# ЛАБОРАТОРНАЯ РАБОТА №3

# ИССЛЕДОВАНИЕ КОЛЛЕКЦИЙ И ИТЕРАТОРОВ В ЯЗЫКЕ JAVA

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

Цель работы

Ознакомиться с организацией коллекций объектов на языке Java, приобрести практические навыки использования списков, очередей, хеш-таблиц при создании Java программ.

Задачи

1. В соответствии с вариантом задания (Таблица 1) реализовать класс для представления требуемой информации;
2. Реализовать коллекцию типа Т1 (см. вариант задания) объектов класса, разработанного в п. 1, с возможностью ввода элементов из файла, вывода на консоль, проверки членства по введённому с консоли значению поля 1. Имя файла вводить параметром командной строки -i;
3. Реализовать коллекцию типа LinkedList объектов класса, разработанного в п. 1, с возможностью: упорядочивания по полю 1 (использовать Collections.sort(list)); упорядочивания по полю P (см. вариант задания) в направлении U (использовать Collections.sort(list, myComp), где myComp – экземпляр разработанного класса, реализующего интерфейс Comparator); ввода элементов из файла, вывода на консоль и сохранения в файл. Имена файлов вводить параметрами командной строки -i и -o;
4. Реализовать коллекцию типа Т2 (см. вариант задания) объектов разработанного в п. 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 введённому 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. 1-4:
   1. Ввести записи из файла, заданного параметром командной строки -i в коллекцию T1. Отобразить записи в консоли;
   2. Предложить пользователю ввести значение поля 1. Отобразить в консоли результат проверки наличия записи по введённому значению поля 1;
   3. Ввести записи из файла, заданного параметром командной строки -i, в коллекцию LinkedList;
   4. Отобразить записи в консоли. Отсортировать по полю 1. Отобразить записи в консоли. Отсортировать по полю P в направлении U. Отобразить записи в консоли;
   5. Вывести записи в файл, заданный параметром командной строки -o;
   6. Ввести записи из файла, заданного параметром командной строки -i, в коллекцию T2. Отобразить записи в консоли;
   7. Предложить пользователю ввести значение поля 1. Отобразить в консоли значения остальных полей по введённому значению поля 1;

Вариант задания

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Тип информации | Поле для сортировки (P) | Направление (U) | Тип коллекции (Т1) | Тип коллекции (T2) |
| 13 | C | 1 | Убывание | HashSet | HashMap |

С: Компакт диск (название альбома, исполнитель, количество треков, длительность звучания).

Ход работы

Программа была написана таким образом, чтобы считывание файлов происходило через файл формата .csv. Файл был заполнен рядом значений (Рисунок 1).

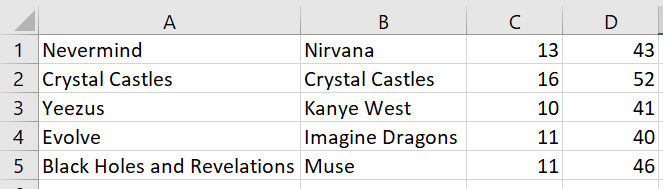


Рисунок 1 – Заполнение начальных значений

Была запущена написанная программа, в качестве параметра файла ввода был указан созданный выше файл. Программа началась с обработки коллекции типа HashSet. Данные были считаны верно, без упорядочивания и сортировки, при вводе названия альбома была выведена информация о нём (Рисунок 2).

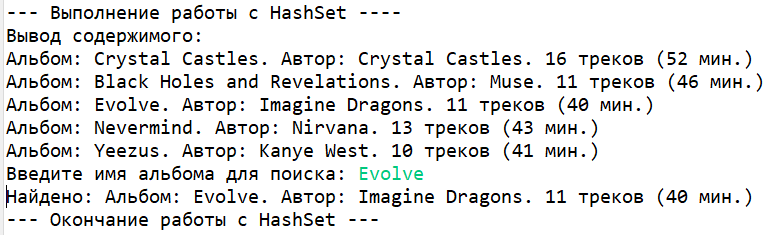


Рисунок 2 – Обработка коллекции HashSet

При работе с коллекцией класса LinkedList данные были считаны в том же порядке, что и в исходном файле (Рисунок 3). Функция сортировки сработала верно.

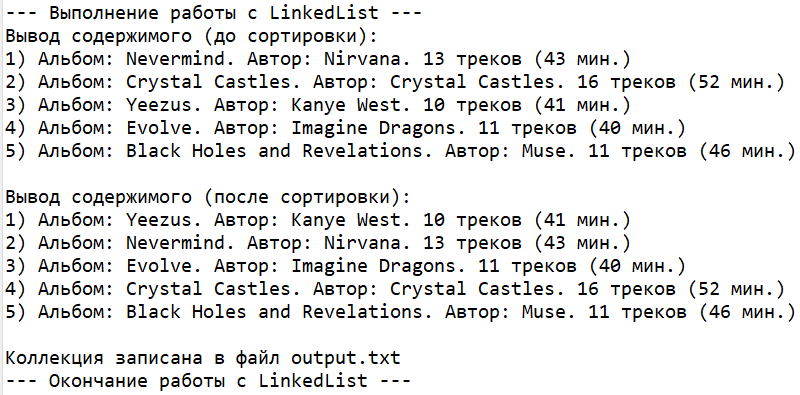


Рисунок 3 – Обработка коллекции LinkedList

По окончании работы с коллекцией отсортированные данные были загружены в файл output.txt (Рисунок 4).

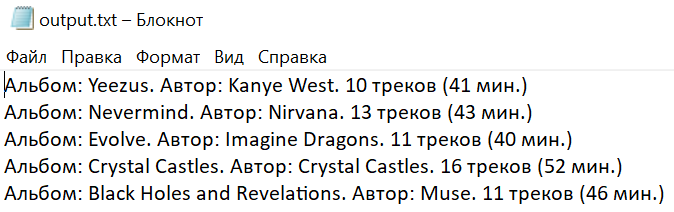


Рисунок 4 – Вывод коллекции в файл

Была обработана коллекция типа HashSet. Данные были считаны верно (ключ – имя альбома, элемент – объект класса Album), без упорядочивания и сортировки, при вводе названия альбома была выведена информация о нём (Рисунок 5).

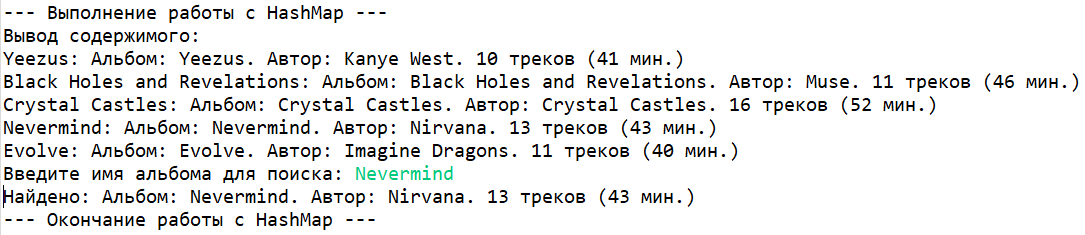


Рисунок 5 – Обработка коллекции HashMap

Также программа была протестирована для исключительных ситуаций (отсутствие параметров командной строки, неверное содержание файлов, ошибка форматов и др.).

Текст программы

Листинг 1 – Lab3Java.java

**import** java.io.File;

**import** java.io.IOException;

**public** **class** Lab3Java

{

**private** **static** File *input* = **null**;

**private** **static** File *output* = **null**;

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

{

**try**

{

*readArguments*(args);

**new** Task2(*input*).execute();

**new** Task3(*input*).execute(*output*);

**new** Task4(*input*).execute();

}

**catch** (IllegalArgumentException e)

{

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

}

**catch** (IOException e)

{

System.***err***.println("Ошибка ввода-вывода");

}

}

// Обработка аргументов командной строки

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

{

**try**

{

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

{

**switch** (args[i])

{

**case** "-i": // параметр файла ввода

*input* = **new** File(args[++i]);

**break**;

**case** "-o": // параметр файла вывода

*output* = **new** File(args[++i]);

**break**;

**default**: // ошибка параметра

**throw** **new** IllegalArgumentException("Ошибка: параметр " + args[0] + " не распознан");

}

}

}

**catch** (IndexOutOfBoundsException e)

{

**throw** **new** IllegalArgumentException("Ошибка: нет значения после параметра");

}

**finally**

{

**if** (*input* == **null**)

**throw** **new** IllegalArgumentException("Ошибка: не введён файл ввода");

**if** (*output* == **null**)

**throw** **new** IllegalArgumentException("Ошибка: не введён файл вывода");

}

}

}

Листинг 2 – Task2.java

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.HashSet;

**import** java.util.Scanner;

**public** **class** Task2

{

**private** HashSet<CD> data;

// Конструктор класса (считывает данные из файла)

**public** Task2(File input) **throws** IOException

{

data = **new** HashSet<CD>();

**try** (BufferedReader reader = **new** BufferedReader(**new** FileReader(input)))

{

String line;

**while** ((line = reader.readLine()) != **null**)

{

String[] params = line.split(";");

CD cd = **new** CD(params[0], params[1], Integer.*parseInt*(params[2]), Integer.*parseInt*(params[3]));

data.add(cd);

}

reader.close();

}

**catch** (NumberFormatException | ArrayIndexOutOfBoundsException e)

{

**throw** **new** IllegalArgumentException("Ошибка считывания файла ввода");

}

}

**public** **void** printAll() // вывод элементов

{

**for** (CD element : data)

{

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

}

}

**public** **void** execute() // выполнение операций с коллекцией

{

System.***out***.println("--- Выполнение работы с HashSet ----");

System.***out***.println("Вывод содержимого:");

printAll();

System.***out***.print("Введите имя альбома для поиска: ");

CD founded = findByAlbum(readKey());

**if** (founded != **null**)

{ System.***out***.println("Найдено: " + founded.toString()); }

**else**

{ System.***out***.println("Ошибка: указанный элемент не найден"); }

System.***out***.println("--- Окончание работы с HashSet ---\n");

}

**public** CD findByAlbum(String album) // поиск по названию альбома

{

CD foundedElement = **null**;

**for** (CD element : data)

{

**if** (element.albumTitle.equals(album))

{

foundedElement = element;

**break**;

}

}

**return** foundedElement;

}

**private** String readKey() // ввод строчного параметра

{

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

String key = scanner.next();

**return** key;

}

}

Листинг 3 – Task3.java

**import** java.io.BufferedReader;

**import** java.io.BufferedWriter;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.util.Collections;

**import** java.util.LinkedList;

**public** **class** Task3

{

**private** LinkedList<CD> data;

// Конструктор класса (считывает данные из файла)

**public** Task3(File input) **throws** IOException

{

data = **new** LinkedList<CD>();

**try** (BufferedReader reader = **new** BufferedReader(**new** FileReader(input)))

{

String line;

**while** ((line = reader.readLine()) != **null**)

{

String[] params = line.split(";");

CD cd = **new** CD(params[0], params[1], Integer.*parseInt*(params[2]), Integer.*parseInt*(params[3]));

data.add(cd);

}

reader.close();

}

**catch** (NumberFormatException | ArrayIndexOutOfBoundsException e)

{

**throw** **new** IllegalArgumentException("Ошибка считывания файла ввода");

}

}

**public** **void** printAll() // вывод элементов

{

**int** i = 0;

**for** (CD element : data)

{

System.***out***.printf("%d) %s\n", ++i, element.toString());

}

}

**public** **void** execute(File file) **throws** IOException

{

System.***out***.println("--- Выполнение работы с LinkedList ---");

System.***out***.println("Вывод содержимого (до сортировки):");

printAll();

System.***out***.println("\nВывод содержимого (после сортировки):");

//Collections.sort(data, Collections.reverseOrder()); // сортировка через Comparable

Collections.*sort*(data, **new** MyComp()); // сортировка через Comparator

printAll();

saveToFile(file);

System.***out***.println("\nКоллекция записана в файл " + file);

System.***out***.println("--- Окончание работы с LinkedList ---\n");

}

// Запись данных из контейнера в файл (текстовый)

**public** **void** saveToFile(File file) **throws** IOException

{

BufferedWriter writer = **new** BufferedWriter(**new** FileWriter(file));

**for** (CD element : data)

{

writer.write(element.toString());

writer.newLine();

}

writer.close();

}

}

Листинг 4 – Task4.java

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.\*;

**public** **class** Task4

{

**private** HashMap<String, CD> data;

// Конструктор класса (считывает данные из файла)

**public** Task4(File input) **throws** IOException

{

data = **new** HashMap<String, CD>();

**try** (BufferedReader reader = **new** BufferedReader(**new** FileReader(input)))

{

String line;

**while** ((line = reader.readLine()) != **null**)

{

String[] params = line.split(";");

String key = params[0];

CD value = **new** CD(params[0], params[1], Integer.*parseInt*(params[2]), Integer.*parseInt*(params[3]));

data.put(key, value);

}

reader.close();

}

**catch** (NumberFormatException | ArrayIndexOutOfBoundsException e)

{

**throw** **new** IllegalArgumentException("Ошибка считывания файла ввода");

}

}

**private** **void** printAll() // вывод содержимого коллекции

{

**for** (Map.Entry<String, CD> entry : data.entrySet())

{

String key = entry.getKey();

CD value = entry.getValue();

System.***out***.println(key + ": " + value.toString());

}

}

**public** **void** execute()

{

System.***out***.println("--- Выполнение работы с HashMap ---");

System.***out***.println("Вывод содержимого:");

printAll();

System.***out***.print("Введите имя альбома для поиска: ");

CD founded = data.get(readKey());

**if** (founded != **null**)

{

System.***out***.println("Найдено: " + founded.toString());

}

**else**

{

System.***out***.println("Ошибка: указанный элемент не найден");

}

System.***out***.println("--- Окончание работы с HashMap ---");

}

**private** String readKey() // ввод строчного параметра

{

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

String key = scanner.next();

**return** key;

}

}

Листинг 5 – MyComp.java

**import** java.util.Comparator;

**class** MyComp **implements** Comparator<CD>

{

**public** **int** compare(CD a, CD b)

{

**return** b.albumTitle.compareTo(a.albumTitle);

}

}

Листинг 6 – CD.java

**public** **class** CD **implements** Comparable<CD>

{

**public** **final** String albumTitle;

**public** **final** String artist;

**public** **final** **int** tracksCount;

**public** **final** **int** duration;

**public** CD(String albumTitle, String artist, **int** tracksCount, **int** duration)

{

**this**.albumTitle = albumTitle;

**this**.artist = artist;

**this**.tracksCount = tracksCount;

**this**.duration = duration;

}

@Override

**public** **int** compareTo(CD obj)

{

**return** albumTitle.compareTo(obj.albumTitle);

}

@Override

**public** String toString()

{

**return** String.*format*("Альбом: %s. Автор: %s. %d треков (%d мин.)", albumTitle, artist, tracksCount, duration);

}

}

Вывод

В ходе работы были изучены принципы организации коллекций объектов на языке Java. Были рассмотрены различные виды коллекций: списки, очереди, множества, отображения, а также встроенные классы языка Java по их реализации.

Результатом работы стала программа на языке Java, реализующая заполнение и обработку коллекций типа множество, список и отображение с помощью встроенных классов.