**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**КАФЕДРА САПР**

**ИДЗ**

**ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ KOTLIN»**

|  |  |  |
| --- | --- | --- |
| Студенты гр. 1301 |  | Чебесова И.Д.  Верхозина А.А. |
| Преподаватель |  | Кулагин М.В. |

Санкт-Петербург

2023

# Консольная программа для поиска в Википедии

# 1.1. Задание

Напишите программу, которая с консоли считывает поисковый запрос пользователя, и выводит результат поиска по Википедии. После выбора нужной статьи программа должна открывать ее в браузере. Программа должна реагировать корректно на любой пользовательский ввод.

Задача разбивается на 5 этапов:

1. Считать введенные пользователем данные
2. Сделать запрос к серверу
3. Распарсить ответ
4. Вывести результат поиска
5. Открыть нужную страницу в браузере

Использовать готовые библиотеки для работы с Википедией нельзя.

# 1.2. Спецификация программы

В данной программе реализованы 4 класса: Main, Get\_response, Parsing и Result:

* В Main происходит ввод пользователем запроса, инициализация url строки и вызов методов других классов;
* В Get\_response метод response\_api устанавливает соединение с url строкой и возвращает данные полученные после подключения в формате String;
* В методе output класса Parsing с помощью библиотеки gson происходит выделение нужных полей исходной строки – title и pageid – и занесение последнего значения в массив для дальнейшего использования;
* В Results происходит открытие браузера с полученной добавлением нужного pageid к строке ссылкой на статью Википедии с помощью класса Desktop.

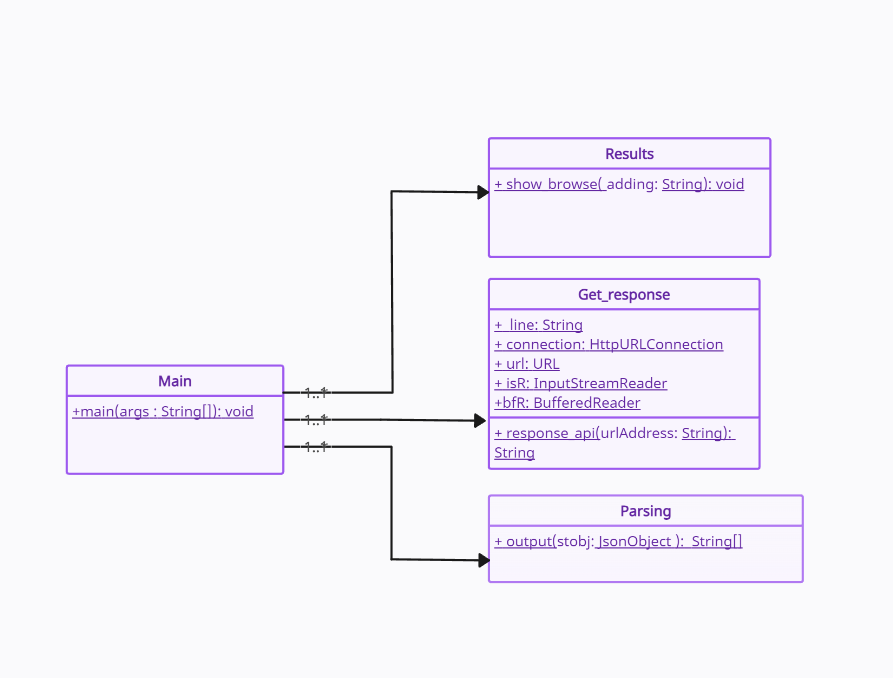
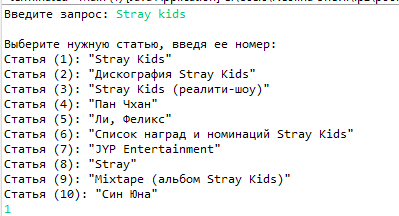
На Рисунке 1.1 отображена общая структура иерархии классов системы.

Рисунок 1.1. Диаграмма классов

# 1.3. Описание интерфейса пользователя программы

По заданию было реализовано консольное приложение, отображающее статью в браузере по выбору пользователя.

После запуска программы в консоль выводится текстовое сообщение, приглашающее пользователя ввести поисковой запрос для Википедии. Запрос может состоять как из одного слова, так и быть строкой. Далее выводятся первые 10 статей из поиска (или меньшее количество при отсутствии 10), из которых пользователь должен выбрать одну, введя ее номер (Рисунок 1.2). При корректном вводе происходит открытие браузера с выбранной статьей Википедии (Рисунок 1.3). В противном случае программа отображает текст, оповещающий о том, что запрос необходимо повторить, после чего заканчивает свою работу (Рисунок 4).

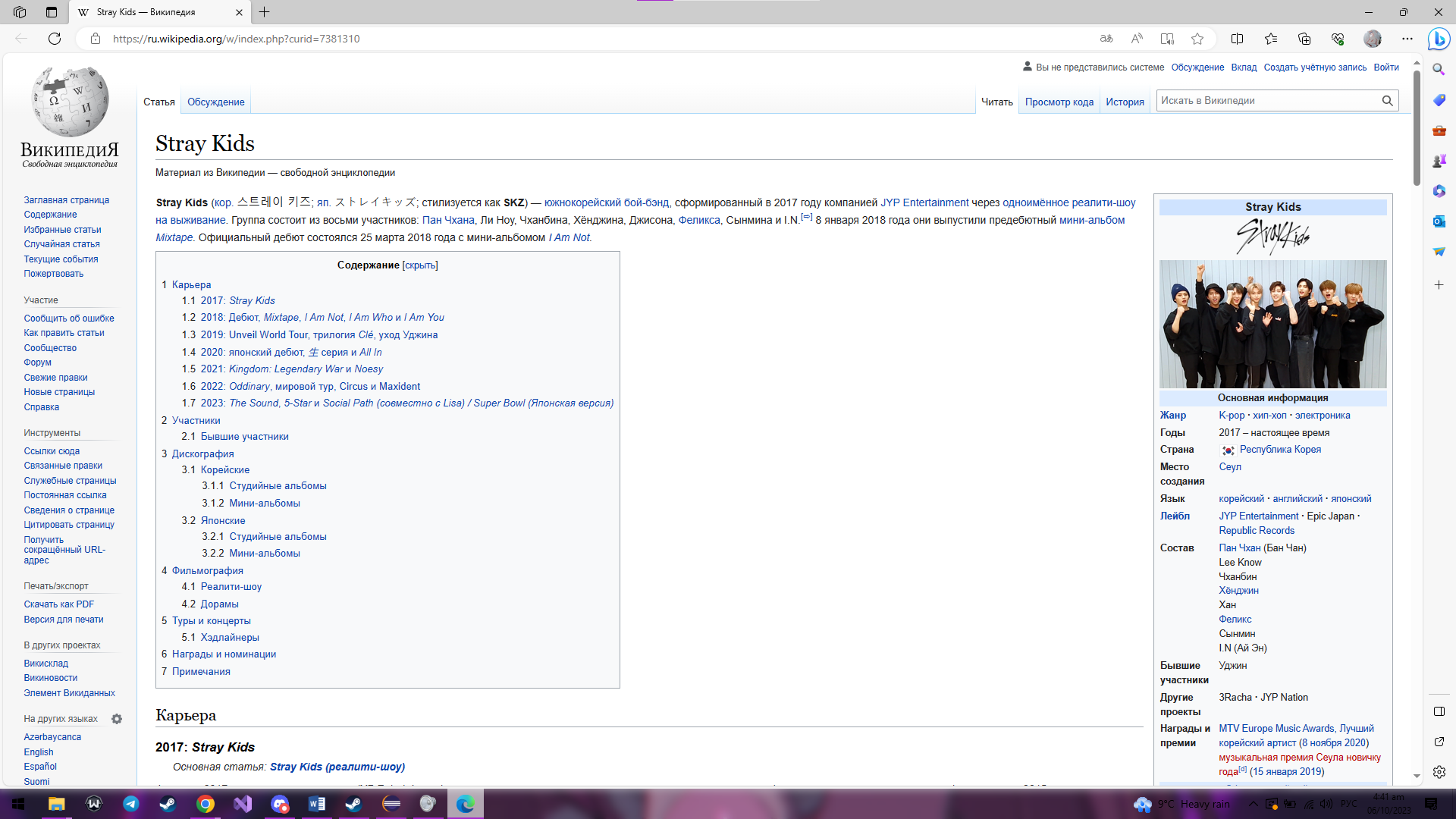
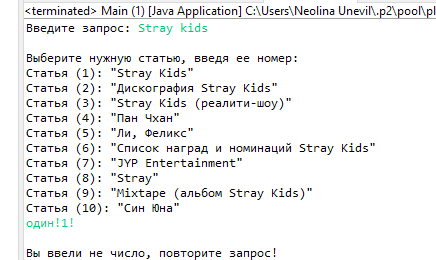
Рисунок 1.2

Рисунок 1.3

Рисунок 1.4

# 1.4. Текст программы

**Файл Main.java:**

**import** java.net.URLEncoder;

**import** java.nio.charset.StandardCharsets;

**import** java.util.InputMismatchException;

**import** java.util.Scanner;

**import** com.google.gson.JsonObject;

**import** com.google.gson.JsonParser;

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

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

{

//ввод запроса

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

**System**.out.print("Bведите запрос: ");

**String** ask =in.nextLine();

**String** urlAddress = "https://ru.wikipedia.org/w/api.php?action=query&list=search&utf8=&format=json&srsearch="+ **URLEncoder**.encode(ask, **StandardCharsets**.UTF\_8);

**String** line="";

line= Get\_response.response\_api(urlAddress);//получение api

//распределение полученной строки по json объектам

JsonObject stobj= JsonParser.parseString(line).getAsJsonObject();

**String** pageid[]=Parsing.output(stobj);

int j=0;//индекс массива pageid

**try**

{

j =in.nextInt();

}

**catch**(**InputMismatchException** ex)

{

**System**.out.print("\nBы ввели не число, повторите запрос!\n");

**return**;

}

//этап отображения выбранной статьи в браузере

**if**(j>0&&j<pageid.length+1)

{

**String** adding = pageid[j-1];

in.close();

Results.show\_browse(adding);

}

**else**

**System**.out.print("\nBаше число выходит за границы заданного диапазона, повторите запрос!\n");

}

}

**Файл Parsing.java:**

**import** com.google.gson.JsonArray;

**import** com.google.gson.JsonObject;

**public** **class** Parsing

{

**static** **String**[] output(JsonObject stobj)

{

//создание объектов json по кодовым словам

JsonObject dataobj = stobj.getAsJsonObject("query");

JsonArray search = dataobj.getAsJsonArray("search");

**String**[] pageid =**new** **String**[search.size()];//строка для хранения pageid

**System**.out.print("\nBыберите нужную статью, введя ее номер: \n");

**for**(int i=0;i<search.size();i++)

{

JsonObject temp = search.get(i).getAsJsonObject();//получение i-того поля массива

**System**.out.printf("Статья (%d): %s\n",i+1, temp.getAsJsonPrimitive("title").toString());

pageid[i]= temp.getAsJsonPrimitive("pageid").toString();

}

**return** pageid;

}

}

**Файл Get\_response.java:**

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.net.HttpURLConnection;

**import** java.net.URL;

**public** **class** Get\_response

{

**static** **String** line;

**static** **HttpURLConnection** connection;

**static** **URL** url;

**static** **InputStreamReader** isR;

**static** **BufferedReader** bfR;

**static** **String** response\_api(**String** urlAddress)

{

**try**

{

url = **new** **URL**(urlAddress);

connection = (**HttpURLConnection**) url.openConnection();//возвращение в connection объекта URLConnection

**if** (**HttpURLConnection**.HTTP\_OK == connection.getResponseCode()) //удалось установить соединение

{

isR = **new** **InputStreamReader**(connection.getInputStream());//декодирование считанных байтов в символы во входном потоке

bfR = **new** **BufferedReader**(isR);//буферизация символов с потока для чтения далее

line = bfR.readLine();

}

**else**

{

**System**.out.printf("Fail %s", connection.getResponseCode());

}

}

**catch** (**IOException** e)

{

e.printStackTrace();

}

**return** line;

}

}

**Файл Results.java:**

**import** java.awt.Desktop;

**import** java.io.IOException;

**import** java.net.URI;

**import** java.net.URISyntaxException;

**import** java.net.URLEncoder;

**import** java.nio.charset.StandardCharsets;

**public** **class** Results

{

**static** void show\_browse(**String** adding)

{

**try**

{ //проверка на то, поддерживается ли класс и возможно ли открытие браузера

**if** (Desktop.isDesktopSupported() && Desktop.getDesktop().isSupported(Desktop.**Action**.BROWSE))

{

//возвращает экземпляр контекста браузера с указанным URI

Desktop.getDesktop().browse(**new** **URI**("https://ru.wikipedia.org/w/index.php?curid="+**URLEncoder**.encode(adding, **StandardCharsets**.UTF\_8)));

}

}

**catch** (**IOException** e)

{

**throw** **new** **RuntimeException**(e);

}

**catch** (**URISyntaxException** e)

{

**throw** **new** **RuntimeException**(e);

}

}

}

# 1.5. Выводы

В ходе данной практики были получены навыки работы с некоторыми функциями языка java для работы со ссылками. Каждый шаг выполнения представлял собой изучение возможностей новых для нас библиотек, классов и методов. Наибольший интерес в программе представляет парсинг строк с использованием библиотеки gson, так как возможно большое количество реализаций данного задания. Еще один новый для нас класс – Desktop. С помощью этого класса было реализовано открытие ссылки в браузере, среди других его функций есть также открытие приложения почты. Подводя итоги, можно сказать, что умения, полученные при написании программы, обязательно пригодятся в дальнейшем, так как они затрагивают возможности работы с интернет пространством.

# Консольная программа для работы с CSV и XML файлами

# 2.1. Задание

Даны 2 файла-справочника городов. Один файл в формате xml, другой в формате csv.

Необходимо разработать консольное приложение для работы с ними.

После запуска приложение ожидает ввода пути до файла-справочника либо команды на завершение работы (какая-то комбинация клавиш).

По команде завершения работы приложение завершает свою работу.

После ввода пути до файла-справочника приложение формирует сводную статистику:

1) Отображает дублирующиеся записи с количеством повторений.

2) Отображает, сколько в каждом городе: 1, 2, 3, 4 и 5 этажных зданий.

3) Показывает время обработки файла.

После вывода статистики приложение снова ожидает ввода пути до файла-справочника либо команды на завершение работы.

В процессе работы приложение падать не должно, выход только по команде на завершение работы.

# 2.2. Спецификация программы

В данной программе реализованы 5 классов: Main, SAXparser, SAXhandler, CSV и HashMapping:

* В Main происходит ввод пользователем пути файла до тех пор пока не будет введена команда для выхода;
* В SAXparser метод parseзапускает обработку файла типа xml с помощью handler’а;
* В методах класса handler’а обычно описаны возможные события во время обработки – StartElement, EndElement, StartDocument, EndElement. В классе SAXhandler данной программы присутствует только метод StartElement, который передает атрибуты каждого тега на дальнейшую обработку;
* В CSV.parse происходит разделение строк из csv файла на атрибуты и передача их на дальнейшую обработку;
* HashMapping – класс, который собирает необходимую статистику с полученных данных – поиск повторов и количество этажей по всем городам. Назван так, потому что реализация сбора статистики сделана с помощью структуры данных HashMap.

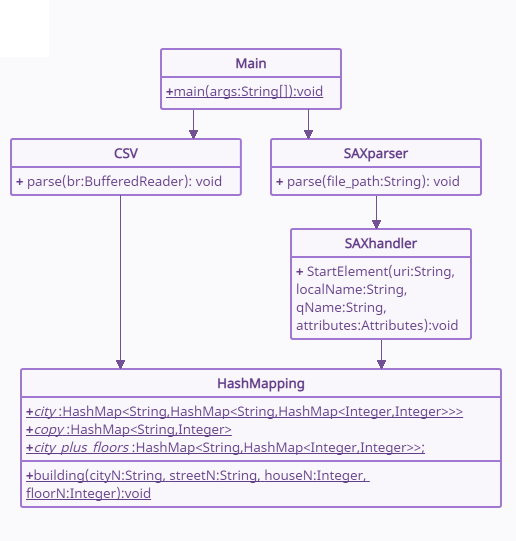
На Рисунке 2.1 отображена общая структура иерархии классов системы.

Рисунок 2.1. Диаграмма классов

# 1.3. Описание интерфейса пользователя программы

По заданию было реализовано консольное приложение. После запуска программы в консоль выводится текстовое сообщение, приглашающее пользователя ввести путь до файла или команду выхода, в данном случае - «0». После проверки файла и его обработки в консоль выводится собранная статистика: повторы (адрес, который повторился, и количество повторов), количество этажей для каждого города (название города и список возможных этажей -1, 2, 3, 4, 5 с количеством совпадений) и время обработки файла (Рис. 2.2 – Обрезанная статистика). Далее пользователю будет снова предложено ввести новый путь или ввести команду для выхода. При нахождении проблемы с файлом (неправильный тип, путь или несовпадающие с кодом программы теги внутри файла) пользователю будет выведено сообщение с просьбой повторить ввод. Примеры реакции программы на ошибки ввода пользователя – Рис. 2.3 – 2.5.

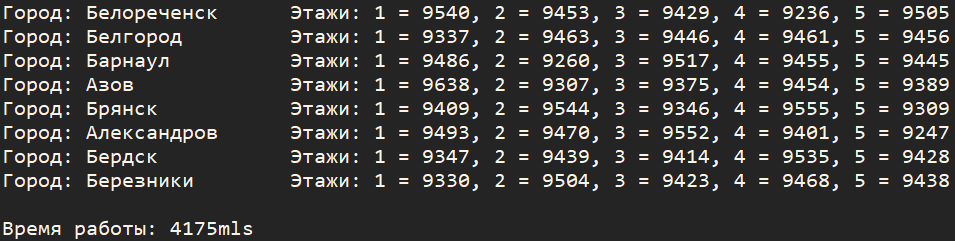
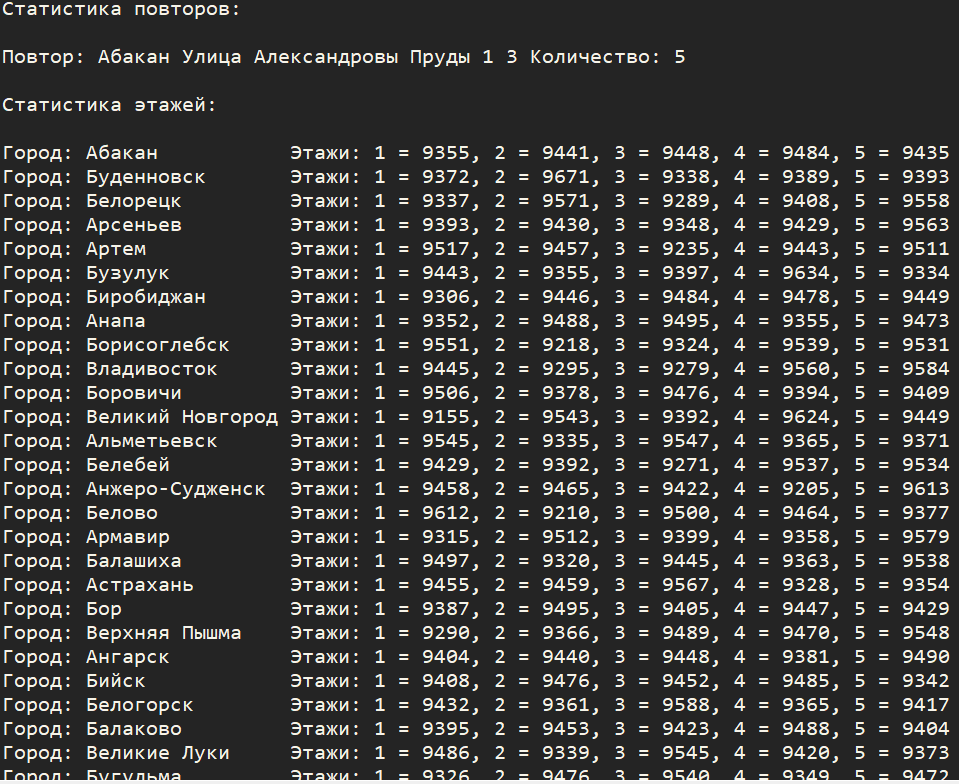
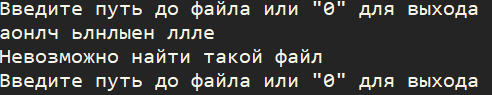
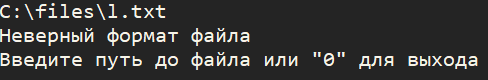


Рисунок 2.2. Обрезанная статистика

Рисунок 2.3. Неверный ввод

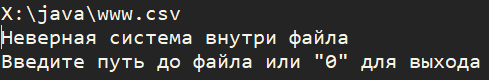
Рисунок 2.4. Неверный тип

Рисунок 2.5. Несоответствие внутреннему формату

# 2.4. Текст программы

**Файл Main.java:**

**package** lab2\_kotlin;

**import** **java.io.BufferedReader**;

**import** **java.io.FileNotFoundException**;

**import** **java.io.FileReader**;

**import** **java.util.HashMap**;

**import** **java.util.Map**;

**import** **java.util.Scanner**;

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

{

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

{

String exit="f";

BufferedReader br=**null**;

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

**while**(exit.equals("0")==**false**)

{

System.out.println("Введите путь до файла или \"0\" для выхода");

exit = in.nextLine();

**if**(exit.equals("0")==**false**)

{

**try**

{

br = **new** BufferedReader(**new** FileReader(exit));

}

**catch** (FileNotFoundException e)

{

System.out.println("Невозможно найти такой файл");

**continue**;

}

**if**(exit.endsWith("xml")==**false**&&exit.endsWith("csv")==**false**)

{

System.out.println("Неверный формат файла");

}

**else**

{

**long** startTime = System.currentTimeMillis();

**if**(exit.endsWith("xml")==**true**)

{

SAXparser parser = **new** SAXparser();

parser.parse(exit);

}

**if**(exit.endsWith("csv")==**true**)

{

**try**

{

CSV.parse(br);

}

**catch** (Exception e)

{

e.printStackTrace();

}

}

**long** endTime = System.currentTimeMillis();

**if**(HashMapping.city\_plus\_floors.isEmpty()==**false**)

{

System.out.println("\nСтатистика повторов:\n");

**for** (Map.Entry<String, Integer> entry : HashMapping.copy.entrySet())

{

System.out.println("Повтор: " + entry.getKey() + " Количество: " + entry.getValue());

}

System.out.println("\nСтатистика этажей:\n");

**for** (Map.Entry<String,HashMap<Integer,Integer>> entry : HashMapping.city\_plus\_floors.entrySet())

{

System.out.printf("Город: %-17s",entry.getKey());

System.out.printf("Этажи: %-10s\n",entry.getValue().toString().replace("{", "").replace("}", "").replace("=", " = "));

}

System.out.println("\nВремя работы: " + (endTime-startTime) +"mls\n");

}

}

}

HashMapping.city.clear();

HashMapping.copy.clear();

HashMapping.city\_plus\_floors.clear();

}

System.out.println("Окончание работы");

in.close();

}

}

**Файл SAXparser.java:**

**package** lab2\_kotlin;

**import** **java.io.File**;

**import** **javax.xml.parsers.SAXParser**;

**import** **javax.xml.parsers.SAXParserFactory**;

**public** **class** **SAXparser**

{

**public** **void** **parse**(String file\_path)

{

SAXParserFactory factory = SAXParserFactory.newInstance();

SAXhandler hndl = **new** SAXhandler();

SAXParser parser = **null**;

**try**

{

parser = factory.newSAXParser();

}

**catch** (Exception e)

{

e.printStackTrace();

}

File file = **new** File(file\_path);

**try**

{

parser.parse(file, hndl);

}

**catch** (Exception e)

{

System.out.println("Неверный путь файла или файл создан неверно");

**return**;

}

}

}

**Файл SAXhandler.java:**

**package** lab2\_kotlin;

**import** **org.xml.sax.Attributes**;

**import** **org.xml.sax.SAXException**;

**import** **org.xml.sax.helpers.\***;

**public** **class** **SAXhandler** **extends** DefaultHandler

{

**@Override**

**public** **void** **startElement**(String uri, String localName, String qName, Attributes attributes) **throws** SAXException

{

**int** attributeLength = attributes.getLength();

String attr[] = **new** String[attributeLength];

**if** ("item".equals(qName))

{

**for** (**int** i = **0**; i < attributeLength; i++)

{

attr[i] = attributes.getValue(i);

}

**try**

{

HashMapping.building(attr[**0**],attr[**1**],Integer.parseInt(attr[**2**]), Integer.parseInt(attr[**3**]));

}

**catch**(Exception e)

{

System.out.println("Неверная система внутри файла");

**return**;

}

}

**else** **if**("item".equals(qName)==**false**&&"root".equals(qName)==**false**)

{

System.out.println("Неверная система внутри файла");

**return**;

}

}

}

**Файл CSV.java:**

**package** lab2\_kotlin;

**import** **java.io.BufferedReader**;

**import** **java.io.IOException**;

**public** **class** **CSV**

{

**public** **static** **int** i=**0**;

**public** **static** **void** **parse**(BufferedReader br) **throws** NumberFormatException, IOException

{

String line=**null**;

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

{

**if**(i!=**0**)

{

line = line.replaceAll("\"", "");

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

**try**

{

HashMapping.building(attr[**0**],attr[**1**],Integer.parseInt(attr[**2**]), Integer.parseInt(attr[**3**]));

}

**catch**(Exception e)

{

System.out.println("Неверная система внутри файла");

**return**;

}

}

**else**

i++;

}

}

}

**Файл HashMapping.java:**

**package** lab2\_kotlin;

**import** **java.util.HashMap**;

**public** **class** **HashMapping**

{

**static** HashMap<String,HashMap<String,HashMap<Integer,Integer>>> city = **new** HashMap<String,HashMap<String,HashMap<Integer,Integer>>>();

**static** HashMap<String, Integer> copy = **new** HashMap<String,Integer>();

**static** HashMap<String,HashMap<Integer,Integer>> city\_plus\_floors = **new** HashMap<String,HashMap<Integer,Integer>>();

**public** **static** **void** **building**(String cityN, String streetN, Integer houseN, Integer floorN)

{

HashMap<String,HashMap<Integer,Integer>> street = **new** HashMap<String,HashMap<Integer,Integer>>();

HashMap<Integer,Integer> house = **new** HashMap<Integer,Integer>();

HashMap<Integer,Integer> each\_floor = **new** HashMap<Integer,Integer>();

**if**(city.isEmpty()==**false**&&city.get(cityN)!=**null**)//city exists:)

{

**if**(city.get(cityN).get(streetN)==**null**)//no street at the moment:(

{

street = city.get(cityN);

house.put(houseN, floorN);

street.put(streetN, house);

city.put(cityN, street);

}

**else**//this street exists:)

{

**if**(city.get(cityN).get(streetN).get(houseN)==floorN) //copy:(

{

**if**(copy.isEmpty()==**true**||copy.get(cityN+' '+streetN+' '+houseN+' '+floorN)==**null**)//first copy

{

copy.put(cityN+' '+streetN+' '+houseN+' '+floorN, **2**);

}

**else**//number'th copy

{

Integer repeat =copy.get(cityN+' '+streetN+' '+houseN+' '+floorN)+**1**;

copy.put(cityN+' '+streetN+' '+houseN+' '+floorN, repeat);

}

}

**else**//no copy:)

{

street = city.get(cityN);

house = street.get(streetN);

house.put(houseN, floorN);

street.put(streetN, house);

city.put(cityN, street);

}

}

}

**else**//city doesn't exist:(

{

house.put(houseN, floorN);

street.put(streetN, house);

city.put(cityN, street);

}

**if**(copy.get(cityN+' '+streetN+' '+houseN+' '+floorN)==**null**)//this counts if it's not a copy:)

{

**if**(city\_plus\_floors.isEmpty()==**false**&&city\_plus\_floors.get(cityN)!=**null**)//not first counter for city

{

Integer fl=city\_plus\_floors.get(cityN).get(floorN);

each\_floor = city\_plus\_floors.get(cityN);

fl++;

each\_floor.put(floorN, fl);

city\_plus\_floors.put(cityN,each\_floor);

}

**else**

{

**for**(**int** j =**1**;j<=**5**;j++)

{

each\_floor.put(j, **0**);

}

each\_floor.put(floorN, **1**);

city\_plus\_floors.put(cityN, each\_floor);

}

}

}

}

# 2.5. Выводы

В ходе данной практики были получены навыки работы с файлами типов csv и xml. Данные файлы необходимы для обмена информацией между различными приложениями и системами, из-за их многофункциональности и распространённости необходимо уметь работать с ними. В созданной программе работа с файлом csv производилась посредством строчных функций, а с файлом xml – с помощью SAX парсера (Simple API for XML). Данный парсер проходит по документу линейно сверху вниз (не читает его весь сразу), анализирует события в документе и уведомляет программу об их происхождении, после чего производятся нужные действия для их обработки. SAX был выбран из-за принципа своей работы, дающего ему преимущество в скорости при работе с большими объёмами информации.