МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

Институт информационных технологий и управления в технических системах

кафедра Информационные системы

09.03.02 Информационные системы и технологии (уровень бакалавриата)

Лабораторная работа №2

по дисциплине: «Платформа Java»

ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ JAVA

Вариант – 8

Выполнил

студент 3 курса группы ИС/б-33-о

Генералов Николай Николаевич

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

(дата)

Проверил

ст. пр. \_\_\_\_\_\_\_\_\_\_\_ Кузнецов С. А.

(должность) (подпись) (фамилия, инициалы)

г. Севастополь

2018 г.

# ЦЕЛЬ РАБОТЫ

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

# ПОСТАНОВКА ЗАДАЧИ

1. Описать абстрактный класс CBuffer, содержащий следующие поля:

* идентификатор буфера (intBufID) –уникальный идентификатор буфера;
* размер буфера (intBufSize) –максимальный размер буфера;
* количество созданных буферов(intBufCount).

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:

* intGetBufCount();
* intGetBufID().

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().

1. Реализовать дочерний класс для создания буфера, хранящего значения типа int.Для хранения значений реализовать поле – массив значений типа int. В конструкторе класса использовать вызов конструктора родительского класса CBuffer, и кроме того создать массив значений типа int (с использованием оператора new)и проинициализировать его с имcпользованием 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.

Генерация случайных чисел выполняется методами экземпляра класса Random.

1. Описать интерфейсы:

* 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.

* IBufferSortable–описывает метод для сортировки массива:

public void Sort();

* IBufferComputable–описывает методы для вычисления статистики значений буфера.

publicvoidMax()–вычисляет максимальный элемент буфера;

publicvoidMin()–вычисляет минимальный элемент буфера;

publicvoidSum()–вычисляет сумму элементов буфера;

* 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.2.2, и реализующий методы интерфейсов из п.2.3, необходимых для выполнения задания в соответствии с вариантом.

1. Реализовать класс Lab2Java, в методе main которого в соответствии с вариантом задания реализовать работу с объектами класса из п.2.4 с использованием их методов:

* Создать 5 буферов заданного типа и размера L;
* Вывести на экран информацию o буферах;
* Вывести на экран первые 10 элементов буферов;
* Вычислить функцию Max() для каждого буфера и вывести результат на экран;
* Выполнить сортировку буферов методом «Быстрой сортировки»;
* Вывести на экран первые 10 элементов буферов;
* Сохранить буферы в файл с использованием метода SaveSeparateLines(String filename).

# ХОД РАБОТЫ

В соответствии с постановкой задачи была разработана программа, модульная структура которой изображена на рисунке 1.

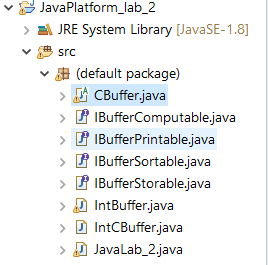


Рисунок 1 – Модульная структура программы

Метод main, последовательно выполняет набор операторов выполняющих задачи поставленные в п.2.5.

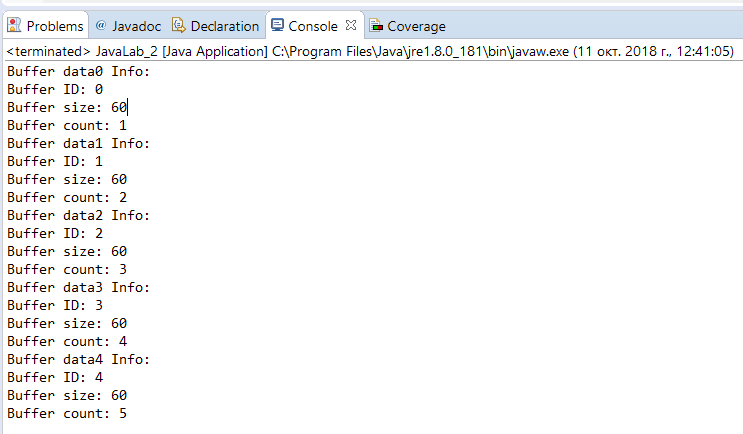


Рисунок 2 – Создание буферов и вывод информации о них

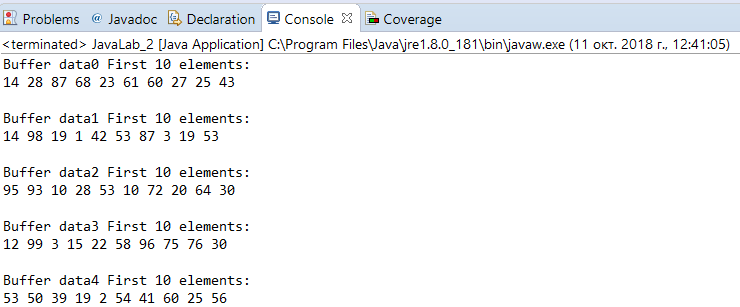


Рисунок 3 – Вывод содержимого первых 10 элементов буферов в консоль

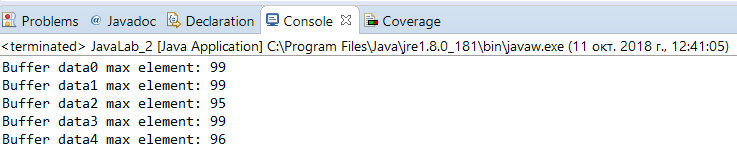


Рисунок 4 – Выполнение функции Max() – Поиска максимального элемента массива

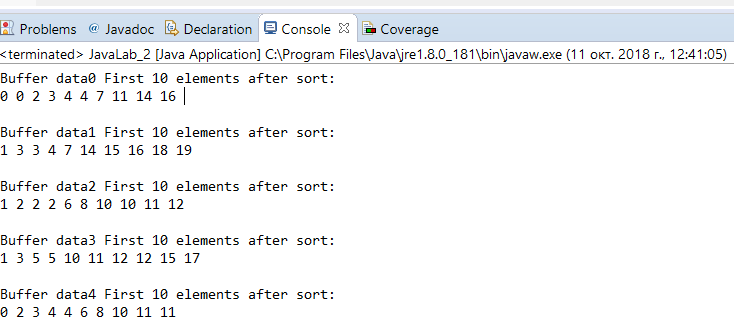


Рисунок 5 – Вывод содержимого первых 10 элементов буферов в консоль, после выполнения сортировки

В качестве дополнительных были реализованы:

* метод finallize() класса CBuffer – осуществляющий декрементацию счетчика буферов, при их утилизации из памяти сборщиком мусора;
* метод QuickSort класса IntCBuffer, имеющий сигнатуру public static void QuickSort(int[] \_array, int start, int end), непосредственно осуществляющий сортировку переданного ему массива.

Для методов, не требующих реализации по варианту задания, были поставлены заглушки.

# ВЫВОД

В ходе выполнения лабораторной работы были исследованы основные средства и особенности объектно-ориентированного программирования на языке Java. Для закрепления теоретических знаний, а так же освоения основных принципов объектно-ориентированных программ, таких как: инкапсуляция, наследование и полиморфизм, была разработана программа выполняющая задачи поставленные в п.2 данной лабораторной работы.

По результатам выполнения программы можно сделать вывод, что все поставленные задачи выполнены и программа работоспособна, и поставленная цель достигнута.

Приложение А

Текст программы

«JavaLab\_2.java»

import java.util.Scanner;

import java.util.InputMismatchException;

public class JavaLab\_2 {

public static final int buffCount = 5;

public static final int buffSize = 60;

public static void main(String[] args) {

IntBuffer[] data = new IntBuffer[buffCount];

for (int i = 0; i < data.length; ++i) {

data[i] = new IntBuffer(buffSize);

System.out.println("Buffer data" + data[i].GetBudID() + " Info: ");

data[i].PrintInfo();

}

for (int i = 0; i < data.length; ++i) {

System.out.print("\nBuffer data" + data[i].GetBudID() + " First 10 elements: ");

data[i].PrintFirstN(10);

System.out.println();

}

for (int i = 0; i < data.length; ++i) {

System.out.println("Buffer data" + data[i].GetBudID() + " max element: " + data[i].Max());

}

for (int i = 0; i < data.length; ++i) {

data[i].Sort();

}

for (int i = 0; i < data.length; ++i) {

System.out.print("\nBuffer data" + data[i].GetBudID() + " First 10 elements after sort: ");

data[i].PrintFirstN(10);

System.out.println();

}

for (int i = 0; i < data.length; ++i) {

data[i].SaveSeparateLines("logFile" + data[i].GetBudID() + ".txt");;

}

return;

}

}

«IntBuffer.java»

import java.io.\*;

public class IntBuffer extends IntCBuffer

implements IBufferComputable, IBufferPrintable, IBufferSortable, IBufferStorable {

public IntBuffer(int count) {

super(count);

// TODO Auto-generated constructor stub

}

@Override

public void SaveOneLine(String filename) {

// TODO Auto-generated method stub

}

@Override

public void SaveSeparateLines(String filename) {

// TODO Auto-generated method stub

PrintWriter outFile = null;

try {

outFile = new PrintWriter(filename);

for (int i = 0; i < array.length; ++i) {

outFile.println(array[i]);

}

} catch(FileNotFoundException ex) {

System.out.println(ex);

System.exit(0);

}

finally{

if (outFile != null) {

outFile.close();

}

}

}

@Override

public void Sort() {

QuickSort(array, 0, array.length-1);

}

public static void QuickSort(int[] \_array, int start, int end) {

if (start >= end){

return;

}

int i = start;

int j = end;

int midElIndex = (start + end) / 2;

while (i < j) {

while (\_array[i] <= \_array[midElIndex] && i < midElIndex){

++i;

}

while (\_array[j] >= \_array[midElIndex] && midElIndex < j) {

--j;

}

if (i <j) {

int temp = \_array[i];

\_array[i] = \_array[j];

\_array[j] = temp;

if (i == midElIndex)

midElIndex = j;

else if (j == midElIndex)

midElIndex = i;

}

}

QuickSort(\_array, start, midElIndex);

QuickSort(\_array, midElIndex + 1, end);

return;

}

@Override

public void PrintInfo() {

// TODO Auto-generated method stub

System.out.println("Buffer ID: " + this.bufID);

System.out.println("Buffer size: " + this.bufSize);

System.out.println("Buffer count: " + this.bufCount);

}

@Override

public void Print() {

// TODO Auto-generated method stub

for(int i = 0; i < array.length; ++i) {

if (i % 10 == 0) {

System.out.println();

}

System.out.println(array[i] + " ");

}

}

@Override

public void PrintFirstN(int n) {

// TODO Auto-generated method stub

for(int i = 0; (i < array.length) && (i < n); ++i) {

if (i % 10 == 0) {

System.out.println();

}

System.out.print(array[i] + " ");

}

}

@Override

public void PrintLastN(int n) {

// TODO Auto-generated method stub

for(int i = array.length - 1; (i >= 0) && (array.length - i - 1 < n); --i) {

if (i % 10 == 0) {

System.out.println();

}

System.out.print(array[i] + " ");

}

}

@Override

public int Max() {

// TODO Auto-generated method stub

int indexOfMax = 0;

for (int i = 0; i < array.length; ++i) {

if (array[i] >= array[indexOfMax]) {

indexOfMax = i;

}

}

return array[indexOfMax];

}

@Override

public int Min() {

// TODO Auto-generated method stub

return 0;

}

@Override

public int Sum() {

// TODO Auto-generated method stub

return 0;

}

}

«IntCBuffer.java»

import java.util.Random;

public class IntCBuffer extends CBuffer {

public int[] array;

public IntCBuffer(int count) {

super(count);

array = new int[count];

Generate();

}

@Override

protected void Generate() {

Random randomGenerator = new Random();

try {

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

array[i] = randomGenerator.nextInt(100);

}

} catch(Exception ex) {

System.out.println(ex.toString());

}

}

}

«IBufferStorable.java»

public interface IBufferStorable {

void SaveOneLine(String filename);

void SaveSeparateLines(String filename);

}

«IBufferSortable.java»

public interface IBufferSortable {

void Sort();

}

«IBufferPrintable.java»

public interface IBufferPrintable {

void PrintInfo();

void Print();

void PrintFirstN(int n);

void PrintLastN(int n);

}

«IBufferComputable.java»

public interface IBufferComputable {

int Max();

int Min();

int Sum();

}

«CBuffer.java»

public abstract class CBuffer {

protected int bufID = 0;

protected int bufSize = 0;

protected static int bufCount = 0;

public CBuffer(int count) {

bufID = bufCount++;

bufSize = count;

}

protected abstract void Generate();

public int GetBufCount() {

return bufCount;

}

public int GetBudID() {

return bufID;

}

private void finallize() {

// TODO Auto-generated method stub

--bufCount;

}}