Лабораторная работа №2

«ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ JAVA»

2.1 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.

2.2 Постановка задачи

Требуется 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().

Реализовать дочерний класс для создания буфера, хранящего значения типа long. Для хранения значений реализовать поле – массив значений типа long. В конструкторе класса использовать вызов конструктора родительского класса CBuffer,

и кроме того создать массив значений типа long с использованием оператора new и проинициализировать его с импользованием метода Generate().

Реализовать метод Generate(), заполняющий массив случайными числами.

Требуется описать 4 интерфейса. Первый описывает методы вывода на экран: вывод на экран идентификатора, типа и размера буфера; вывод на экран содержимого буфера; вывод на экран первых n элементов буфера; вывод на экран последних n элементов буфера. Второй интерфейс описывает метод для сортировки массива. Третий описывает методы для вычисления статистики значений буфера: вычисляет максимальный/минимальный элемент буфера; вычисляет сумму элементов буфера. Четвёртый интерфейс описывает методы для выгрузки буфера в текстовый файл: сохранение буфера в файл в одну строку; сохранение буфера в файл по одному элементу в строке.

Создать произвольный класс, унаследованный от предыдущего и реализующий методы четырёх интерфейсов.

Реализовать в методе main работу с объектами произвольного класса с использованием их методов, в соответствии с вариантом задания:

- Создать N буферов заданного типа T и размера L;
- Вывести на экран информацию о 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.

Таблица 1 – Задание по варианту (Вариант 11)

Количество буферов (N)	Типы элементов буфера (T)	Число элементов в буферах (L)	Сортировка (S)	Вычисление (F)	Сохранение (O)
3	long	90	пузырька	max	saveOneLine

2.3 Ход работы

2.3.1 Текст программы

Была разработана программа на языке Java и представлена в листингах 1-8.

Листинг 1 – Класс Main

```
public class Main {

	public static void main(String[] args) {

		//Создать "3" буфера заданного типа "long" и размера "90"
		ArbitraryClass buf1 = new ArbitraryClass(90);
		ArbitraryClass buf2 = new ArbitraryClass(90);
		ArbitraryClass buf3 = new ArbitraryClass(90);

		//Вывести на экран информацию о буферах
		buf1.PrintInfo();
		buf2.PrintInfo();
		buf3.PrintInfo();

		//Вывести на экран первые 10 элементов буферов
		buf1.PrintFirstN(10);
		buf2.PrintFirstN(10);
		buf3.PrintFirstN(10);

		//вычислить функцию "max" для каждого буфера
		buf1.Max();
		buf2.Max();
		buf3.Max();

		//Выполнить сортировку буферов методом "Пузырька"
		buf1.Sort();
		buf2.Sort();
		buf3.Sort();

		//Вывести на экран первые 10 элементов буферов
		buf1.PrintFirstN(10);
		buf2.PrintFirstN(10);
		buf3.PrintFirstN(10);
```

```
            //Сохранить буферы в файл с использованием метода "сохранить в одну строку"
            buf1.SaveOneLine("buffer1.txt");
            buf2.SaveOneLine("buffer2.txt");
            buf3.SaveOneLine("buffer3.txt");
      }
}
```

Листинг 2 – Класс CBuffer

```
//абстрактный класс CBuffer
abstract class CBuffer {

//protected - Доступ к полям класса имеют только методы этого класса и методы его потомков (на самом деле будут ещё и иметь методы классов в данном пакете)
      protected int bufID;        //уникальный идентификатор буфера
      protected int bufSize;      //макисмальный разер буфера
      protected static int bufCount = 0; //кол-во созданных буферов (изначально 0) (общая переменная для всех объектов класса т.к. static)

      //конструктор выполняющий инициализацию всех полей
      CBuffer(int bufSize) {
            this.bufSize = bufSize;
            bufCount++;
            bufID = bufCount;
      }

//Для организации доступа к нашим полям из других классов реализованы следующие методы
      public int getBufCount() {
            return bufCount;
      }

      public int getBudID() {
            return bufID;
      }

      //абстрактный метод Generate()
      abstract void Generate();
}
```

Листинг 3 – Класс CreatingBuffer

```
//импорт класса рандом для создания случайных чисел
import java.util.Random;
```

```
//дочерний клас для создания буфера, хранящего значения типа long
public class CreatingBuffer extends CBuffer {

	protected long [] buffer; //массив значений

	//конструктор класса CreatingBuffer
	CreatingBuffer(int bufSize) {
		super(bufSize);
		buffer = new long[bufSize];
		Generate();
	}

	@Override
	protected void Generate() {
		//использование конструктора Random() для создания генератора
		Random rand = new Random();
		//заполнение массива случайными числами типа long
		for (int i = 0; i < buffer.length; i++) {
			buffer[i] = rand.nextLong();
		}
	}
}
```

Листинг 4 – Класс произвольный (основной класс в программе)

```
import Interfaces.*;
import java.io.*;

//произвольный класс, унаследованный от класса CreatingBuffer
//реализующий методы интерфейсов необходимых для выполнения задания в соответствии с вариантом

public class ArbitraryClass extends CreatingBuffer implements IBufferComputable, IBufferPrintable, IBufferSortable, IBufferStorable {

	//конструктор с параметром
	ArbitraryClass(int bufSize) {
		super(bufSize);
	}

	//выводит на экран идентификатор, тип и размер буфера
```

```
@Override
public void PrintInfo() {
	System.out.println("Идентификатор буфера: " + getBudID());
	System.out.println("Тип буфера: " + "long");
	System.out.println("Размер буфера: " + bufSize);
}

//выводит на экран содержимое буфера
@Override
public void Print() {
	for (int i = 0; i < bufSize; i++) {
		System.out.print(buffer[i] + " ");
	}
	System.out.println();
}

//выводит на экран первые n элементов буфера
@Override
public void PrintFirstN(int n) {
	for (int i = 0; i < n; i++) {
		System.out.print(buffer[i] + " ");
	}
	System.out.println();
}

//выводит на экран последние n элементов буфера
@Override
public void PrintLastN(int n) {
	for (int i = bufSize-n; i < bufSize; i++) {
		System.out.print(buffer[i] + " ");
	}
	System.out.println();
}

//описывает метод для сортировки массива
@Override
public void Sort() {
	long temp;
	boolean isSorted = false;
	while(!isSorted) {
		isSorted = true;
```

```
            for (int i = 1; i < buffer.length; i++) {
                  if (buffer[i] < buffer[i-1]) {
                        temp = buffer[i];
                        buffer[i] = buffer[i-1];
                        buffer[i-1] = temp;
                        isSorted = false;
                  }
            }
      }
}

//вычисляет максимальный элемент буфера
@Override
public void Max() {
      long max = buffer[0];
      for (int i = 1; i < bufSize; i++) {
            if (buffer[i] > max) {
                  max = buffer[i];
            }
      }
      System.out.println("Максимальный элемент буфера = " + max);
}

//вычисляет минимальный элемент буфера
@Override
public void Min() {
      long min = buffer[0];
      for (int i = 1; i < bufSize; i++) {
            if (buffer[i] < min) {
                  min = buffer[i];
            }
      }
      System.out.println("Минимальный элемент буфера = " + min);
}

//вычисляет сумму элементов буфера
@Override
public void Sum() {
      long sum = 0;
      for (int i = 0; i < bufSize; i++) {
            sum += buffer[i];
```

```
            }
            System.out.println("Сумма элементов буфера = " + sum);
      }

      //сохраняет буфер в файл в одну строку
      @Override
      public void SaveOneLine(String filename) {
            try {
                  File file = new File(filename);
                  PrintWriter pw = new PrintWriter(file);
                  for (int i = 0; i < buffer.length; i++) {
                        pw.print(buffer[i] + " ");
                  }
                  pw.close();
            }
            catch (IOException ex) {
                  System.out.println(ex.getMessage());
            }
      }

      //сохраняет буфер в файл по одному элементу в строке
      @Override
      public void SaveSeparateLines (String filename) {
            try {
                  File file = new File(filename);
                  PrintWriter pw = new PrintWriter(file);
                  for (int i = 0; i < buffer.length; i++) {
                        pw.println(buffer[i]);
                  }
                  pw.close();
            }
            catch (IOException ex) {
                  System.out.println(ex.getMessage());
            }
      }
}
```

Листинг 5 – Интерфейс IBufferPrintable

```
package Interfaces;

//Интерфейс описывающий методы вывода на экран
```

```
public interface IBufferPrintable {
	//выводит на экран идентификатор, тип и размер буфера
	public void PrintInfo();
	//выводит на экран содержимое буфера
	public void Print();
	//выводит на экран первые n элементов буфера
	public void PrintFirstN(int n);
	//выводит на экран последние n элементов буфера
	public void PrintLastN(int n);
}
```

Листинг 6 – Интерфейс IBufferSortable

```
package Interfaces;

//описывает метод для сортировки массива
public interface IBufferSortable {
	public void Sort();
}
```

Листинг 7 – Интерфейс IBufferComputable

```
package Interfaces;

//Интерфейс описывает методы для вычисления статистики значений буфера
public interface IBufferComputable {
	//вычисляет максимальный элемент буфера
	public void Max();
	//вычисляет минимальный элемент буфера
	public void Min();
	//вычисляет сумму элементов буфера
	public void Sum();
}
```

Листинг 8 – Интерфейс IBufferStorable

```
package Interfaces;

//описывает методы для выгрузки буфера в текстовый файл
public interface IBufferStorable {
	//сохраняет буфер в файл в одну строку
	public void SaveOneLine(String filename);
	//сохраняет буфер в файл по одному элементу в строке
	public void SaveSeparateLines (String filename);
}
```

2.3.2 Тестирование программы

Программа была скомпилирована и запущена. Результат запуска представлен на рисунке 1. Программа вывела на экран корректные, ожидаемые данные. В результате работы программы были созданы 3 текстовых файла (рисунок 2) в которых были записаны буферы в одну строку (рисунок 3).

Результаты тестирования полностью соответствуют ожиданиям.

```
Идентификатор буфера: 1
Тип буфера: long
Размер буфера: 90
Идентификатор буфера: 2
Тип буфера: long
Размер буфера: 90
Идентификатор буфера: 3
Тип буфера: long
Размер буфера: 90
-9089245628372263225 -2099417397115885989 -2509531878846043554 1090056925566921853 -5804127765110845941 953959501365702341 2852420977481717059 -8645970851363199824 7489
5553616984088348939 6712790013855762073 3248089476806207572 4866710033772282990 -3059866496001402 78 1367162868146627693 -5012462708988501750 4824943571488454381 -539615
3964579045740365059 -5131131776150729454 -1641199523177871033 -3718179964353757873 -1315537699248397569 109597467513239422 -6420159553208239008 7503717027536837189 6001
Максимальный элемент буфера = 9169466017130187285
Максимальный элемент буфера = 9172164383282289146
Максимальный элемент буфера = 8887565332907046835
-9089245628372263225 -8983176624931903406 -8794239265027858000 -8682804502289018862 -8675597000922389414 -8645970851363199824 -8565809892763765405 -8458793407054535190
-9135985280728186100 -9085769975188269259 -8957557866487273017 -8386209684601699233 -8194129795050379454 -8075325656433025309 -7379319385849007062 -7262586330248910057
-9136487809201329105 -8994615594702165643 -8729576934616439573 -8551000582673288302 -8481344091504721947 -8465237519583350260 -8232900117694327511 -8209881154582730664
```

Рисунок 1 – Результат работы программы в консоли вывода

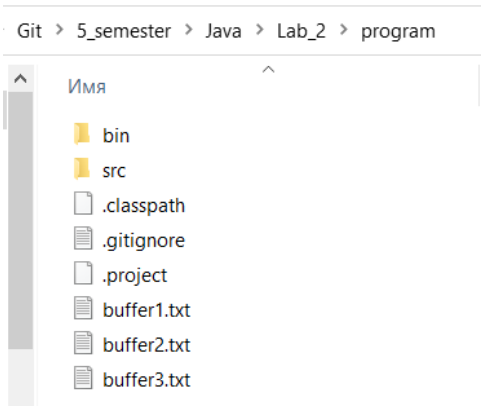

Рисунок 2 – Список созданных файлов в каталоге с программой

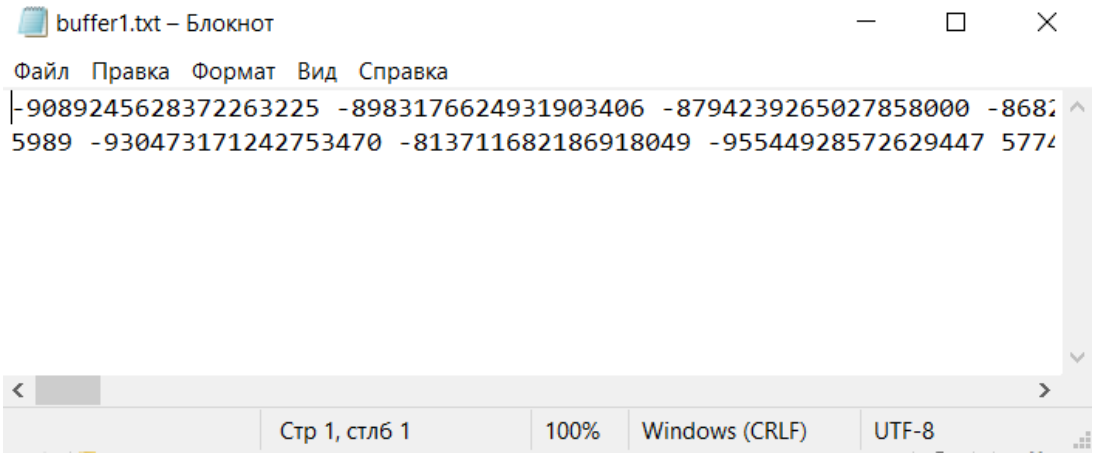

Рисунок 3 – Текстовый файл с буфером записанным в одну строку

Выводы

В ходе выполнения данной лабораторной работы ознакомились с особенностями объектно-ориентированного программирования (ООП) на языке Java, приобретены практические навыки программирования на языке Java с использованием основных принципов ООП. Были повторно изучены модификаторы доступа. Изучены интерфейсы. Полученные навыки и опыт помогут при дальнейшем программировании на языке Java.