Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №6

по дисциплине «Платформонезависимое

программирование»

Тема: «Синхронизация потоков, работа с потоками ввода-вывода»

Выполнил:

студент гр. ПРИ-123

А.Ц. Нямаа

Приняла:

Ст. пр. кафедры ИСПИ

О.Н. Шамышева

Владимир, 2025 г.

ЦЕЛЬ РАБОТЫ:

Освоить приемы многопоточной синхронизации.

ЗАДАНИЕ (вариант 12).

Имеется два потока. Поток А считывает данные с внешнего устройства и записывает их в общий буфер. Поток Б считывает данные из буфера и записывает их в файл. Требуется организовать взаимодействие потоков так, чтобы они работали максимально эффективно, и при этом не происходила потеря данных.

ВЫПОЛНЕНИЕ РАБОТЫ:

В начале выполнения данного задания я определил физическую структуру приложения.

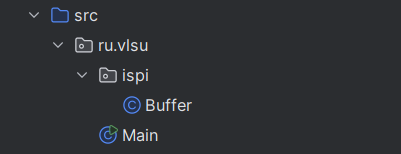


Рисунок 1. Физическая структура программы.

Далее написал следующие необходимые классы: Main, который является исполнительным, и Buffer, представляющий буфер.

Листинг Main.java:

package ru.vlsu;  
  
import java.io.FileWriter;  
import java.io.IOException;  
import java.util.Scanner;  
  
import ru.vlsu.ispi.Buffer;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.in);  
  
 int bufferLength = 20;  
 int pocketLength = 5;  
  
 Buffer buffer = new Buffer(bufferLength, pocketLength);  
  
 Thread reader = new Thread(() -> {  
 System.out.println("Введите текст: ");  
 try{  
 while (true){  
 String inputString = sc.nextLine();  
 for (int i = 0; i < inputString.length(); i++){  
 buffer.put(inputString.charAt(i));  
 }  
 buffer.put('\n');  
 }  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 });  
  
 Thread writer = new Thread(() -> {  
 try(FileWriter fileWriter = new FileWriter("Output.txt", true)){  
 while (true){  
 char[] stringToWrite = buffer.take();  
 fileWriter.write(new String(stringToWrite));  
 fileWriter.flush();  
 }  
 } catch (IOException | InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 });  
  
 reader.start();  
 writer.start();  
 }  
}

Листинг Buffer.java:

package ru.vlsu.ispi;  
  
public class Buffer {  
 private final char[] workingString;  
 private int putCurrentChar = 0;  
 private int readCurrentChar = 0;  
 private int count = 0;  
 private final int m;  
  
 public Buffer(int workingStringLength, int m){  
 this.workingString = new char[workingStringLength];  
 this.m = m;  
 }  
  
 public synchronized void put(char currentChar) throws InterruptedException {  
 while (count == workingString.length){  
 wait();  
 }  
 workingString[putCurrentChar] = currentChar;  
  
 if (putCurrentChar == workingString.length - 1){  
 putCurrentChar = 0;  
 }  
 else{  
 putCurrentChar++;  
 }  
  
 count++;  
 notifyAll();  
 }  
  
 public synchronized char[] take() throws InterruptedException{  
 while (count < m) {  
 wait();  
 }  
  
 char[] charsToReturn = new char[m];  
 for (int i = 0; i < m; i++){  
 charsToReturn[i] = workingString[readCurrentChar];  
  
 if (readCurrentChar == workingString.length - 1){  
 readCurrentChar = 0;  
 }  
 else{  
 readCurrentChar++;  
 }  
  
 count--;  
 }  
 notifyAll();  
 return charsToReturn;  
 }  
}

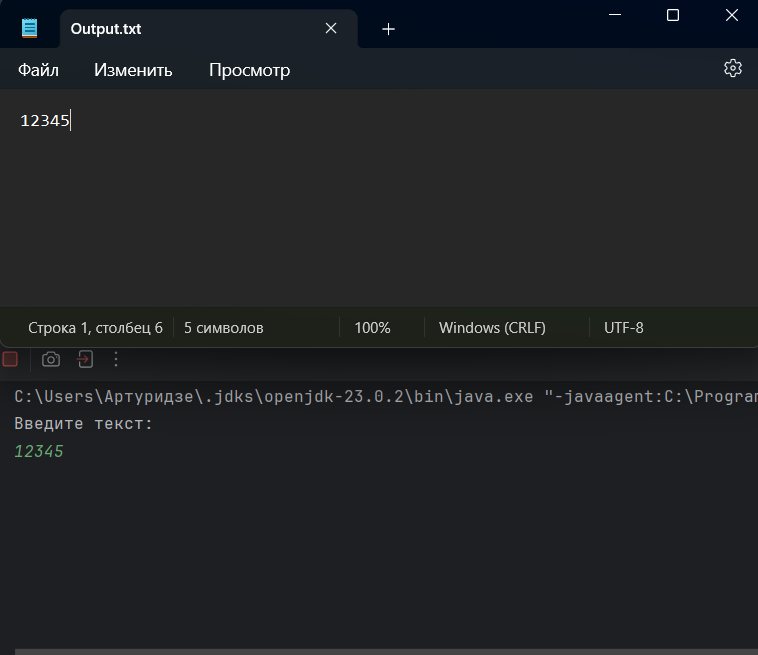
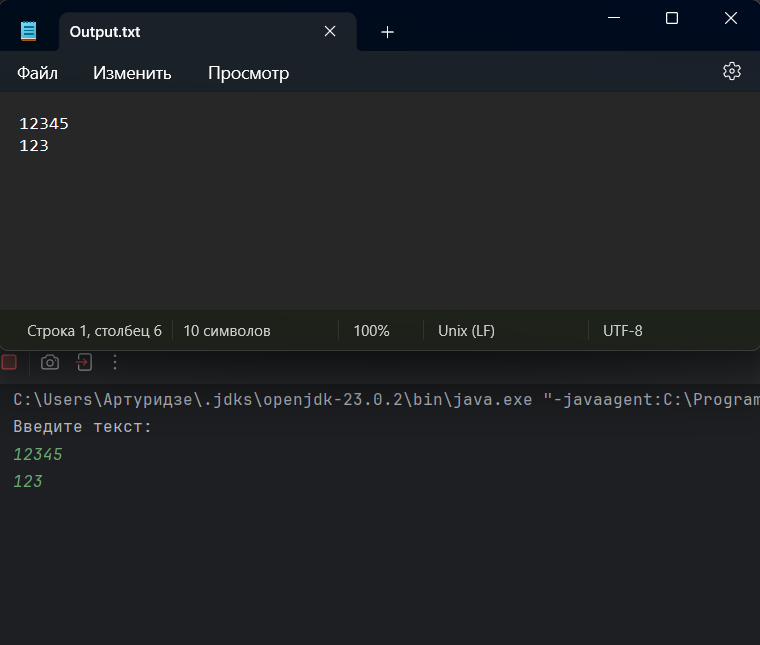


Рисунок 2. Тестирование созданной программы. Часть 1.

  
Рисунок 3. Тестирование созданной программы. Часть 2.

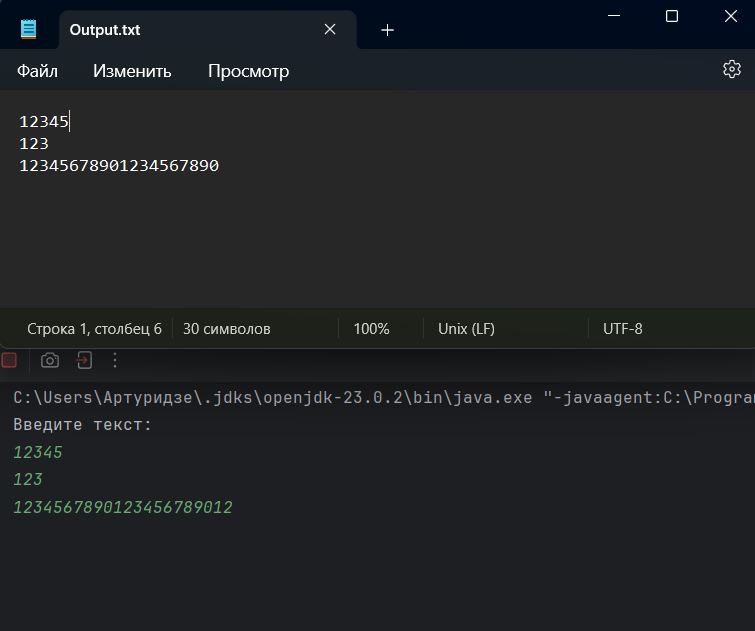


Рисунок 4. Тестирование созданной программы. Часть 3.

ВЫВОД К РАБОТЕ:

Я освоил приемы многопоточной синхронизации.