МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ХАРЬКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ»

Кафедра стратегического управления

Отчет

с лабораторной работы №5

**Применение потоков**  
Вариант №7

Выполнили:

Студенты группы КН-36а

Гордийчук, Антоненко, Кулик (В.В.)

Проверил:

Кондратов А. М.

Харьков, 2017

**Задание:** рассмотрим взаимодействие двух потоков, один из которых пишет данные в буферный пул, а другой считывает их из пула. Буферный пул состоит из N буферов, каждый содержит одну запись. В общем случае поток-писатель и поток-читатель имеют разные скорости и обращаются к пулу с переменной интенсивностью. Для правильной работы поток-писатель приостанавливается, когда все буферы заняты, и переходит в активное состояние при наличии хотя бы одного свободного буфера. Поток-читатель приостанавливается, когда все буферы пусты, и активизируется, когда появляется, по крайней мере, одна запись.

Описанный процесс происходит бесконечно.

Воспользоваться объектами синхронизации типа «мьютекс».

// Реализация поставщика и потребителя, использующая

// семафоры для управления синхронизацией

**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStreamReader;  
**import** java.util.LinkedList;  
**import** java.util.concurrent.Semaphore;  
  
**class** Mutex {  
  
 BufferedReader **console** = **new** BufferedReader(**new** InputStreamReader(System.***in***));  
 LinkedList<String> **buffer** = **new** LinkedList<>();  
  
 *// начать с недоступного семафора потребителя* **static** Semaphore *sem* = **new** Semaphore(1);  
  
 **void** get() {  
 **try** {  
 *sem*.acquire();  
 } **catch** (InterruptedException e) {  
 System.***out***.println(**"Перехвачено исключение типа InterruptedException"**);  
 }  
 System.***out***.printf(**"Вы ввели: "**);  
 **for** (String s : **buffer**) {  
 System.***out***.printf(s + **", "**);  
 }  
 **buffer**.clear();  
 *sem*.release();  
 }  
  
 **void** put() **throws** IOException {  
 **try** {  
 *sem*.acquire();  
 } **catch** (InterruptedException e) {  
 System.***out***.println(**"Перехвачено исключение типа InterruptedException"**);  
 }  
 **for** (**int** i = 0; i < 5; i++) {  
 **buffer**.addLast(**console**.readLine().toString());  
 }  
 System.***out***.println(**"Отправлено."**);  
 *sem*.release();  
 }  
}

**import** java.io.IOException;

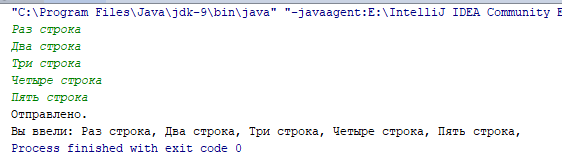
**class** Writer **implements** Runnable {  
 Mutex **mutex**;  
  
 Writer(Mutex mutex) {  
 **this**.**mutex** = mutex;  
 **new** Thread(**this**, **"Writer"**).start();  
 }  
  
 **public void** run() {  
 **try** {  
 **mutex**.put();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

**class** Reader **implements** Runnable {  
 Mutex **mutex**;  
  
 Reader(Mutex mutex) {  
 **this**.**mutex** = mutex;  
 **new** Thread(**this**, **"Reader"**).start();  
 }  
  
 **public void** run() {  
 **mutex**.get();  
 }  
}

**class** Main {  
 **public static void** main(String args[]) {  
 Mutex mutex = **new** Mutex();  
 **new** Writer(mutex);  
 **new** Reader(mutex);  
 }  
 }

}

Результат:



Вывод: в ходе лабораторной работы мы научились работать с потоками.