания и навыки работы с многопоточной моделью программирования в JAVA.