

Объектно-ориентированное программирование на языке Java**1.1. 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.

1.2. Постановка задачи

Был выдан вариант 13.

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()`.

В соответствии с вариантом задания реализовать дочерний класс для создания буфера, хранящего значения заданного типа `Double`. Для хранения значений реализовать поле – массив значений типа `Double`. В конструкторе класса использовать вызов конструктора родительского класса `CBuffer`, и кроме того создать мас-

сив значений типа `Double` с использованием оператора `new` и проинициализировать его с использованием метода `Generate()`.

Реализовать 4 интерфейса. Создать произвольный класс, унаследованный от созданного ранее абстрактного класса и реализующий методы интерфейсов, необходимых для выполнения задания в соответствии с вариантом.

Реализовать класс `Lab2Java`, в методе `main` которого в соответствии с вариантом задания реализовать работу с объектами класса с использованием их методов: 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`.

1.3. Ход выполнения работы

1.3.1. Текст программы

Программа составлена на языке Java и представлена в листингах 1 — 7.

Листинг 1 — Основной класс `App`

```
import java.io.File;
import java.io.IOException;

public class App
{
    static final int BUFFER_INIT_SIZE = 20, BUFFER_COUNT = 5;

    public static void main (String[] args)
    {
        try
        {
            for (int i = 0; i < BUFFER_COUNT; ++i)
            {
                var buffer = new MyBuffer(BUFFER_INIT_SIZE);
                buffer.printInfo();
                buffer.printFirstN(10);
                buffer.max();
                buffer.min();
                buffer.sort();
                buffer.printLastN(10);

                if (i % 2 == 0)
                {
                    buffer.saveOneLine(File.createTempFile("lab2-", "-buffer" + buffer.getId() + "-online"));
                }
            }
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
    }
}
```

```

    }
    else
    {
        buffer.saveSeparateLines(File.createTempFile("lab2-", "-buffer" + buffer.getId() + "-multilines"));
    }
}
}
catch (IOException e)
{
    System.err.println(e.getMessage());
}
}
}

```

Листинг 2 — Класс AbstractBuffer

```

import java.util.Random;

public abstract class AbstractBuffer
{
    private static int count = 0;
    protected static final Random generator = new Random();

    protected final int id;
    protected final int size;

    public AbstractBuffer (int size)
    {
        this.id = ++count;
        this.size = size;
    }

    public final int getId ()
    { return id; }

    public final int size ()
    {
        return size;
    }

    protected abstract void generate ();
}

```

Листинг 3 — Класс IBufferComputable

```

public interface IBufferComputable
{
    public void max ();

    public void min ();

    public void sum ();
}

```

Листинг 4 — Класс IBufferPrintable

```

public interface IBufferPrintable
{
    public void printInfo ();

    public void printAll ();

    public void printFirstN (int n);

    public void printLastN (int n);
}

```

Листинг 5 — Класс IBufferSortable

```
public interface IBufferSortable
{
    public void sort ();
}
```

Листинг 6 — Класс IBufferStorable

```
import java.io.File;
import java.io.IOException;

public interface IBufferStorable
{
    public void saveOneLine (File file) throws IOException;

    public void saveSeparateLines (File file) throws IOException;
}
```

Листинг 7 — Класс MyBuffer

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.Arrays;
import java.util.stream.Collectors;

public class MyBuffer extends AbstractBuffer
implements IBufferComputable, IBufferPrintable, IBufferSortable, IBufferStorable
{
    protected final Double[] data;

    public MyBuffer (int size)
    {
        super(size);
        this.data = new Double[this.size];
        generate();
    }

    @Override
    protected void generate ()
    {
        for (int i = 0; i < size; ++i)
            data[i] = generator.nextDouble();
    }

    @Override
    public void saveOneLine (File file) throws IOException
    {
        try (var writer = Files.newBufferedWriter(file.toPath()))
        {
            for (var element : data)
            {
                writer.write(element.toString());
                writer.write(" ");
            }
        }
    }

    @Override
    public void saveSeparateLines (File file) throws IOException
    {
        try (var writer = Files.newBufferedWriter(file.toPath()))
        {
            for (var element : data)
            {
                writer.write(element.toString());
                writer.newLine();
            }
        }
    }
}
```

```

@Override
public void sort ()
{
    int i, j;
    for (int gap = data.length / 2; gap > 0; gap /= 2)
    {
        for (i = gap; i < data.length; i++)
        {
            Double tmp = data[i];
            for (j = i; j >= gap && tmp.compareTo(data[j - gap]) < 0; j -= gap)
            {
                data[j] = data[j - gap];
            }
            data[j] = tmp;
        }
    }
}

@Override
public void printInfo ()
{
    System.out.println(this.toString());
}

@Override
public void printAll ()
{
    System.out
        .println(
            "Содержимое буфера: "
            + Arrays.stream(data).map(x -> x.toString()).collect(Collectors.joining(", ", "{", "}")) + ".");
}

@Override
public void printFirstN (int n)
{
    System.out
        .println(
            "Первые " + n + " элементов: "
            + Arrays
                .stream(Arrays.copyOfRange(data, 0, n))
                .map(x -> x.toString())
                .collect(Collectors.joining(", ", "{", "}"))
            + ".");
}

@Override
public void printLastN (int n)
{
    System.out
        .println(
            "Последние " + n + " элементов: "
            + Arrays
                .stream(Arrays.copyOfRange(data, size - n, size))
                .map(x -> x.toString())
                .collect(Collectors.joining(", ", "{", "}"))
            + ".");
}

@Override
public void max ()
{
    System.out.println("Максимальное значение в буфере: " + getMaxValue() + ".");
}

private Double getMaxValue ()
{
    Double max = data[0];
    for (int i = 1; i < size; ++i)
    {
        if (data[i] > max)
        {
            max = data[i];
        }
    }
    return max;
}

```

```

@Override
public void min ()
{
    System.out.println("Минимально значение в буфере: " + getMinValue() + ".");
}

private Double getMinValue ()
{
    Double min = data[0];
    for (int i = 1; i < size; ++i)
    {
        if (data[i] < min)
        {
            min = data[i];
        }
    }
    return min;
}

@Override
public void sum ()
{
    double sum = 0.0;
    for (var element : data)
    {
        sum += element;
    }

    System.out.println("Сумма: " + sum + ".");
}

@Override
public String toString ()
{
    return String.format("ID: %d. Тип: %s. Размер: %d.", id, data[0].getClass().getName(), size);
}
}

```

1.3.2. Результаты тестирования

Программа была скомпилирована и запущена. Результат запуска представлен на рисунке 1. Программа вывела на экран ожидаемые данные. В результате работы было создано во временном каталоге операционной системы 5 файлов (рисунок 2), каждый из которых соответствует содержимому буфера. Рисунок 3 демонстрирует содержимое файла: элементы буфера, как и ожидалось, записаны в одну строчку. Рисунок 4 демонстрирует содержимое другого файла: каждый элемент буфера записан на новой строчке.

```
alex@handpan:/tmp/test
java -jar Lab_2.jar
ID: 1. Тип: java.lang.Double. Размер: 20.
Первые 10 элементов: {0.9682158480688602; 0.23422571602734454; 0.8376365683760806; 0.5659474282262578; 0.6211495754839552; 0.3214598584921785; 0.07843805542878557; 0.2338143121583629; 0.6197101665864402; 0.4165767377082432}.
Максимальное значение в буфере: 0.9682158480688602.
Минимально значение в буфере: 0.05080998510221124.
Последние 10 элементов: {0.5659474282262578; 0.612161048624499; 0.6197101665864402; 0.6211495754839552; 0.6636231807201782; 0.666841962992864; 0.703149254522432; 0.8347987752299739; 0.8376365683760806; 0.9682158480688602}.
ID: 2. Тип: java.lang.Double. Размер: 20.
Первые 10 элементов: {0.9055071000106122; 0.5122676961884588; 0.2795393733359236; 0.2752133605050473; 0.9718342339383542; 0.30088701158761133; 0.2834962062392957; 0.26928568628297844; 0.6929042529190056; 0.9092634481802144}.
Максимальное значение в буфере: 0.9955040475413319.
Минимально значение в буфере: 0.18670659055585437.
Последние 10 элементов: {0.5838945084752634; 0.5122676961884588; 0.6929042529190056; 0.7438012340794195; 0.8654923161974888; 0.9055071000106122; 0.9092634481802144; 0.9718342339383542; 0.9732974553584699; 0.9955040475413319}.
ID: 3. Тип: java.lang.Double. Размер: 20.
Первые 10 элементов: {0.7517485004859228; 0.48705455880477544; 0.40871258912046604; 0.23104291544539401; 0.43511697432568464; 0.2058043432062211; 0.3288731685950753; 0.6845223178512545; 0.6725251343826565; 0.42820358930170455}.
Максимальное значение в буфере: 0.98578852535076.
Минимально значение в буфере: 0.06417280566763583.
Последние 10 элементов: {0.4296230081958121; 0.43511697432568464; 0.48705455880477544; 0.5704992765533529; 0.6725251343826565; 0.6845223178512545; 0.7490986425960837; 0.7517485004859228; 0.8945454832084049; 0.98578852535076}.
ID: 4. Тип: java.lang.Double. Размер: 20.
Первые 10 элементов: {0.8513030497172297; 0.9711545057281953; 0.9651458746847746; 0.5324656657537047; 0.6813863869656444; 0.7762311326424609; 0.05503298801597312; 0.895529403257416; 0.06881410532360921; 0.8174629592727565}.
Максимальное значение в буфере: 0.9938079968851009.
Минимально значение в буфере: 0.03624068643719569.
Последние 10 элементов: {0.7788071269417731; 0.8166391379436561; 0.8174629592727565; 0.8513030497172297; 0.8592735269248392; 0.895529403257416; 0.9651458746847746; 0.9711545057281953; 0.9884001407794679; 0.9938079968851009}.
ID: 5. Тип: java.lang.Double. Размер: 20.
Первые 10 элементов: {0.08297416843324801; 0.6278651442656965; 0.9746637231885995; 0.6669287390685166; 0.5743794087078511; 0.3830282403549361; 0.02472663385375995; 0.7890662436278233; 0.1988554018985217; 0.4524120123498777}.
Максимальное значение в буфере: 0.9864878443113269.
Минимально значение в буфере: 0.0035393864344843643.
Последние 10 элементов: {0.5382686787591134; 0.5743794087078511; 0.6278651442656965; 0.6637004771050561; 0.6669287390685166; 0.754894398361403; 0.7890662436278233; 0.8105829882132969; 0.9746637231885995; 0.9864878443113269}.
```

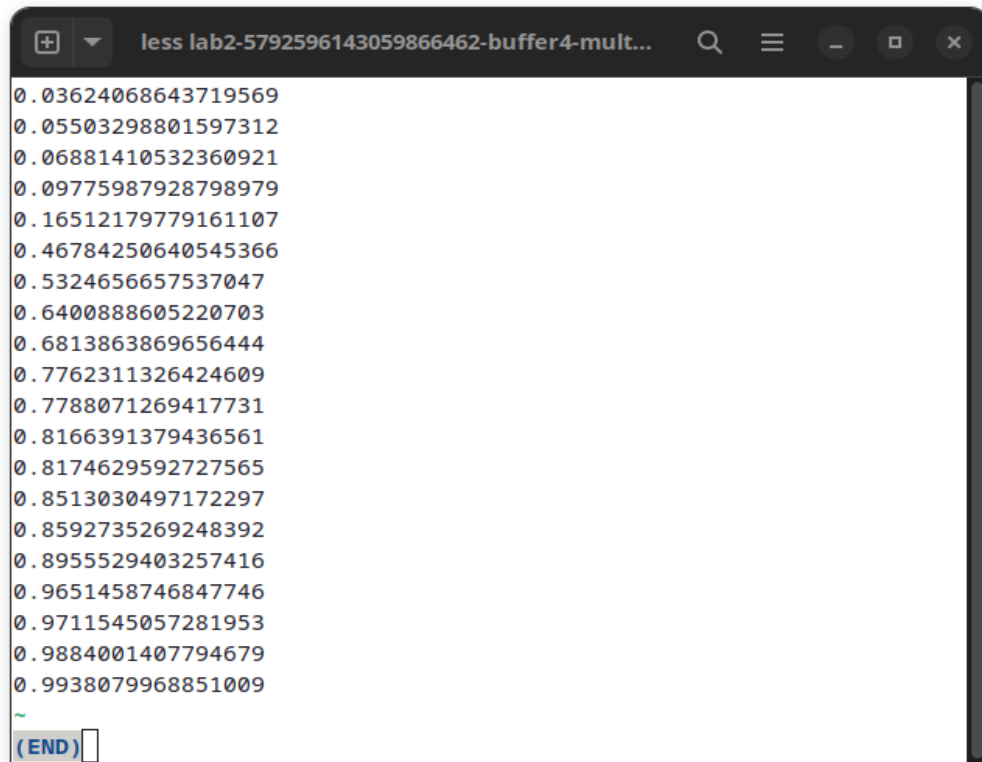
Рисунок 1 — Результат работы программы

```
alex@handpan:/tmp
ls lab2-*
-rw-r--r-- 1 alex alex 386 Sep 11 22:07 lab2-11971226432704790004-buffer3-oneline
-rw-r--r-- 1 alex alex 385 Sep 11 22:07 lab2-17891163206277436132-buffer2-multilines
-rw-r--r-- 1 alex alex 382 Sep 11 22:07 lab2-4303051326428348736-buffer1-oneline
-rw-r--r-- 1 alex alex 386 Sep 11 22:07 lab2-5792596143059866462-buffer4-multilines
-rw-r--r-- 1 alex alex 387 Sep 11 22:07 lab2-6613516195517197748-buffer5-oneline
```

Рисунок 2 — Список созданных файлов

```
less lab2-4303051326428348736-buffer1-oneline
0.05080998510221124 0.05942155898711632 0.07843805542878557 0.2338143121583629 0.23422571602734454 0.29018077111663343 0.296904
0045502881 0.3192968152521839 0.3214598584921785 0.4165767377082432 0.5659474282262578 0.612161048624499 0.6197101665864402 0.6
211495754839552 0.6636231807201782 0.666841962992864 0.703149254522432 0.8347987752299739 0.8376365683760806 0.9682158480688602
(END)
```

Рисунок 3 — Содержимое буфера, записанное в одну строку



```
less lab2-5792596143059866462-buffer4-mult...
0.03624068643719569
0.05503298801597312
0.06881410532360921
0.09775987928798979
0.16512179779161107
0.46784250640545366
0.5324656657537047
0.6400888605220703
0.6813863869656444
0.7762311326424609
0.7788071269417731
0.8166391379436561
0.8174629592727565
0.8513030497172297
0.8592735269248392
0.8955529403257416
0.9651458746847746
0.9711545057281953
0.9884001407794679
0.9938079968851009
~
(END)
```

Рисунок 4 — Содержимое буфера, записанное на нескольких строках

Вывод

При выполнении данной лабораторной работы были получены навыки создания классов, абстрактных классов, интерфейсов, объявления методов и свойств в них и принципы наследования классов в языке Java.