МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Сирота Марина Романовна

Институт информационных технологий и управления в технических системах

курс 3 группа ИС/б-32-о

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №2

по дисциплине: «Платформа Java»

по теме: «Основы объектно-ориентированного программирования Java»

Отметка о зачете \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

(дата)

Руководитель практикума

ст. пр. Кузнецов С.А.

(должность) (подпись) (инициалы, фамилия)

Севастополь

2018

1. ЦЕЛЬ РАБОТЫ

В ходе выполнения данной лабораторной работы необходимо ознакомиться с особенностями объектно-ориентированного программирования (ООП) на языке Java, приобрести практические навыки программирования на языке Java с использованием основных принципов ООП.

1. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

2.1. Описать абстрактный класс CBuffer, содержащий следующие поля:

* идентификатор буфера (**int** bufID) – уникальный идентификатор буфера;
* размер буфера (**int** bufSize) – максимальный размер буфера;
* количество созданных буферов **(int** BufCount).

Доступ к полям класса CBuffer должны иметь только методы этого класса и методы его потомков. Для организации доступа к этим полям из других классов необходимо реализовать общедоступные методы:

* **int** GetBufCount();
* **int** GetBufID().

Реализовать конструктор класса CBuffer (**int** count), выполняющий инициализацию идентификатора буфера (в качестве идентификатора использовать номер по порядку создаваемого буфера), размера буфера (значением count, передаваемым конструктору), увеличение количества созданных буферов.

В классе CBuffer описать абстрактный метод Generate().

2.2. В соответствии с вариантом задания реализовать дочерний класс для создания буфера, хранящего значения заданного типа T(см. таблицу 4.1). Для хранения значений реализовать поле – массив значений типа Т. В конструкторе класса использовать вызов конструктора родительского класса CBuffer, и кроме того создать массив значений типа Т(с использованием оператора **new)** и проинициализировать его с импользованием метода Generate().

Pеализовать метод Generate(), заполняющий массив случайными числами.

Для генерации случайных чисел необходимо, используя оператор import, подключить пакет java.util.Random. Для использования генератора случайный чисел сначала необходимо создать экземпляр класса Random:

*Randomrandom = newRandom();*

Генерация случайных чисел выполняется методами экземпляра класса Random. Например, для генерации случайного целого числа:

*random.nextInt();*

2.3. Описать интерфейсы:

1) IBufferPrintable – описывающий методы вывода на экран:

* **publicvoid**PrintInfo() – выводит на экран идентификатор, тип и размер буфера.
* **publicvoid**Print() – выводит на экран содержимое буфера.
* **publicvoid**PrintFirstN(**int**n) – выводит на экран первые n элементов буфера.
* **publicvoid**PrintLastN(**int**n) – выводит на экран последние n элементов буфера.

2)IBufferSortable – описывает метод для сортировки массива:

* **publicvoid**Sort();

3) IBufferComputable – описывает методы для вычисления статистики значений буфера.

* **publicvoid**Max() – вычисляет максимальный элемент буфера;
* **publicvoid**Min() – вычисляет минимальный элемент буфера;
* **publicvoid**Sum() – вычисляет сумму элементов буфера;

4) IBufferStorable – описывает методы для выгрузки буфера в текстовый файл.

* **publicvoid**SaveOneLine(Stringfilename) – сохраняет буфер в файл в одну строку;
* **publicvoid**SaveSeparateLines (Stringfilename) – сохраняет буфер в файл по одному элементу в строке;

2.4. Создать произвольный класс, унаследованный от класса, разработанного при выполнении п. 2.2, и реализующий методы интерфейсов из п. 2.3, необходимых для выполнения задания в соответствии с вариантом (см. таблицу 4.1).

2.5. Реализовать класс Lab2Java, в методе main которого в соответствии с вариантом задания (см. таблицу 4.1) реализовать работу с объектами класса из п. 2.4 с использованием их методов:

* Создать N буферов заданного типа T и размера L;
* Вывести на экран информацию o буферах;
* Вывести на экран первые 10 элементов буферов;
* Вычислить функцию F для каждого буфера и вывести результат на экран;
* Выполнить сортировку буферов методом S;
* Вывести на экран первые 10 элементов буферов;
* Сохранить буферы в файл с использованием метода O.

При написании программы допускается расширение классов необходимыми полями и методами.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | Long | 60 | Выборки | Max | SaveSeparateLines |

Вариант № 2

1. ВЫПОЛНЕНИЕ РАБОТЫ
   1. Листинг

**package** lab2;

**public** **class** Lab2Java {

**public** **static** **void** main(String[] args) {

**int** n = 4;

**int** l = 60;

CLongBuffer T[] = **new** CLongBuffer[n];

**for** (**int** i = 0; i < n; i++) {

T[i] = **new** CLongBuffer(l);

T[i].PrintInfo();

T[i].PrintFirstN(10);

T[i].Max();

T[i].Sort();

T[i].PrintFirstN(10);

T[i].SaveSeparateLine("buf" + i + ".txt");

}

}

}

**package** lab2;

**public** **class** Lab2Java {

**public** **static** **void** main(String[] args) {

**int** n = 4;

**int** l = 60;

CLongBuffer T[] = **new** CLongBuffer[n];

**for** (**int** i = 0; i < n; i++) {

T[i] = **new** CLongBuffer(l);

T[i].PrintInfo();

T[i].PrintFirstN(10);

T[i].Max();

T[i].Sort();

T[i].PrintFirstN(10);

T[i].SaveSeparateLine("buf" + i + ".txt");

}

}

}

package lab2;

import java.util.Random;

import java.io.\*;

public class CLongBuffer extends CBuffer

implements IBufferComputable, IBufferSortable, IBufferStorable, IBufferPrintable {

protected long T[];

public CLongBuffer(int count) {

super(count);

T = new long[count];

generate();

}

@Override

public void generate() {

Random rand = new Random();

int buffSize = getBufSize();

for (int i = 0; i < buffSize; i++) {

T[i] = rand.nextLong();

}

}

@Override

public void Max() {

long sum = T[0];

for (int i = 1; i < bufSize; i++) {

if(T[i]>sum)

sum=T[i];

}

System.out.print("Max: " + sum + '\n');

}

@Override

public void Sort() {

int m;

long tmp;

for(int i=0;i<bufSize-1;i++) {

m = i;

for(int j=i+1;j<bufSize;j++) {

if(T[j] < T[m]) {

m = j;

}

}

if(m != i) {

tmp = T[i];

T[i] = T[m];

T[m] = tmp;

}

}

}

@Override

public void PrintInfo() {

System.out.print('\t' + "Номер буфера: " + bufID + " Размер: " + bufSize + '\n');

}

@Override

public void Print() {

for (int i = 0; i < bufSize; i++) {

System.out.printf("%f ; ", T[i]);

}

System.out.print('\n');

}

@Override

public void PrintFirstN(int n) {

for (int i = 0; i < Math.min(bufSize, n); i++) {

System.out.printf("%d ; ", T[i]);

}

System.out.print('\n');

}

@Override

public void PrintLastN(int n) {

for (int i = Math.max(0, bufSize - n); i < bufSize; i++) {

System.out.printf("%d ; ", T[i]);

}

System.out.print('\n');

}

@Override

public void SaveSeparateLine(String filename) {

PrintFile(filename, '\n');

}

private void PrintFile(String filename, char separator) {

try (PrintWriter out = new PrintWriter(new FileOutputStream(filename, true))) {

for (int i = 0; i < bufSize; i++) {

out.print(T[i]);

out.print(separator);

}

out.print('\n');

out.close();

} catch (FileNotFoundException e) {

System.out.print("Не найден файл: " + filename);

e.printStackTrace();

}

}

}

**package** lab2;

**public** **interface** IBufferComputable {

**public** **void** Max();

}

**package** lab2;

**public** **interface** IBufferPrintable {

**public** **void** PrintInfo();

**public** **void** Print();

**public** **void** PrintFirstN(**int** n);

**public** **void** PrintLastN(**int** n);

}

**package** lab2;

**public** **interface** IBufferSortable {

**public** **void** Sort();

}

**package** lab2;

**public** **interface** IBufferStorable {

**public** **void** SaveSeparateLine(String fileName);

}

* 1. Результаты выполнения

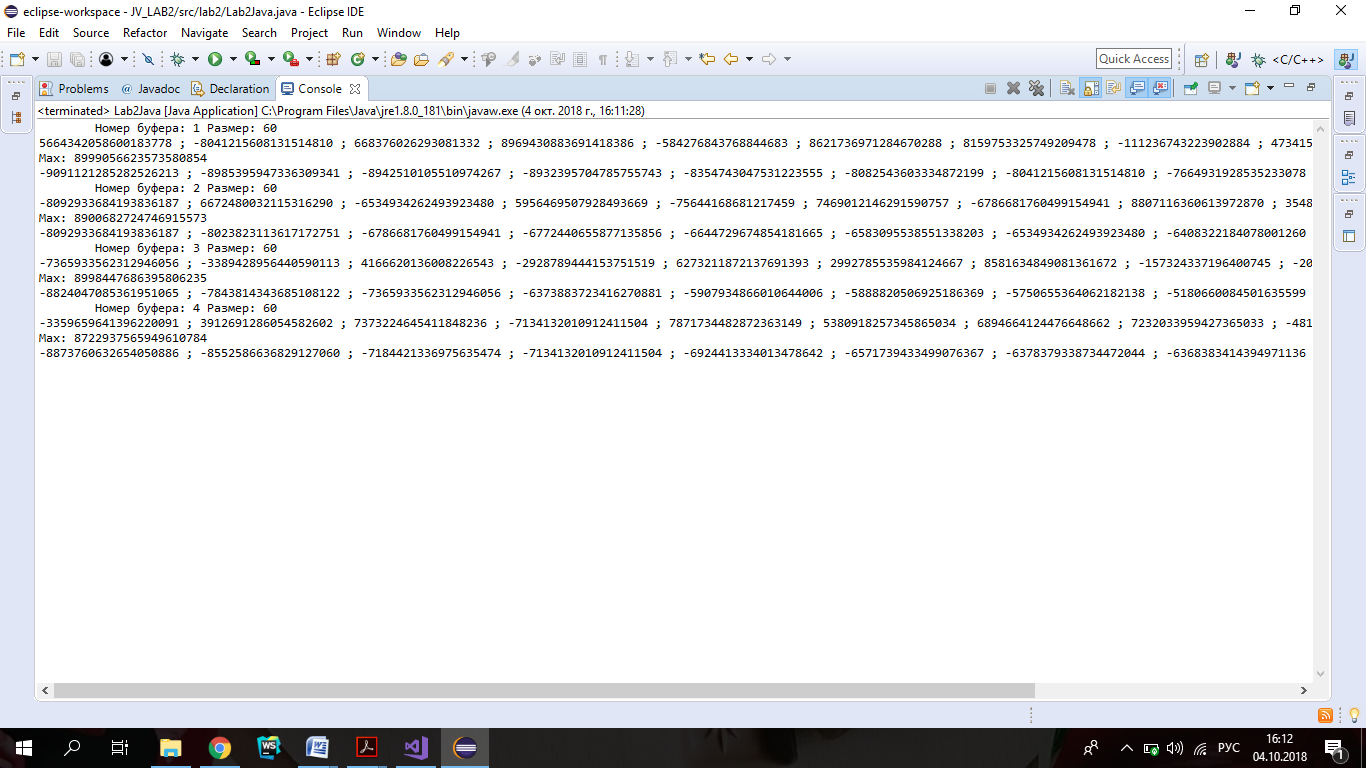


Рисунок 3.1 – Результаты в консоли

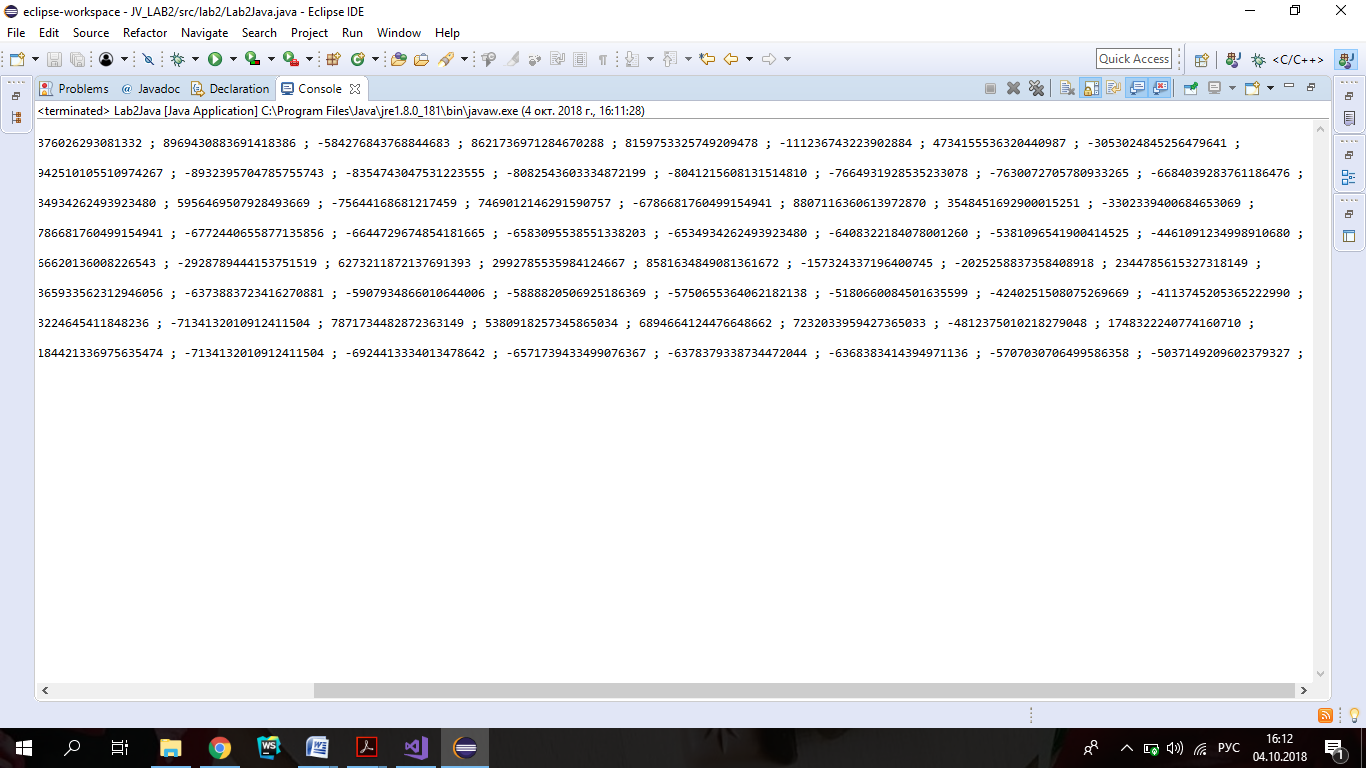


Рисунок 3.2 – Результаты в консоли

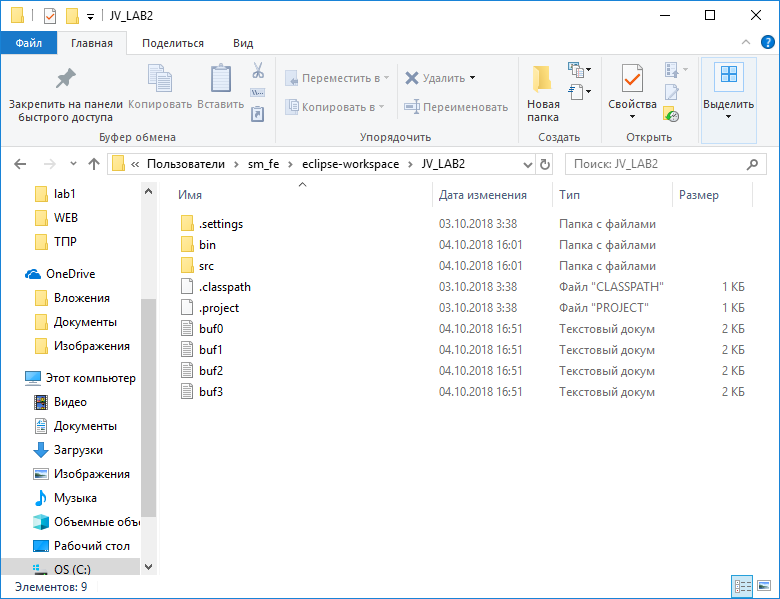


Рисунок 3.3 – Результаты в файлах

ВЫВОДЫ

В ходе данной лабораторной работы были изучены особенности объектно-ориентированного программирования (ООП) на языке Java.

Были приобретены практические навыки программирования на языке Java с использованием основных принципов ООП.

Была написана программа в соответствии с вариантом задания.