**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 №1 по дисциплине

**«Параллельное, распределенное программирование»**

Выполнил:

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

Принял:

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

Томск 2019

**Содержание**

[Цель работы 3](#_Toc23452511)

[Задание 1 4](#_Toc23452512)

[Упражнение 1.2 5](#_Toc23452513)

[Упражнение 1.3 6](#_Toc23452514)

[Вывод 6](#_Toc23452515)

# Цель работы

Познакомиться с вариантами исполнения многопоточности в Java.

# Задание 1

В нижеприведённом коде производится изменение наименования главного потока исполнения и вывод цифр в обратном порядке в консоль из главного потока исполнения.

*/\*\*  
 \* Демонстрация управления главным потоком исполнения.  
 \*/***public class** CurrentThreadDemo {  
 **public static void** main(String[] args) {  
 Thread t = Thread.*currentThread*();  
 System.***out***.println(String.*format*(**"Имя текущего потока: %s"**, t.getName()));  
  
 t.setName(**"Название потока."**);  
 System.***out***.println(String.*format*(**"Имя потока после изменения: %s"**, t.getName()));  
  
 ThreadWorkImitator.*run*(**"Главный поток исполнения"**, 1000);  
 }  
}

*/\*\*  
 \* Имитация работы потока.  
 \*/***public class** ThreadWorkImitator {  
 */\*\*  
 \* Имитация работы потока.  
 \* Производит обратный отсчет с выводом в консоль  
 \* раз в {****@code*** *time} миллисекунд.  
 \** ***@param identifier*** *Наименование потока при выводе.  
 \** ***@param time*** *Задержка между итерациями отсчета.  
 \*/* **public static void** run(String identifier, **int** time) {  
 **try** {  
 **for** (**int** i = 5; i > 0; i--) {  
 System.***out***.println(identifier + **": "** + i);  
 Thread.*sleep*(time);  
 }  
 } **catch** (InterruptedException e) {  
 System.***out***.println(identifier + **" прерван."**);  
 }  
 System.***out***.println(identifier + **" завершен."**);  
 }  
}

На рисунке 1 приведен результат работы программы.

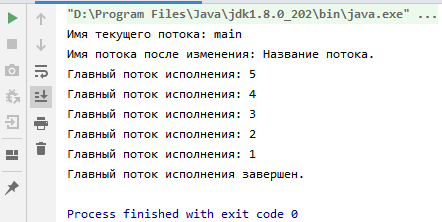


Рисунок 1 - результат работы программы

# Задание 2

В нижеприведённом коде производится создание и запуск нового потока, используя интерфейс Runnable:

*/\*\*  
 \* Поток исполнения, реализующий интерфейс Runnable.  
 \*/***public class** NewThreadRunnable **implements** Runnable {  
 Thread **t**;  
  
 **public** NewThreadRunnable() {  
 **t** = **new** Thread(**this**);  
 System.***out***.println(**"Дочерний поток создан: "** + **t**);  
 **t**.start();  
 }  
  
 @Override  
 **public void** run() {  
 ThreadWorkImitator.*run*(**"Дочерний поток"**, 500);  
 }  
}

**public class** ThreadDemo {  
 **public static void** main(String[] args) {  
 **new** NewThreadRunnable();  
 ThreadWorkImitator.*run*(**"Главный поток"**, 1000);  
 }  
}

На рисунке 2 приведен результат работы программы.

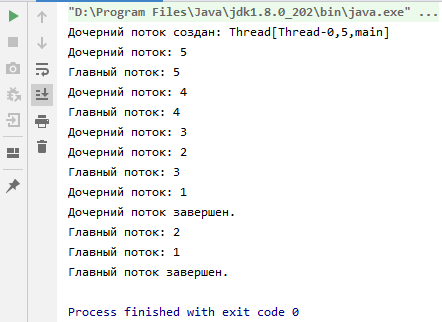


Рисунок 2 - результат работы программы

# Задание 3

В нижеприведённом коде производится создание и запуск нового потока, расширяющего класс Thread:

**public class** NewThreadExtends **extends** Thread {  
 **public** NewThreadExtends() {  
 **super**(**"Демонстрационный поток"**);  
 System.***out***.println(**"Дочерний поток создан: "** + **this**);  
 start();  
 }  
 **public void** run() {  
 ThreadWorkImitator.*run*(**"Дочерний поток: "**, 500);  
 }  
}

**public class** ExtendThreadDemo {  
 **public static void** main(String[] args) {  
 **new** NewThreadExtends();  
 ThreadWorkImitator.*run*(**"Главный поток"**, 1000);  
 }  
}

На рисунке 3 приведен результат работы программы.

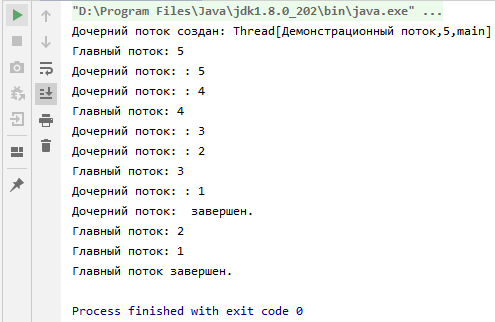


Рисунок 3 - результат работы программы

# Вывод

В результате выполненной работы были получены базовые навыки по реализации многопоточных программ с использованием языка Java.