**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 №3 по дисциплине

**«Параллельное, распределенное программирование»**

Выполнил:

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

Принял:

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

Томск 2019

**Содержание**

[Цель работы 3](#_Toc23454460)

[Ход работы 4](#_Toc23454461)

[Вывод 6](#_Toc23454462)

# Цель работы

Использование фабричных методов при использовании потоков в Java.

# Ход работы

В ходе работы был модифицирован класс из предыдущей лабораторный работы добавлением фабричного метода createAndStart, создающего и запускающего поток.

*/\*\*  
 \* Поток вывода информации из текстового файла.  
 \*/***public class** MyNewThread **implements** Runnable {  
 */\*\*  
 \* Наименование потока.  
 \*/* **private final** String **name**;  
 */\*\*  
 \* Поток.  
 \*/* Thread **thread**;  
 */\*\*  
 \* Флаг приостановки.  
 \*/* **private boolean suspendFlag**;  
  
 */\*\*  
 \* Конструктор.  
 \*  
 \** ***@param name*** *Наименование потока.  
 \*/* MyNewThread(String name) {  
 **this**.**name** = name;  
 **thread** = **new** Thread(**this**, name);  
 System.***out***.println(**"Новый поток: "** + **thread**);  
 **suspendFlag** = **false**;  
 }  
  
 */\*\*  
 \* Создание и запуск потока.  
 \*  
 \** ***@param name*** *Наименование потока.  
 \** ***@return*** *Поток.  
 \*/* **static** MyNewThread createAndStart(String name) {  
 MyNewThread thread = **new** MyNewThread(name);  
 thread.**thread**.start();  
 **return** thread;  
 }  
  
 */\*\*  
 \* Тело потока. Считывает информацию  
 \* из файла и выводит в консоль.  
 \*/* @Override  
 **public void** run() {  
 **try** {  
 File file = **new** File(**"src/Lab3/Demo/input.txt"**);  
 BufferedReader br = **new** BufferedReader(**new** FileReader(file));  
 String st;  
 **while** ((st = br.readLine()) != **null**) {  
 System.***out***.println(**name** + **": "** + st);  
 Thread.*sleep*(1000);  
 **synchronized** (**this**) {  
 **while** (**suspendFlag**) {  
 wait();  
 }  
 }  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* Приостановка выполнения потока.  
 \*/* **synchronized void** suspend() {  
 **suspendFlag** = **true**;  
 }  
  
 */\*\*  
 \* Возобновление выполнения потока.  
 \*/* **synchronized void** resume() {  
 **suspendFlag** = **false**;  
 notify();  
 }  
}

Также был создан класс StateInfoThread, позволяющий выводить информацию о состоянии какого-либо потока в консоль определенное число раз с указанной задержкой.

**package** Lab6;  
  
*/\*\*  
 \* Поток, выводящий информацию о состоянии  
 \* другого потока указанное число раз.  
 \*/***public class** StateInfoThread **implements** Runnable {  
 */\*\*  
 \* Поток.  
 \*/* **private final** Thread **thread**;  
 */\*\*  
 \* Поток, состояние которого нужно выводить.  
 \*/* **private** Thread **watchedThread**;  
 */\*\*  
 \* Количество раз вывода состояния.  
 \*/* **private int count**;  
  
 */\*\*  
 \* Задержка между выводами состояния.  
 \*/* **private int delay**;  
  
 */\*\*  
 \* Конструктор.  
 \*  
 \** ***@param watchedThread*** *Поток, состояние которого нужно выводить.  
 \** ***@param count*** *Сколько раз вывести состояние.  
 \** ***@param delay*** *Задержка между выводами.  
 \*/* StateInfoThread(Thread watchedThread, **int** count, **int** delay) {  
 **this**.**watchedThread** = watchedThread;  
 **this**.**count** = count;  
 **this**.**delay** = delay;  
 **thread** = **new** Thread(**this**);  
 }  
  
 */\*\*  
 \* Создание и запуск потока.  
 \*  
 \** ***@param thread*** *\** ***@param count*** *\** ***@param delay*** *\** ***@return*** *Поток.  
 \*/* **static** StateInfoThread createAndStart(Thread thread, **int** count, **int** delay) {  
 StateInfoThread t = **new** StateInfoThread(thread, count, delay);  
 t.**thread**.start();  
 **return** t;  
 }  
  
 */\*\*  
 \* Вывод состояния рассматриваемого потока.  
 \*/* @Override  
 **public void** run() {  
 **for** (; **count** > 0; **count**--) {  
 System.***out***.println(**"Состояние потока: "** + **watchedThread**.getState());  
 **try** {  
 Thread.*sleep*(**delay**);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

Запуск потоков происходит во входной точке Demo.

**public class** Demo {  
 **public static void** main(String[] args) **throws** Exception {  
 MyNewThread t1 = MyNewThread.*createAndStart*(**"Первый поток"**);  
 MyNewThread t2 = MyNewThread.*createAndStart*(**"Второй поток"**);  
 StateInfoThread.*createAndStart*(t1.**thread**, 12, 1000);  
 }  
}

На рисунке 1 приведен результат работы программы.

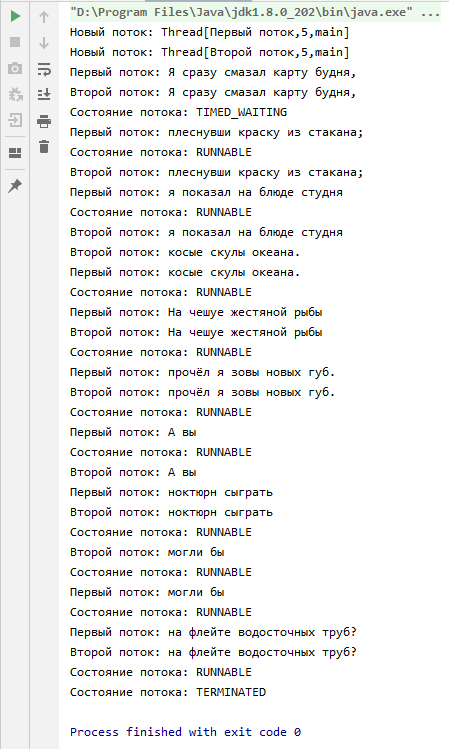


Рисунок 1 - результат работы программы

Как видно из рисунка, состояние потока меняется на протяжении времени. Состояние потока не приняло значение NEW, т.к. поток вывода состояния потока создавался после запуска первого потока.

# Вывод

В результате выполненной работы были улучшены навыки по работе с потоками Java, а также более подробно изучены их состояния. Был изучен фабричный метод.