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 №3

по дисциплине: «Платформа Java»

ИССЛЕДОВАНИЕ КОЛЛЕКЦИЙ И ИТЕРАТОРОВ В 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 организацией коллекций объектов на языке Java, приобрести практические навыки использования списков, очередей, хеш-таблиц при создании Java программ.

# ПОСТАНОВКА ЗАДАЧИ

1. Реализовать класс содержащий название альбома, исполнителя, количество треков, и длительность звучания для представления информации о компакт диске;
2. Реализовать коллекцию типа TreeSet заданного 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.
3. Реализовать коллекцию типа LinkedList объектов с возможностью упорядочивания по полю 4 в направлении возрастания. Предусмотреть возможность ввода элементов из файла, вывода на консоль и сохранения в файл. Имена файлов вводить параметрами командной строки –i и –o.
4. Реализовать коллекцию типа TreeMap 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.
5. Реализовать класс Lab3Java, в методе main которого реализовать работу с объектами классов из п.2.1-2.4:

* Ввести записи из файла заданного параметром командной строки –i в коллекцию TreeSet.
* Отобразить записи в консоли.
* Предложить пользователю ввести значение поля 1.
* Отобразить в консоли результат проверки наличия записи по введенному значению поля 1.
* Ввести записи из файла заданного параметром командной строки –i в коллекцию LinkedList.
* Отобразить записи в консоли. Отобразить записи в консоли.
* Отсортировать по полю 4 в направлении возрастания. Отобразить записи в консоли.
* Вывести записи в файл, заданный параметром командной строки –o.
* Ввести записи из файла заданного параметром командной строки –i в коллекцию T2.
* Отобразить записи в консоли.
* Предложить пользователю ввести значение поля 1.
* Отобразить в консоли значения остальных полей по введенному значению поля 1.

# ХОД РАБОТЫ

При запуске программы, для открытия файлов записи и чтения были переданы следующие аргументы командной строки: «-i file\_1.txt –o file\_2.txt».

При этом файл для вывода “file\_2.txt” не существовал, а “file\_1.txt” содержал данные:

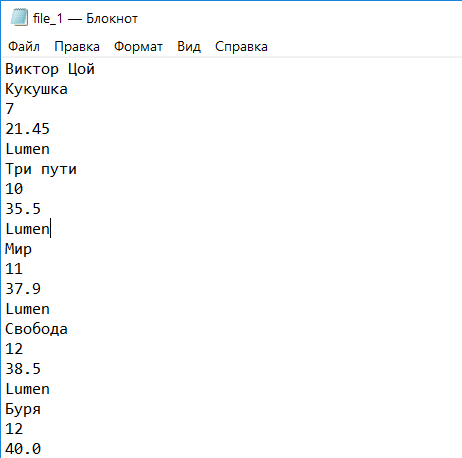


Рисунок 1 – Исходные данные для программы

После запуска данные загружаются в коллекцию TreeSet и выводятся в консоль:

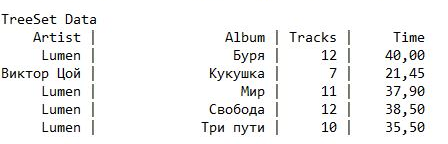


Рисунок 2 – Содержимое коллекции TreeSet

Далее пользователю предлагается ввести название альбома исполнителя, для проверки его существования в коллекции объектов и вывода результата проверки в консоль:

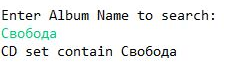


Рисунок 3 – Результат проверки выхождения пользовательских данных в коллекцию TreeSet

Далее данные из файла «file\_1.txt» введены в коллекцию типа LinkedList:

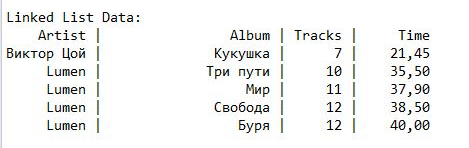


Рисунок 4 – Содержимое коллекции LinkedList

Далее проведена сортировка, с использованием функции Sort и созданного класса-компаратора.

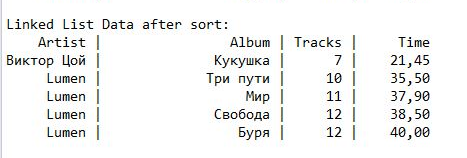


Рисунок 5 – Содержимое коллекции LinkedList после сортировки

Содержимое буфера было сохранено в файл “file\_2.txt”.

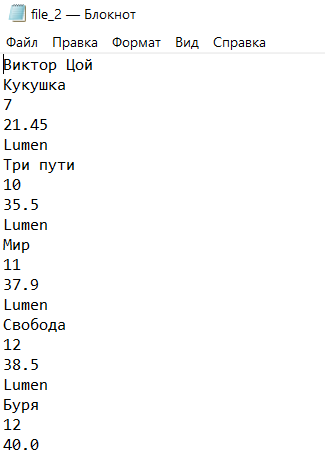


Рисунок 6 – Содержимое файла «file\_2.txt» после сохранения данных ищ коллекции LinkedList

Аналогичным образом данные были загружены в коллекцию TreeMap и выведены в консоль, а так же выполнен поиск и выведена информация о запрашиваемом пользователем объекте из коллекции:



Рисунок 7 – Содержимое коллекции TreeMap и результат поиска заданного объекта в данной коллеции

# ВЫВОД

В ходе выполнения лабораторной работы были исследованы основы организации коллекций объектов на языке Java. Были получены теоретические знания для использования связных список, очередей, хэш-таблиц.

Для закрепления теоретических знания была разработана программа, использующая коллекции стандартных библиотек языка Java и позволяющая ввести данные о компакт диске из файла и вывести их на экран.

По результатам выполнения программы можно сделать вывод, что поставленная цель достигнута.

Приложение А

Текст программы

“Lab3Java.java”

import java.util.Scanner;

public class Lab3Java {

public static void main(String[] args) {

// TODO Auto-generated method stub

try {

FileStreamOpener fileStreams= new FileStreamOpener(args);

TreeSetCollection CDSet = new TreeSetCollection();

CDSet.FillDataFromInputStream(fileStreams.getInputStream());

System.out.println("\nTreeSet Data:");

CDSet.Print();

System.out.println("\nEnter Album Name to search: ");

@SuppressWarnings("resource")

Scanner scanner = new Scanner(System.in);

String foundingData = scanner.next();

String result = CDSet.Contain(foundingData) ? "contain" : "not contain";

System.out.println("CD set " + result + " " + foundingData);

LinkedListCollection CdLList = new LinkedListCollection();

fileStreams.ReopenInputStream();

CdLList.FillDataFromInputStream(fileStreams.getInputStream());

System.out.println("\nLinked List Data:");

CdLList.Print();

CdLList.sort(new MyComp());

System.out.println("\nLinked List Data after sort: ");

CdLList.Print();

CdLList.SaveToFile(fileStreams.getOutputStream());

TreeMapCollection CDMap = new TreeMapCollection();

fileStreams.ReopenInputStream();

CDMap.FillDataFromInputStream(fileStreams.getInputStream());

System.out.println("\nTreeMap Data:");

CDMap.Print();

System.out.println("\nEnter Album Name to search: ");

foundingData = scanner.next();

CDMap.SearchAndPrint(foundingData);

} catch(Exception ex) {

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

}

}

}

“Cd.java”

import java.util.Locale;

import java.util.Scanner;

public class Cd implements Comparable<Cd>{

static final String DefaultName = "None";

private String \_AlbumName;

private String \_Artist;

private int \_trackCount;

private float \_time;

public Cd() {

\_AlbumName = DefaultName;

\_Artist = DefaultName;

\_trackCount = 0;

\_time = 0.0f;

}

public Cd(String AlbumName, String Artist, int trackCount, float time) {

this.setAlbumName(AlbumName);

this.setArtist(Artist);

this.setTrackCount(trackCount);

this.setTime(time);

}

@Override

public int compareTo(Cd \_Cd) {

return this.getAlbumName().compareTo(\_Cd.getAlbumName());

}

public static Cd ReadFromScanner(Scanner scanner) {

scanner.useLocale(Locale.ENGLISH);

String AuthorName = null;

String AlbumName = null;

int TrackCount = 0;

float Time = 0.0f;

if (scanner.hasNext()) {

AuthorName = scanner.nextLine();

}

if (scanner.hasNext()) {

AlbumName = scanner.nextLine();

}

if (scanner.hasNextInt()) {

TrackCount = scanner.nextInt();

}

if (scanner.hasNextFloat()) {

Time = scanner.nextFloat();

}

if (scanner.hasNext()) scanner.nextLine(); //skip '\n'

return new Cd(AlbumName, AuthorName, TrackCount, Time);

}

public String getAlbumName() {

return \_AlbumName;

}

public void setAlbumName(String \_AlbumName) {

this.\_AlbumName = \_AlbumName;

}

public String getArtist() {

return \_Artist;

}

public void setArtist(String \_Artist) {

this.\_Artist = \_Artist;

}

public float getTime() {

return \_time;

}

public void setTime(float \_time) {

this.\_time = \_time;

}

public int getTrackCount() {

return \_trackCount;

}

public void setTrackCount(int \_trackCount) {

this.\_trackCount = \_trackCount;

}

}

“FileStreamOpener.java”

import java.io.\*;

public class FileStreamOpener{

private BufferedReader \_fin = null;

private PrintWriter \_fout = null;

private String InputFilePath = null;

private String OutputFilePath = null;

public FileStreamOpener(String args[]){

try {

for (int i = 0; (i < args.length - 1); ++i) {

switch (args[i]) {

case "-i":{

InputFilePath = args[i + 1];

\_fin = new BufferedReader (new FileReader(InputFilePath));

break;

}

case "-o":{

OutputFilePath = args[i + 1];

\_fout = new PrintWriter(new FileOutputStream(OutputFilePath));

break;

}

}

}

}

catch (FileNotFoundException exOb) {

System.out.println(exOb.getMessage());

System.exit(1);

}

}

public void ReopenInputStream() {

try {

if (\_fin != null) {

\_fin.close();

}

\_fin = new BufferedReader (new FileReader(InputFilePath));

}

catch (IOException exOb) {

System.out.println(exOb.getMessage());

System.exit(1);

}

}

public void ReopenOutputStream() {

try {

if (\_fout != null) {

\_fout.close();

}

\_fout = new PrintWriter(new FileOutputStream(OutputFilePath));

}

catch (IOException exOb) {

System.out.println(exOb.getMessage());

System.exit(1);

}

}

public boolean finIsOpen() {

return (\_fin == null ? false : true);

}

public boolean foutIsOpen() {

return (\_fout == null ? false : true);

}

public BufferedReader getInputStream() {

return \_fin;

}

public PrintWriter getOutputStream() {

return \_fout;

}

@Override

protected void finalize() throws Throwable {

// TODO Auto-generated method stub

if (\_fin != null) {

\_fin.close();

}

if (\_fout != null) {

\_fout.close();

}

super.finalize();

}}

“LinkedListCollection.java”

import java.io.BufferedReader;

import java.io.PrintWriter;

import java.util.InputMismatchException;

import java.util.LinkedList;

import java.util.Scanner;

@SuppressWarnings("serial")

public class LinkedListCollection extends LinkedList<Cd>{

public void FillDataFromInputStream(BufferedReader InputStream) {

Scanner scanner = null;

try {

scanner = new Scanner(InputStream);

while(scanner.hasNext()) {

Cd CdInformation = Cd.ReadFromScanner(scanner);

this.addLast(CdInformation);

}

} catch(InputMismatchException ex){

System.out.println(ex);

} finally {

if (scanner != null) {

scanner.close();

}

}

}

public void Print() {

System.out.printf("%10s | %20s | %6s | %8s\n","Artist","Album", "Tracks", "Time");

for(Cd temp : this) {

System.out.printf("%10s | %20s | %6d | %8.2f\n", temp.getArtist(), temp.getAlbumName(), temp.getTrackCount(), temp.getTime());

}

}

public void SaveToFile(PrintWriter out\_stream) {

for(Cd temp : this) {

out\_stream.println(temp.getArtist());

out\_stream.println(temp.getAlbumName());

out\_stream.println(temp.getTrackCount());

out\_stream.println(temp.getTime());

}

out\_stream.close();

}

}

“TreeMapCollection.java”

import java.io.BufferedReader;

import java.util.\*;

@SuppressWarnings("serial")

public class TreeMapCollection extends TreeMap<String, Cd>{

public void FillDataFromInputStream(BufferedReader InputStream) {

Scanner scanner = null;

try {

scanner = new Scanner(InputStream);

while(scanner.hasNext()) {

Cd CdInformation = Cd.ReadFromScanner(scanner);

this.put(CdInformation.getAlbumName(),CdInformation);

}

} catch(InputMismatchException ex){

System.out.println(ex);

} finally {

if (scanner != null) {

scanner.close();

}

}

}

public void Print() {

System.out.printf("%10s | %20s | %6s | %8s\n","Artist","Album", "Tracks", "Time");

Set<Map.Entry<String,Cd>> tempSet = this.entrySet();

for(Map.Entry<String,Cd> temp : tempSet) {

System.out.printf("%10s | %20s | %6d | %8.2f\n",

temp.getValue().getArtist(), temp.getKey(),

temp.getValue().getTrackCount(), temp.getValue().getTime());

}

}

public boolean Contain(String AlbumName) {

return this.containsKey(AlbumName);

}

public void SearchAndPrint(String AlbumName) {

if (containsKey(AlbumName)) {

Cd temp = this.get(AlbumName);

System.out.printf("%10s | %20s | %6s | %8s\n","Artist","Album", "Tracks", "Time");

System.out.printf("%10s | %20s | %6d | %8.2f \n", temp.getArtist(), temp.getAlbumName(), temp.getTrackCount(), temp.getTime());

} else {

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

}

}

}

“TreeSetCollection.java”

import java.io.BufferedReader;

import java.util.\*;

@SuppressWarnings("serial")

public class TreeSetCollection extends TreeSet<Cd>{

public void FillDataFromInputStream(BufferedReader InputStream) {

Scanner scanner = null;

try { scanner = new Scanner(InputStream);

while(scanner.hasNext()) {

Cd CdInformation = Cd.ReadFromScanner(scanner);

this.add(CdInformation);

}

} catch(InputMismatchException ex){

System.out.println(ex);

} finally {

if (scanner != null) {

scanner.close();

} } }

public void Print() {

System.out.printf("%10s | %20s | %6s | %8s\n","Artist","Album", "Tracks", "Time");

for(Cd temp : this) {

System.out.printf("%10s | %20s | %6d | %8.2f\n", temp.getArtist(), temp.getAlbumName(), temp.getTrackCount(), temp.getTime());

}

}

public boolean Contain(String AlbumName) {

Iterator<Cd> CDiter = this.iterator();

while(CDiter.hasNext()) {

Cd temp = CDiter.next();

if (temp.getAlbumName().equals(AlbumName)) {

return true;

}

}

return false;

}}

“MyComp.java”

import java.util.Comparator;

public class MyComp implements Comparator<Cd> {

@Override

public int compare(Cd o1, Cd o2) {

return Float.compare(o1.getTime(), o2.getTime());

}

}