# ЛАБОРАТОРНАЯ РАБОТА №2

# ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ JAVA

Работы выполнил: Мовенко Константин, ИС/б-21-2-о

Цель работы

Ознакомиться с особенностями объектно-ориентированного программирования (ООП) на языке Java, приобрести практические навыки программирования на языке Java с использованием основных принципов ООП.

Задачи

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 создаваемого буфера по порядку), размера буфера (значением count, передаваемым конструктору), увеличение количества созданных буферов.

В классе CBuffer описать абстрактный метод Generate().

1. В соответствии с вариантом задания (Таблица 1) реализовать дочерний класс для создания буфера, хранящего значения заданного типа T. 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 использованием метода Generate().

Реализовать метод Generate(), заполняющий массив случайными числами.

1. Описать интерфейсы:

1) IBufferPrintable – описывающий методы вывода на экран:

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

2) IBufferSortable – описывает метод для сортировки массива:

* **public void** Sort();

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

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

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

* **public void** SaveOneLine(String filename) – сохраняет буфер в файл в одну строку;
* **public void** SaveSeparateLines(String filename) – сохраняет буфер в файл по одному элементу в строке;

1. Создать произвольный класс, унаследованный от класса, разработанного при выполнении п.2, и реализующий методы интерфейсов из п.3, необходимых для выполнения задания в соответствии с вариантом (Таблица 1).
2. Реализовать класс Lab2Java, в методе main которого в соответствии с вариантом задания реализовать работу с объектами класса из п.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 последние 10 элементов буферов;
* Сохранить буферы в файл с использованием метода O;

Вариант задания

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Количество буферов (N) | Типы элементов буфера (T) | Число элементов в буферах (L) | Сортировка (S) | Вычисление (F) | Сохранение (O) |
| 13 | 5 | Double | 50 | Шелла | Min | SaveOneLine |

Ход работы

В среде разработки Eclipse был создан проект LAB\_2, в котором были созданы и написаны указанные в задании классы и интерфейсы. Далее был начат процесс тестирования полученной программы.

Программа начинается с выполнения метода main класса Lab2Java. В консоль была выведена информация об операциях над пятью созданными буферами (Рисунок 1). Программа способна автоматически нумеровать и подсчитывать буферы, выводить и сортировать их содержимое.

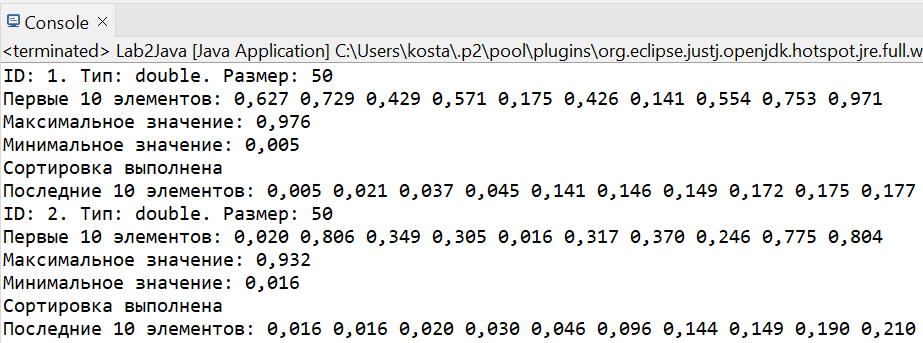


Рисунок 1 – Результат выполнения программы

С помощью метода SaveOneLine класса Buffer1 отсортированное содержимое буферов было сохранено в виде одной строки в пять различных файлов (Рисунок 2).

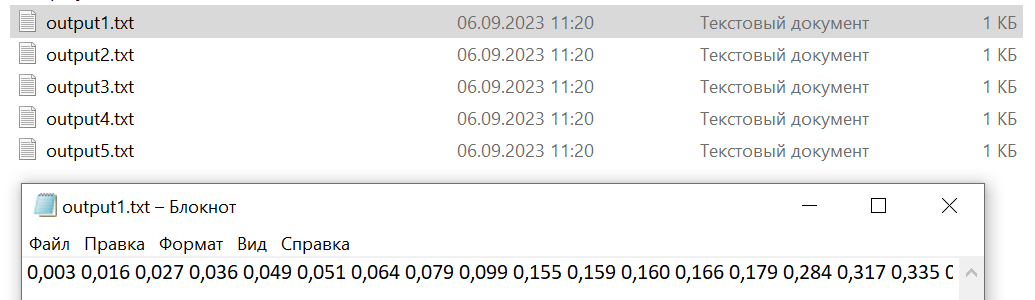


Рисунок 2 – Проверка метода SaveOneLine

Также был рассмотрен метод SaveSeparateLines, делающий то же самое, но не в одну строку, а построчно (Рисунок 3).

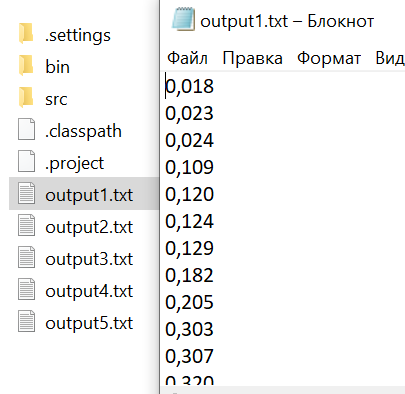


Рисунок 3 – Проверка метода SaveSeparateLines

Текст программы

Листинг 1 − Lab2Java.java:

**import** java.io.IOException;

**public** **class** Lab2Java

{

**static** **final** **int** ***L*** = 50, ***N*** = 5;

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

{

**for** (**int** i = 0; i < ***N***; i++)

{

**Buffer** buffer = **new** Buffer(***L***);

buffer.PrintInfo();

buffer.PrintFirstN(10);

buffer.Max();

buffer.Min();

buffer.Sort();

buffer.PrintLastN(10);

**try**

{

buffer.SaveOneLine("output" + buffer.GetBufID() + ".txt");

}

**catch** (IOException e)

{

System.***err***.println(e.getMessage());

}

}

}

}

Листинг 2 − CBuffer.java:

**public** **abstract** **class** CBuffer

{

**private** **static** **int** *bufCount* = 0; // количество буферов

**protected** **final** **int** bufID; // идентификатор

**protected** **final** **int** bufSize; // размер буфера

**public** CBuffer(**int** count) // конструктор класса

{

bufID = ++*bufCount*;

bufSize = count;

}

**public** **final** **int** GetBufID()

{ **return** bufID; }

**public** **final** **int** GetBufCount()

{ **return** *bufCount*; }

**protected** **abstract** **void** Generate();

}

Листинг 3 − Buffer.java:

**import** java.util.Random;

**public** **class** Buffer **extends** CBuffer

{

**protected** **double**[] bufArray; // массив для чисел

**protected** **static** **final** Random ***random*** = **new** Random(); // ГСЧ

**public** Buffer(**int** size) // конструктор класса

{

**super**(size);

bufArray = **new** **double**[bufSize];

Generate();

}

@Override

**protected** **void** Generate() // заполнение буфера случайными числами

{

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

bufArray[i] = ***random***.nextDouble();

}

}

Листинг 4 − IBufferPrintable.java:

**public** **interface** IBufferPrintable

{

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

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

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

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

}

Листинг 5 − IBufferSortable.java:

**public** **interface** IBufferSortable

{

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

}

Листинг 6 − IBufferComputable.java:

**public** **interface** IBufferComputable

{

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

**public** **void** Min();

**public** **void** Sum();

}

Листинг 7 − IBufferStorable.java:

**import** java.io.IOException;

**public** **interface** IBufferStorable

{

**public** **void** SaveOneLine(String filename) **throws** IOException;

**public** **void** SaveSeparateLines(String filename) **throws** IOException;

}

Листинг 8 – Buffer1.java

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**public** **class** Buffer1 **extends** Buffer

**implements** IBufferComputable, IBufferPrintable, IBufferSortable, IBufferStorable

{

**public** Buffer1(**int** size) // конструктор класса

{

**super**(size);

}

@Override

**public** **void** PrintInfo() // вывод информации о буфере

{

System.***out***.printf("ID: %d. Тип: double. Размер: %d\n", bufID, bufSize);

}

@Override

**public** **void** Print() // вывод содержимого буфера

{

System.***out***.print("Содержимое буфера: ");

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

{

System.***out***.printf("%.3f ", bufArray[i]);

}

System.***out***.printf("\n");

}

@Override

**public** **void** PrintFirstN(**int** n) // первые N элементов

{

System.***out***.printf("Первые %d элементов: ", n);

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

{

System.***out***.printf("%.3f ", bufArray[i]);

}

System.***out***.printf("\n");

}

@Override

**public** **void** PrintLastN(**int** n) // последние N элементов

{

System.***out***.printf("Последние %d элементов: ", n);

**for** (**int** i = bufSize-n; i < bufSize; i++)

{

System.***out***.printf("%.3f ", bufArray[i]);

}

System.***out***.printf("\n");

}

@Override

**public** **void** Sort() // сортировка буфера (Шелла)

{

**int** i, j;

**for** (**int** gap = bufSize / 2; gap > 0; gap /= 2)

{

**for** (i = gap; i < bufSize; i++)

{

**double** temp = bufArray[i];

**for** (j = i; j >= gap && bufArray[j - gap] > temp; j -= gap)

{

bufArray[j] = bufArray[j - gap];

}

bufArray[j] = temp;

}

}

System.***out***.println("Сортировка выполнена");

}

@Override

**public** **void** Max() // максимальный элемент

{

**double** max = bufArray[0];

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

{

**if** (bufArray[i] > max)

{

max = bufArray[i];

}

}

System.***out***.printf("Максимальное значение: %.3f\n", max);

}

@Override

**public** **void** Min() // минимальный элемент

{

**double** min = bufArray[0];

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

{

**if** (bufArray[i] < min)

{

min = bufArray[i];

}

}

System.***out***.printf("Минимальное значение: %.3f\n", min);

}

@Override

**public** **void** Sum() // сумма элементов буфера

{

**double** sum = 0.0;

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

{

sum += bufArray[i];

}

System.***out***.printf("Сумма элементов: %.3f\n", sum);

}

@Override // вывод в файл одной строкой

**public** **void** SaveOneLine(String filename) **throws** IOException

{

**try**

{

PrintWriter writer = **new** PrintWriter(**new** FileWriter(filename));

//writer.println("");

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

{

writer.printf("%.3f ", bufArray[i]);

}

writer.close();

} **catch** (IOException e) {

**throw** e;

}

}

@Override // вывод в файл построчно

**public** **void** SaveSeparateLines(String filename) **throws** IOException

{

**try**

{

PrintWriter writer = **new** PrintWriter(**new** FileWriter(filename));

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

{

writer.printf("%.3f\n", bufArray[i]);

}

writer.close();

} **catch** (IOException e) {

**throw** e;

}

}

}

Вывод

В ходе работы было проведено ознакомление с особенностями и принципами объектно-ориентированного программирования на языке Java. Были получены навыки работы с классами, абстрактными классами и интерфейсами.

Результатом работы стала программа, производящая работу с несколькими классами. При её написании были использованы основные принципы ООП: наследование (от абстрактного класса), полиморфизм (переопределение методов наследуемого класса, интерфейсы), инкапсуляция (выделение методов и полей модификаторами доступа).