Министерство науки и высшего образования РФ

Севастопольский государственный университет

Кафедра информационных систем

ОТЧЕТ

по лабораторной работе №3

ИССЛЕДОВАНИЕ КОЛЛЕКЦИЙ И ИТЕРАТОРОВ В ЯЗЫКЕ JAVA

по дисциплине «Платформа Java»

Выполнил:

Студент группы ИС/б 17-2-о

Черняев Н.Г.

Проверил:

Кузнецов С.А.

г. Севастополь 2019

**1.Цель работы**

В ходе выполнения данной лабораторной работы необходимо ознакомиться с организацией коллекций объектов на языке Java, приобрести практические навыки использования списков, очередей, хеш-таблиц при создании Java программ.

**2.Постановка задачи**

**Вариант №10**

2.1. В соответствии с вариантом задания реализовать класс Смартфон(Модель, Размер экрана, Тип экрана, Объем встроенной флэшпамяти) для представления требуемой информации.

2.2. Реализовать коллекцию типа HashSet объектов разработанного в п. 2.1. класса с возможностью ввода элементов из файла, вывода на консоль, проверки членства по введенному с консоли значению поля 1. Имя файла вводить параметром командной строки –i.

2.3. Реализовать коллекцию типа LinkedList объектов разработанного в п. 2.1. с возможностью: упорядочивания по полю 1 (использовать Collections.sort(list)); с возможностью упорядочивания по полю 2 (см. таблицу 4.1) в направлении возрастания класса (использовать Collections.sort(list, myComp), где myComp – экземпляр разработанного класса, реализующего интерфейс Comparator); с возможностью ввода элементов из файла, вывода на консоль и сохранения в файл. Имена файлов вводить параметрами командной строки –i и –o.

2.4. Реализовать коллекцию типа HashMap объектов разработанного в п. 2.1. класса c ключом по значению поля 1, с возможностью ввода элементов из файла, вывода на консоль в виде «Ключ -> Значения» (значения остальных полей), вывода значения полей по введенному с консоли значению поля 1. Имя файла вводить параметром командной строки –i.

2.5. Реализовать класс Lab3Java, в методе main которого реализовать работу с объектами классов из п. 2.1-2.4:

1. Ввести записи из файла заданного параметром командной строки –i в коллекцию HashSet.

2. Отобразить записи в консоли.

3. Предложить пользователю ввести значение поля 1.

4. Отобразить в консоли результат проверки наличия записи по введенному значению поля 1.

5. Ввести записи из файла заданного параметром командной строки –i в коллекцию LinkedList.

6. Отобразить записи в консоли. Отсортировать по полю 1. Отобразить записи в консоли. Отсортировать по полю 2 в направлении возрастания. Отобразить записи в консоли.

7. Вывести записи в файл, заданный параметром командной строки –o.

8. Ввести записи из файла заданного параметром командной строки –i в коллекцию HashMap.

9. Отобразить записи в консоли.

10. Предложить пользователю ввести значение поля 1.

11. Отобразить в консоли значения остальных полей по введенному значению поля 1.

**3.Текст программы**

Класс Main:

**package** lab3java;

**import** java.io.IOException;

**import** java.util.Collections;

**import** java.util.Scanner;

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

**private** **static** Scanner *scn*;

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

String pathIn = **null**;

String pathOut = **null**;

pathIn = Smartphone.*find*("-i", args);

pathOut = Smartphone.*find*("-o", args);

*scn* = **new** Scanner(System.***in***);

List1st list1 = **new** List1st();

list1.read("E:\\" + pathIn);

System.***out***.println("\n 1st list: \n");

list1.print();

System.***out***.println("Enter phone model");

**if** (list1.contains(*scn*.next())) {

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

} **else** {

System.***out***.println("Doesnt contain");

}

List2nd list2 = **new** List2nd();

list2.read("E:\\" + pathIn);

System.***out***.println("\n 2nd list: \n");

list2.print();

System.***out***.println("\n Sorted by 1st field 2nd list: \n");

Collections.*sort*(list2);

list2.print();

System.***out***.println("\n Sorted by 2nd field 2nd list: \n");

Collections.*sort*(list2, **new** SmartphoneComparator());

list2.print();

list2.write("E:\\" + pathOut);

List3rd list3 = **new** List3rd();

list3.read("E:\\" + pathIn);

System.***out***.println("\n 3rd list: \n");

list3.print();

System.***out***.println("Enter phone model \n");

list3.print(*scn*.next());

*scn*.close();

}

}

Класс Smartphone:

**package** lab3java;

**import** java.util.Scanner;

**public** **class** Smartphone **implements** Comparable<Smartphone>{

String phone\_model;

**int** phone\_screen\_size;

String phone\_screen\_type;

**int** phone\_memory\_capacity;

**public** Smartphone(String phone\_model, **int** phone\_screen\_size, String phone\_screen\_type, **int** phone\_memory\_capacity) {

**this**.phone\_model = phone\_model;

**this**.phone\_screen\_size = phone\_screen\_size;

**this**.phone\_screen\_type = phone\_screen\_type;

**this**.phone\_memory\_capacity = phone\_memory\_capacity;

}

**public** **int** getPhone\_memory\_capacity() {

**return** phone\_memory\_capacity;

}

**public** String getPhone\_model() {

**return** phone\_model;

}

**public** **int** getPhone\_screen\_size() {

**return** phone\_screen\_size;

}

**public** String getPhone\_screen\_type() {

**return** phone\_screen\_type;

}

**public** **static** Smartphone read(Scanner scn) {

**return** **new** Smartphone(scn.next(),scn.nextInt(),scn.next(),scn.nextInt());

}

@Override

**public** String toString() {

**return** "Phone model: " + phone\_model +

"\n Phone screen size: " + phone\_screen\_size +

"\n Phone screen type: " + phone\_screen\_type +

"\n Phone memory capacity: " + phone\_memory\_capacity;

}

**public** **int** compareTo(Smartphone s) {

**return** phone\_model.compareTo(s.getPhone\_model());

}

**public** **static** String find(String target, String[] args) {

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

**if** (args[i].equals(target)) {

**return** args[i + 1];

}

}

**return** **null**;

}

}

Класс SmartphoneComparator:

**package** lab3java;

**import** java.util.Comparator;

**public** **class** SmartphoneComparator **implements** Comparator<Smartphone> {

@Override

**public** **int** compare(Smartphone s1, Smartphone s2) {

**return** s1.getPhone\_screen\_size() - s2.getPhone\_screen\_size();

}

}

Класс List1st:

package lab3java;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.util.HashSet;

import java.util.Scanner;

public class List1st extends HashSet<Smartphone> {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

public void read(String filename) {

this.clear();

try (Scanner scn = new Scanner(new FileInputStream(filename))) {

while (scn.hasNext()) {

add(Smartphone.read(scn));

}

} catch (FileNotFoundException e) {

System.out.println("File not found");

}

}

public void print() {

for (Smartphone s : this) {

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

}

}

public boolean contains(String model) {

for (Smartphone s : this) {

if (s.getPhone\_model().equalsIgnoreCase(model)) {

return true;

}

}

return false;

}

}

Класс List2nd

package lab3java;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.PrintWriter;

import java.util.LinkedList;

import java.util.Scanner;

public class List2nd extends LinkedList<Smartphone> {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

public void read(String filename) {

this.clear();

try (Scanner scn = new Scanner(new FileInputStream(filename))) {

while (scn.hasNext()) {

add(Smartphone.read(scn));

}

} catch (FileNotFoundException e) {

System.out.println("File not found");

}

}

public void print() {

for (Smartphone s : this) {

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

}

}

public void write(String filename) {

try (PrintWriter wrt = new PrintWriter(new FileOutputStream(filename))) {

for (Smartphone s : this) {

wrt.write(s.toString());

wrt.write('\n');

}

} catch (FileNotFoundException e) {

System.out.println("File not found");

}

}

}

Класс List3rd:

package lab3java;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.util.HashMap;

import java.util.Scanner;

public class List3rd extends HashMap<String, Smartphone>{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

public void read(String filename) {

this.clear();

Smartphone s;

try(Scanner scn = new Scanner(new FileInputStream(filename))) {

while(scn.hasNext()) {

s = Smartphone.read(scn);

put(s.getPhone\_model(),s);

}

} catch (FileNotFoundException e) {

System.out.println("File not found");

}

}

public void print() {

for(Smartphone s : this.values()) {

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

}

}

public void print(String author) {

Smartphone s = get(author);

if(s != null) {

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

} else {

System.out.println("There's no smartphone with such model");

}

}

}

**4.Результаты**

Пример №1 работы программы:

1st list:

Phone model: Klenguit

Phone screen size: 170

Phone screen type: JKL

Phone memory capacity: 1300

Phone model: Samsung

Phone screen size: 155

Phone screen type: DMK

Phone memory capacity: 1400

Phone model: Jumbolak

Phone screen size: 144

Phone screen type: AKL

Phone memory capacity: 1450

Enter phone model

Samsung

Contain

2nd list:

Phone model: Samsung

Phone screen size: 155

Phone screen type: DMK

Phone memory capacity: 1400

Phone model: Klenguit

Phone screen size: 170

Phone screen type: JKL

Phone memory capacity: 1300

Phone model: Jumbolak

Phone screen size: 144

Phone screen type: AKL

Phone memory capacity: 1450

Sorted by 1st field 2nd list:

Phone model: Jumbolak

Phone screen size: 144

Phone screen type: AKL

Phone memory capacity: 1450

Phone model: Klenguit

Phone screen size: 170

Phone screen type: JKL

Phone memory capacity: 1300

Phone model: Samsung

Phone screen size: 155

Phone screen type: DMK

Phone memory capacity: 1400

Sorted by 2nd field 2nd list:

Phone model: Jumbolak

Phone screen size: 144

Phone screen type: AKL

Phone memory capacity: 1450

Phone model: Samsung

Phone screen size: 155

Phone screen type: DMK

Phone memory capacity: 1400

Phone model: Klenguit

Phone screen size: 170

Phone screen type: JKL

Phone memory capacity: 1300

3rd list:

Phone model: Klenguit

Phone screen size: 170

Phone screen type: JKL

Phone memory capacity: 1300

Phone model: Samsung

Phone screen size: 155

Phone screen type: DMK

Phone memory capacity: 1400

Phone model: Jumbolak

Phone screen size: 144

Phone screen type: AKL

Phone memory capacity: 1450

Enter phone model

Jumbolak

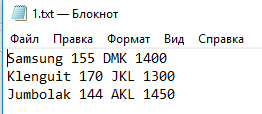
Phone model: Jumbolak

Phone screen size: 144

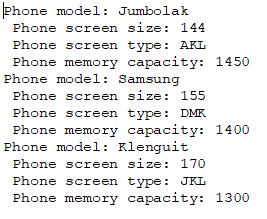
Phone screen type: AKL

Phone memory capacity: 1450

Файл 1.txt:



Файл 111.txt:



**5.Вывод**

Вданной лабораторной работе была изучена организация коллекций объектов на языке Java, также были приобретены практические навыки использования списков, очередей, хеш-таблиц при создании Java программ.