|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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-01-19 Кузин Д.О.

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

(подпись студента)

**Руководитель практической работы**  Китанин С.С.

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

(подпись руководителя)

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

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

Москва 2021

# Задание на практическую работу

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 «Pong», 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.

# Ход работы

Содержимое файлов программы представлено на листингах 1-3.

Листинг 1. Содержимое файла ThreadPingPong.java

public class ThreadPingPong extends Thread{  
 static final Object *monitor* = new Object();  
 static volatile int *count* = 0;  
  
 public ThreadPingPong(String name) {  
 super(name);  
 }  
  
 @Override  
 public void run() {  
 synchronized (*monitor*){  
 while(*count*<10){  
 *monitor*.notify();  
 System.*out*.println(this.getName());  
 *count*++;  
 try {  
 *monitor*.wait();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

Листинг 2. Содержимое файла Main.java

public class Main {  
  
 public static void main (String[]args) throws InterruptedException {  
 ThreadPingPong ping = new ThreadPingPong("ping");  
 ThreadPingPong pong = new ThreadPingPong("pong");  
  
 ping.setDaemon(true);  
 pong.setDaemon(true);  
  
 ping.start();  
 pong.start();  
  
 Thread.*sleep*(2000);  
 }  
}

Результаты работы программы представлены на рисунке 1.

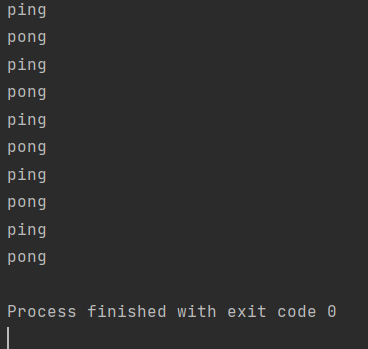


Рисунок – Результат работы программы

# Вывод

В результате работы была изучена модель многопоточного программирования в Java и была использована на практике на примере классов Thread и методов notify и wait.