**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное   
образовательное учреждение

Высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Направление: «Программная инженерия»

Отчет по лабораторной работе №2 по дисциплине

**«Параллельное, распределенное программирование»**

Выполнил:

Студент группы 8К61 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Исламов Е.Р.

Принял:

Доцент ОИТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мокина Е.Е.

Томск 2019

**Содержание**

[Цель работы 3](#_Toc23453489)

[Ход работы 4](#_Toc23453490)

[Вывод 6](#_Toc23453491)

# Цель работы

Познакомиться с приоритетами потоков в Java.

# Ход работы

В рамках работы создан класс NewThread, реализующий интерфейс Runnable. Для имитации работы потока используется класс ThreadWorkImitator с методом run, реализованный в рамках лабораторной работы №1.

**public class** NewThread **implements** Runnable {  
 **public** String **name**;  
 **public** Thread **t**;  
  
 **public** NewThread(String name) {  
 **this**.**name** = name;  
 **t** = **new** Thread(**this**);  
 System.***out***.println(**"Новый поток создан: "** + **t**);  
 **t**.start();  
 }  
  
 @Override  
 **public void** run() {  
 ThreadWorkImitator.*run*(**name**, 1000);  
 }  
}

Класс DemoJoin позволяет ввести количество потоков, а также приоритет двух случайных потоков в консоли. После этого происходит создание указанного количества потоков. Потоки производят обратный отсчет в консоль и завершаются.

**public class** DemoJoin {  
 */\*\*  
 \* Входная точка.  
 \*/* **public static void** main(String[] args) {  
 Scanner scanner = **new** Scanner(System.***in***);  
 System.***out***.print(**"Количество потоков: "**);  
 **int** threadCount = scanner.nextInt();  
  
 System.***out***.print(**"Приоритет: "**);  
 **int** priority = scanner.nextInt();  
  
 *// Создаем n потоков.* List<NewThread> threads = IntStream  
 .*range*(0, threadCount)  
 .mapToObj(x -> **new** NewThread(**"Thread-"** + x))  
 .collect(Collectors.*toList*());  
  
 *// Изменяем приоритет для двух потоков  
 // и выводим информацию о активности потока.  
 setPriority*(priority, 2, threads);  
 *printAliveInfo*(threads);  
  
 *// Ожидаем потоки.* **try** {  
 **for** (NewThread thread : threads) {  
 thread.**t**.join();  
 }  
 } **catch** (InterruptedException e) {  
 System.***out***.println(**"Главный поток прерван"**);  
 }  
  
 *// Выводим информацию о том, что потоки завершены.  
 printAliveInfo*(threads);  
 }  
  
 */\*\*  
 \* Задать нескольким случайным элементам списка (с учетом уникальности)  
 \* потоков указанный приоритет.  
 \*  
 \** ***@param priority*** *Приоритет.  
 \** ***@param count*** *Количество элементов.  
 \** ***@param threads*** *Список потоков.  
 \*/* **private static void** setPriority(**int** priority, **int** count, List<NewThread> threads) {  
 ThreadLocalRandom  
 .*current*()  
 .ints(0, threads.size())  
 .distinct()  
 .limit(count)  
 .forEach(x -> {  
 NewThread thread = threads.get(x);  
 thread.**t**.setPriority(priority);  
 System.***out***.println(String.*format*(**"Потоку %s задан приоритет %d."**,  
 thread.**name**, priority));  
 });  
 }  
  
 */\*\*  
 \* Вывести информацию о активности каждого потока.  
 \*  
 \** ***@param newThreads*** *Список потоков.  
 \*/* **private static void** printAliveInfo(List<NewThread> newThreads) {  
 **for** (NewThread t : newThreads) {  
 System.***out***.println(String.*format*(**"Поток %s запущен: %s"**,  
 t.**name**, t.**t**.isAlive()));  
 }  
 }  
}

На рисунке 1 приведен результат работы программы.

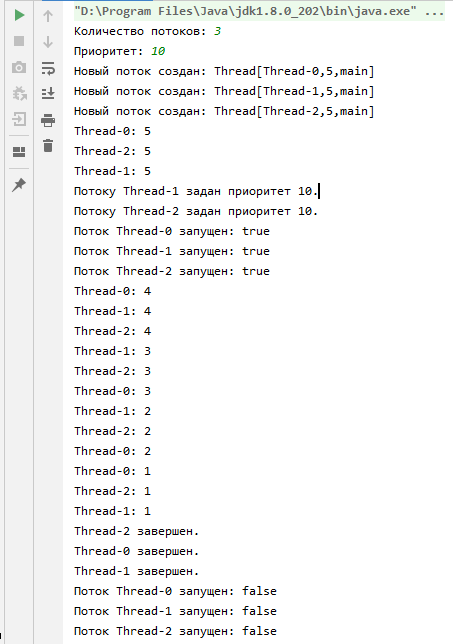


Рисунок - результат работы программы

# Вывод

В результате выполненной работы были улучшены навыки по работе с потоками Java, а также изучены приоритеты потоков.