Министерство образования и науки РФ

Институт информационных технологий и управления в технических системах

Севастопольский государственный университет

Лабораторная работа №2

по дисциплине: «Платформа Java»

Выполнил:

ст.гр. ИСб/22о

Воронин И.Ю.

Проверил:

Кузнецов С.А.

г. Севастополь

2016 г.

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. Вариант задания

Описать абстрактный класс CBuffer, содержащий следующие поля:

В соответствии с вариантом задания реализовать дочерний класс для создания буфера. Описать интерфейсы:

Создать произвольный класс, унаследованный от класса, 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 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.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Количество буферов (N) | Типы элементов буфера (T) | Число элементов в буферах (L) | Сортировка (S) | Вычисление (F) | Сохранение (O) |
| 2 | 4 | Long | 60 | Выборки | Max | SaveSeparateLines |

Таблица 2.1 – Вариант задания.

1. Код программы

**package** Interface;

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

**public** **long** Max(**long** []arr);

}

**package** Interface;

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

**public** **void** PrintInfo(**int** bufID,String typeName, **int** bufSize);

**public** **void** Print(**long** []arr);

**public** **void** PrintFirstN(**long** []arr, **int** n);

**public** **void** PrintLastN(**long** []arr, **int** n);

}

**package** Interface;

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

**public** **void** Sort(**long**[] arr);

}

**package** Interface;

**import** java.io.IOException;

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

//сохраняет буфер в файл по одному элементу в строке;

**public** **void** SaveSeparateLines(String filename, **long** []arr) **throws** IOException;

}

**package** Interfaces.Implementation;

**import** Interface.IBufferComputable;

**public** **class** ImplComputable **implements** IBufferComputable {

//по заданию

@Override

**public** **long** Max(**long** []arr) {

**int** max = (**int**) arr[0];

**for**(**int** i=1; i<arr.length; i++){

**if**(arr[i] > max) max = (**int**)arr[i];

}

**return** max;

}

}

**package** Interfaces.Implementation;

**import** Interface.IBufferPrintable;

**public** **class** ImplPrintable **implements** IBufferPrintable {

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

@Override

**public** **void** PrintInfo(**int** bufID,String typeName, **int** bufSize) {

System.***out***.println("\nID: " + bufID);

System.***out***.println("Type: " + typeName);

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

}

**public** **void** Print(**long** []arr) {

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

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

}

System.***out***.println();

}

**public** **void** PrintFirstN(**long** []arr, **int** n) {

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

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

}

System.***out***.println();

}

**public** **void** PrintLastN(**long** []arr, **int** n) {

**for**(**int** i=arr.length; i>0 & i>arr.length-n; i--){

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

}

System.***out***.println();

}

}

**package** Interfaces.Implementation;

**import** Interface.IBufferSortable;

**public** **class** ImplSortable **implements** IBufferSortable{

@Override

**public** **void** Sort(**long**[] arr) {

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

/\*Предполагаем, что первый элемент (в каждом

подмножестве элементов) является минимальным \*/

**long** min = arr[i];

**long** min\_i = i;

/\*В оставшейся части подмножества ищем элемент,

который меньше предположенного минимума\*/

**for** (**int** j = i+1; j < arr.length; j++) {

//Если находим, запоминаем его индекс

**if** (arr[j] < min) {

min = arr[j];

min\_i = j;

}

}

/\*Если нашелся элемент, меньший, чем на текущей позиции,

меняем их местами\*/

**if** (i != min\_i) {

**long** tmp = arr[i];

arr[i] = arr[(**int**) min\_i];

arr[(**int**) min\_i] = tmp;

}

}

}

}

package Interfaces.Implementation;

import java.io.FileWriter;

import java.io.IOException;

import Interface.IBufferStorable;

public class ImplStorable implements IBufferStorable {

@Override

public void SaveSeparateLines(String filename, long []arr) throws IOException{

FileWriter fout = new FileWriter(filename);

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

fout.write("" + arr[i] + ".\n");

}

fout.close();

}

}

Класс CBuffer:

**package** javaLab2;

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

**protected** String typeName;

**protected** **final** **int** bufID;

**protected** **int** bufSize;

**protected** **static** **int** *bufCount* = 0;

**public** **int** GetBufCount(){

**return** *bufCount*;

}

**public** String getTypeName(){

**return** **this**.typeName;

}

**public** **int** GetBufID(){

**return** bufID;

}

CBuffer(**int** count){

bufID = *bufCount*++;

bufSize = count;

}

**public** **void** Generate(){

//code

}

}

Класс ExtraLomgBuffer:

package javaLab2;

import java.io.IOException;

import Interface.IBufferComputable;

import Interface.IBufferPrintable;

import Interface.IBufferSortable;

import Interface.IBufferStorable;

import Interfaces.Implementation.\*;

public class ExtraLongBuffer extends LongBuffer {

ExtraLongBuffer(int count) {

super(count);

}

public void PrintInf(){

IBufferPrintable IBF = new ImplPrintable();

IBF.PrintInfo(this.bufID,this.typeName,this.bufSize);

}

public void Sort(){

IBufferSortable IBSort = new ImplSortable();

IBSort.Sort(arr);

}

public void Print(){

IBufferPrintable IBPrint = new ImplPrintable();

IBPrint.Print(arr);

}

public void PrintFirstN(int n){

IBufferPrintable IBPrintable = new ImplPrintable();

IBPrintable.PrintFirstN(arr, n);

}

public void PrintLastN(int n){

IBufferPrintable IBPrintable = new ImplPrintable();

IBPrintable.PrintLastN(arr, n);

}

public long Max(){

IBufferComputable IBComputable = new ImplComputable();

return IBComputable.Max(arr);

}

public void SaveSeparateLines(String Filename) throws IOException{

IBufferStorable IBStorable = new ImplStorable();

IBStorable.SaveSeparateLines(Filename,arr);

}

}

Класс LongBuffer:

**package** javaLab2;

**import** java.util.Random;

**public** **class** LongBuffer **extends** CBuffer {

LongBuffer(**int** count) {

**super**(count);

arr = **new** **long**[count];

Generate();

typeName = "long";

}

**protected** **long**[] arr;

//генератор случайных чисел

**public** **void** Generate(){

Random random = **new** Random();

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

arr[i] = random.nextLong()%10000;

}

}

}

Класс Main:

**package** javaLab2;

**import** java.io.IOException;

**public** **class** Main {

//параметры заданный вариантом задания

**private** **static** **int** *L* = 60;

**private** **static** **int** *N* = 4;

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

ExtraLongBuffer []var = **new** ExtraLongBuffer[4];

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

var[i] = **new** ExtraLongBuffer(*L*);

}

//информация о буферах

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

var[i].PrintInf();

}

//первые десять элементов массива

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

var[i].PrintFirstN(10);

}

//максимальное значение в массиве

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

System.***out***.println("Max " + var[i].GetBufID() + " = " + var[i].Max());

}

//сортировка для массивов

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

var[i].Sort();

}

//первые десять элементов массива

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

var[i].PrintFirstN(10);

}

//первые десять элементов массива

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

var[i].SaveSeparateLines(var[i].GetBufID() + " buffer.txt");

}

}

}

1. Выполнение программы

При запуске программы на экран будет выводится следующий текст, который необходимо было вывести по варианту задания.

ID: 0

Type: long

Buffer size: 60

ID: 1

Type: long

Buffer size: 60

ID: 2

Type: long

Buffer size: 60

ID: 3

Type: long

Buffer size: 60

-4570 -8873 6171 -5572 5725 -380 7641 4881 8050 4192

-3169 2998 -4052 2164 -3572 9345 4917 -8438 -760 -3458

7640 6847 -5206 -5315 -7684 -8366 9087 -3475 1561 -4112

-7832 -8694 1429 4545 3265 3517 -701 -6503 4287 8257

Max 0 = 8999

Max 1 = 9345

Max 2 = 9835

Max 3 = 9899

-9296 -9225 -9173 -8873 -7943 -7666 -7091 -6532 -6188 -5572

-9188 -8826 -8589 -8438 -8093 -7974 -7665 -7061 -6665 -6360

-9316 -9315 -9249 -8595 -8366 -8330 -7813 -7684 -6679 -6350

-9990 -9830 -9400 -8949 -8767 -8694 -8551 -7832 -7719 -7129

Изначально выводятся данные о всех созданных буферах, далее необходимо было вывести на экран первые десять элементов каждого буфера, после этого найти их максимальные значение. Далее необходимо было применить сортировку для каждого буфера и снова вывести первые элементов.

ВЫВОДЫ

Была написана программа на языке Java в среде программирования Eclipse Mars. Были рассмотрены методы наследования, использование интерфейсов.