**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 №4 по дисциплине

**«Параллельное, распределенное программирование»**

Выполнил:

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

Принял:

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

Томск 2019

**Содержание**

[Цель работы 3](#_Toc23456848)

[Ход работы 4](#_Toc23456849)

[Вывод 7](#_Toc23456850)

# Цель работы

Познакомиться с взаимодействием потоков исполнения в Java.

# Ход работы

В рамках работы создан класс очереди Queue, позволяющий добавлять и получать из очереди текстовую информацию посимвольно:

*/\*\*  
 \* Сихронизированная очередь.  
 \*/***public class** Queue {  
 */\*\*  
 \* Текущий элемент очереди  
 \*/* **private char n**;  
  
 */\*\*  
 \* Присвоено ли элементу следующее значение.  
 \*/* **private boolean valueSet** = **false**;  
  
 */\*\*  
 \* Получить следующий элемент очереди.  
 \*  
 \** ***@return*** *Значение элемента.  
 \*/* **synchronized char** get() {  
 **while** (!**valueSet**)  
 **try** {  
 wait();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 System.***out***.println(**"Получено: "** + **n**);  
 **valueSet** = **false**;  
 notify();  
 **return n**;  
 }  
  
 */\*\*  
 \* Добавляет элемент в очередь.  
 \*  
 \** ***@param n*** *Значение элемента.  
 \*/* **synchronized void** put(**char** n) {  
 **while** (**valueSet**)  
 **try** {  
 wait();  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 **this**.**n** = n;  
 **valueSet** = **true**;  
 System.***out***.println(**"Отправлено: "** + n);  
 notify();  
 }  
}

Класс Producer позволяет добавлять элементы в очередь (чтение происходит из файла):

*/\*\*  
 \* Источник данных очереди.  
 \*/***public class** Producer **implements** Runnable {  
 */\*\*  
 \* Очередь.  
 \*/* Queue **queue**;  
  
 */\*\*  
 \* Конструктор.  
 \*  
 \** ***@param queue*** *Очередь.  
 \*/* **public** Producer(Queue queue) {  
 **this**.**queue** = queue;  
 **new** Thread(**this**, **"Поставщик"**)  
 .start();  
 }  
  
 */\*\*  
 \* Чтение текстовой информации из файла и добавление в очередь побуквенно.  
 \*/* @Override  
 **public void** run() {  
 FileReader inputStream;  
 **try** {  
 inputStream = **new** FileReader(**"src/Lab3/Demo/input.txt"**);  
 **int** c;  
 **while** ((c = inputStream.read()) != -1)  
 **queue**.put((**char**) c);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

Класс Consumer позволяет получать элементы очереди:

*/\*\*  
 \* Потребитель очереди.  
 \*/***public class** Consumer **implements** Runnable {  
 */\*\*  
 \* Очередь.  
 \*/* **private** Queue **queue**;  
  
 */\*\*  
 \* Конструктор.  
 \*  
 \** ***@param queue*** *Очередь.  
 \*/* **public** Consumer(Queue queue) {  
 **this**.**queue** = queue;  
 **new** Thread(**this**, **"Потребитель"**)  
 .start();  
 }  
  
 */\*\*  
 \* Получение элементов очереди в вечном цикле.  
 \*/* @Override  
 **public void** run() {  
 **while** (**true**) {  
 **queue**.get();  
 }  
 }  
}

Исходный код входной точке представлен в листинге класса PC:

*/\*\*  
 \* Входная точка.  
 \*/***public class** PC {  
 **public static void** main(String args[]) {  
 Queue q = **new** Queue();  
 **new** Producer(q);  
 **new** Consumer(q);  
  
 System.***out***.println(**"Для остановки нажмите Ctrl + C"**);  
 }  
}

На рисунке 1 приведен результат работы программы.

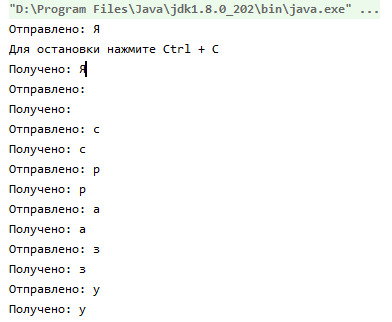


Рисунок 1 - результат работы программы

Как следует из рисунка, добавление элементов в очередь и получение происходит в правильном порядке. Часть вывода опущена, вследствие большого объема текста (используется стихотворение из лабораторной работы №3).

# Вывод

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