МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
КАФЕДРА САПР

Практическая работа №2

по дисциплине

«Объектно-ориентированное программирование»

на тему: «Консольная программа для поиска в Википедии»

Студенты гр. 4351; 4353 _____ **Петухова В.С.**
Сизых П.В.

Преподаватель _____ **Кулагин М.В.**

Санкт-Петербург
2025

1. Задание

Даны 2 файла-справочника городов: address.xml, address.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.

В работе будет использован hashmap для анализа данных.

2. Спецификация программы

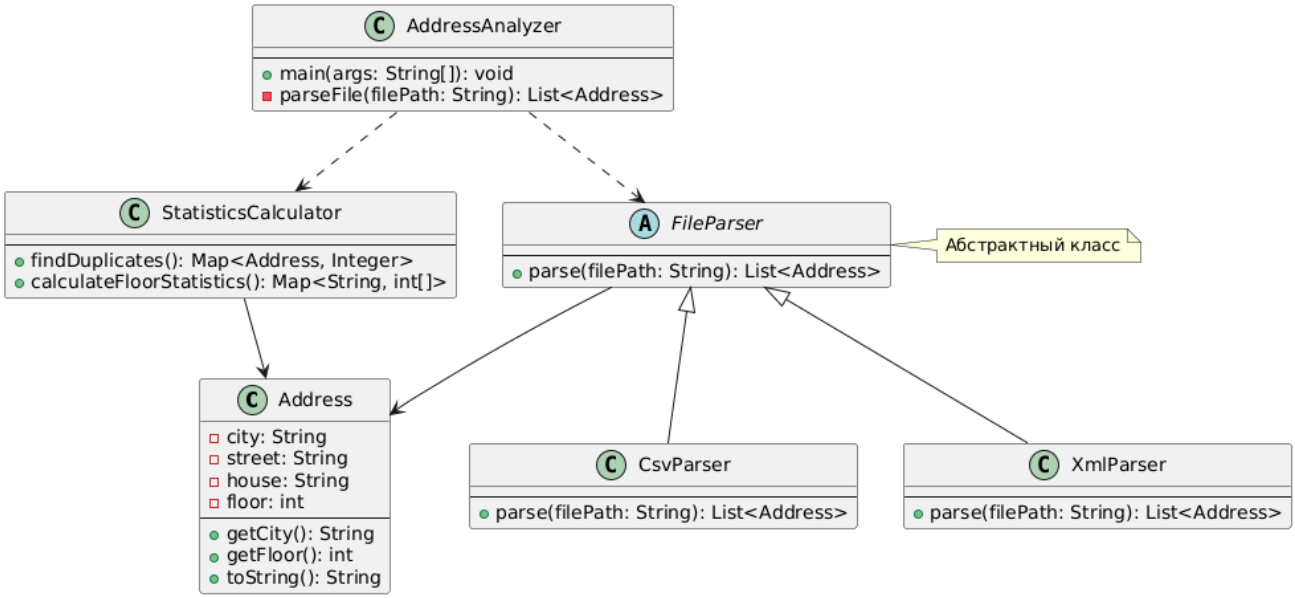

Рисунок 1 – Диаграмма классов

2. Описание интерфейса пользователя программы

Ниже на рисунках 2-7 приведены примеры работы программы.

```
=== Анализатор адресов ===
Введите путь к файлу (CSV или XML)
Для выхода нажмите Ctrl+D или введите 'exit'

> address.csv

=== РЕЗУЛЬТАТЫ СТАТИСТИКИ ===

--- ДУБЛИРУЮЩИЕСЯ ЗАПИСИ ---
Город: Абакан, Улица: Улица Александровы Пруды, Дом: 1, Этажей: 3 - повторений: 5

--- СТАТИСТИКА ПО ЭТАЖАМ ---
Абакан: 1-этажных: 7278, 2-этажных: 7299, 3-этажных: 7336, 4-этажных: 7323, 5-этажных: 7261
Буденновск: 1-этажных: 7238, 2-этажных: 7491, 3-этажных: 7185, 4-этажных: 7285, 5-этажных: 7294
Белорецк: 1-этажных: 7248, 2-этажных: 7402, 3-этажных: 7170, 4-этажных: 7250, 5-этажных: 7423
Арсеньев: 1-этажных: 7257, 2-этажных: 7338, 3-этажных: 7185, 4-этажных: 7338, 5-этажных: 7375
Артем: 1-этажных: 7390, 2-этажных: 7324, 3-этажных: 7108, 4-этажных: 7313, 5-этажных: 7358
Бузулук: 1-этажных: 7277, 2-этажных: 7212, 3-этажных: 7307, 4-этажных: 7429, 5-этажных: 7268
Биробиджан: 1-этажных: 7150, 2-этажных: 7318, 3-этажных: 7405, 4-этажных: 7330, 5-этажных: 7290
Анапа: 1-этажных: 7283, 2-этажных: 7372, 3-этажных: 7270, 4-этажных: 7234, 5-этажных: 7334
Борисоглебск: 1-этажных: 7416, 2-этажных: 7134, 3-этажных: 7182, 4-этажных: 7397, 5-этажных: 7364
Владивосток: 1-этажных: 7383, 2-этажных: 7101, 3-этажных: 7144, 4-этажных: 7457, 5-этажных: 7408
Боровичи: 1-этажных: 7355, 2-этажных: 7260, 3-этажных: 7325, 4-этажных: 7302, 5-этажных: 7251
Великий Новгород: 1-этажных: 7059, 2-этажных: 7415, 3-этажных: 7244, 4-этажных: 7438, 5-этажных: 7337
Альметьевск: 1-этажных: 7385, 2-этажных: 7271, 3-этажных: 7360, 4-этажных: 7305, 5-этажных: 7172
```

Рисунок 2 – Пример 1 работы программы

```
Белебей: 1-этажных: 7311, 2-этажных: 7265, 3-этажных: 7141, 4-этажных: 7373, 5-этажных: 7403
Анжеро-Судженск: 1-этажных: 7369, 2-этажных: 7296, 3-этажных: 7252, 4-этажных: 7170, 5-этажных: 7406
Белово: 1-этажных: 7415, 2-этажных: 7141, 3-этажных: 7398, 4-этажных: 7347, 5-этажных: 7192
Армавир: 1-этажных: 7245, 2-этажных: 7404, 3-этажных: 7203, 4-этажных: 7234, 5-этажных: 7407
Балашиха: 1-этажных: 7319, 2-этажных: 7253, 3-этажных: 7318, 4-этажных: 7233, 5-этажных: 7370
Астрахань: 1-этажных: 7325, 2-этажных: 7267, 3-этажных: 7367, 4-этажных: 7269, 5-этажных: 7265
Бор: 1-этажных: 7256, 2-этажных: 7382, 3-этажных: 7288, 4-этажных: 7271, 5-этажных: 7296
Верхняя Пышма: 1-этажных: 7200, 2-этажных: 7279, 3-этажных: 7285, 4-этажных: 7295, 5-этажных: 7434
Ангарск: 1-этажных: 7334, 2-этажных: 7330, 3-этажных: 7268, 4-этажных: 7193, 5-этажных: 7368
Бийск: 1-этажных: 7322, 2-этажных: 7343, 3-этажных: 7288, 4-этажных: 7274, 5-этажных: 7266
Белогорск: 1-этажных: 7271, 2-этажных: 7298, 3-этажных: 7423, 4-этажных: 7202, 5-этажных: 7299
Балаково: 1-этажных: 7230, 2-этажных: 7318, 3-этажных: 7332, 4-этажных: 7339, 5-этажных: 7274
Великие Луки: 1-этажных: 7318, 2-этажных: 7276, 3-этажных: 7381, 4-этажных: 7242, 5-этажных: 7276
Бугульма: 1-этажных: 7164, 2-этажных: 7347, 3-этажных: 7392, 4-этажных: 7275, 5-этажных: 7315
Буйнакск: 1-этажных: 7361, 2-этажных: 7357, 3-этажных: 7190, 4-этажных: 7319, 5-этажных: 7266
Братск: 1-этажных: 7260, 2-этажных: 7410, 3-этажных: 7229, 4-этажных: 7289, 5-этажных: 7305
Батайск: 1-этажных: 7207, 2-этажных: 7255, 3-этажных: 7406, 4-этажных: 7251, 5-этажных: 7374
Балашов: 1-этажных: 7340, 2-этажных: 7420, 3-этажных: 7262, 4-этажных: 7185, 5-этажных: 7286
Архангельск: 1-этажных: 7289, 2-этажных: 7409, 3-этажных: 7179, 4-этажных: 7466, 5-этажных: 7150
Алексин: 1-этажных: 7319, 2-этажных: 7227, 3-этажных: 7382, 4-этажных: 7276, 5-этажных: 7289
Балахна: 1-этажных: 7364, 2-этажных: 7303, 3-этажных: 7281, 4-этажных: 7173, 5-этажных: 7372
Апатиты: 1-этажных: 7235, 2-этажных: 7183, 3-этажных: 7345, 4-этажных: 7358, 5-этажных: 7372
Видное: 1-этажных: 7378, 2-этажных: 7238, 3-этажных: 7320, 4-этажных: 7249, 5-этажных: 7308
Арзамас: 1-этажных: 7216, 2-этажных: 7403, 3-этажных: 7286, 4-этажных: 7348, 5-этажных: 7240
```

Рисунок 3 – Пример 1 работы программы

```
Березовский: 1-этажных: 7312, 2-этажных: 7475, 3-этажных: 7269, 4-этажных: 7267, 5-этажных: 7170
Ачинск: 1-этажных: 7518, 2-этажных: 7094, 3-этажных: 7292, 4-этажных: 7383, 5-этажных: 7206
Благовещенск: 1-этажных: 7240, 2-этажных: 7292, 3-этажных: 7380, 4-этажных: 7419, 5-этажных: 7162
Асбест: 1-этажных: 7187, 2-этажных: 7305, 3-этажных: 7307, 4-этажных: 7318, 5-этажных: 7376
Белореченск: 1-этажных: 7454, 2-этажных: 7301, 3-этажных: 7262, 4-этажных: 7116, 5-этажных: 7360
Белгород: 1-этажных: 7269, 2-этажных: 7286, 3-этажных: 7332, 4-этажных: 7279, 5-этажных: 7327
Барнаул: 1-этажных: 7262, 2-этажных: 7193, 3-этажных: 7418, 4-этажных: 7334, 5-этажных: 7286
Азов: 1-этажных: 7500, 2-этажных: 7162, 3-этажных: 7253, 4-этажных: 7319, 5-этажных: 7259
Брянск: 1-этажных: 7306, 2-этажных: 7361, 3-этажных: 7184, 4-этажных: 7449, 5-этажных: 7193
Александров: 1-этажных: 7340, 2-этажных: 7363, 3-этажных: 7361, 4-этажных: 7240, 5-этажных: 7189
Бердск: 1-этажных: 7221, 2-этажных: 7267, 3-этажных: 7363, 4-этажных: 7335, 5-этажных: 7307
Березники: 1-этажных: 7244, 2-этажных: 7334, 3-этажных: 7282, 4-этажных: 7369, 5-этажных: 7264

Время обработки: 3384 мс
```

Рисунок 4 – Пример 1 работы программы

```
=== РЕЗУЛЬТАТЫ СТАТИСТИКИ ===

--- ДУБЛИРУЮЩИЕСЯ ЗАПИСИ ---
Город: Абакан, Улица: Улица Александровы Пруды, Дом: 1, Этажей: 3 - повторений: 5

--- СТАТИСТИКА ПО ЭТАЖАМ ---
Абакан: 1-этажных: 9355, 2-этажных: 9441, 3-этажных: 9452, 4-этажных: 9484, 5-этажных: 9435
Буденновск: 1-этажных: 9372, 2-этажных: 9671, 3-этажных: 9338, 4-этажных: 9389, 5-этажных: 9393
Белорецк: 1-этажных: 9337, 2-этажных: 9571, 3-этажных: 9289, 4-этажных: 9408, 5-этажных: 9558
Арсеньев: 1-этажных: 9393, 2-этажных: 9430, 3-этажных: 9348, 4-этажных: 9429, 5-этажных: 9563
Артем: 1-этажных: 9517, 2-этажных: 9457, 3-этажных: 9235, 4-этажных: 9443, 5-этажных: 9511
Бузулук: 1-этажных: 9443, 2-этажных: 9355, 3-этажных: 9397, 4-этажных: 9634, 5-этажных: 9334
Биробиджан: 1-этажных: 9306, 2-этажных: 9446, 3-этажных: 9484, 4-этажных: 9478, 5-этажных: 9449
Анапа: 1-этажных: 9352, 2-этажных: 9488, 3-этажных: 9495, 4-этажных: 9355, 5-этажных: 9473
Борисоглебск: 1-этажных: 9551, 2-этажных: 9218, 3-этажных: 9324, 4-этажных: 9539, 5-этажных: 9531
Владивосток: 1-этажных: 9445, 2-этажных: 9295, 3-этажных: 9279, 4-этажных: 9560, 5-этажных: 9584
Боровичи: 1-этажных: 9506, 2-этажных: 9378, 3-этажных: 9476, 4-этажных: 9394, 5-этажных: 9409
Великий Новгород: 1-этажных: 9155, 2-этажных: 9543, 3-этажных: 9392, 4-этажных: 9624, 5-этажных: 9449
Альметьевск: 1-этажных: 9545, 2-этажных: 9335, 3-этажных: 9547, 4-этажных: 9365, 5-этажных: 9371
Белебей: 1-этажных: 9429, 2-этажных: 9392, 3-этажных: 9271, 4-этажных: 9537, 5-этажных: 9534
Анжеро-Судженск: 1-этажных: 9458, 2-этажных: 9465, 3-этажных: 9422, 4-этажных: 9205, 5-этажных: 9613
Белово: 1-этажных: 9612, 2-этажных: 9210, 3-этажных: 9500, 4-этажных: 9464, 5-этажных: 9377
Армавир: 1-этажных: 9315, 2-этажных: 9512, 3-этажных: 9399, 4-этажных: 9358, 5-этажных: 9579
Балашиха: 1-этажных: 9497, 2-этажных: 9320, 3-этажных: 9445, 4-этажных: 9363, 5-этажных: 9538
Астрахань: 1-этажных: 9455, 2-этажных: 9459, 3-этажных: 9567, 4-этажных: 9328, 5-этажных: 9354
Бор: 1-этажных: 9387, 2-этажных: 9495, 3-этажных: 9405, 4-этажных: 9447, 5-этажных: 9429
```

Рисунок 5 – Пример 2 работы программы

```
Верхняя Пышма: 1-этажных: 9290, 2-этажных: 9366, 3-этажных: 9489, 4-этажных: 9470, 5-этажных: 9548
Ангарск: 1-этажных: 9404, 2-этажных: 9440, 3-этажных: 9448, 4-этажных: 9381, 5-этажных: 9490
Бийск: 1-этажных: 9408, 2-этажных: 9476, 3-этажных: 9452, 4-этажных: 9485, 5-этажных: 9342
Белогорск: 1-этажных: 9432, 2-этажных: 9361, 3-этажных: 9588, 4-этажных: 9365, 5-этажных: 9417
Балаково: 1-этажных: 9395, 2-этажных: 9453, 3-этажных: 9423, 4-этажных: 9488, 5-этажных: 9404
Великие Луки: 1-этажных: 9486, 2-этажных: 9339, 3-этажных: 9545, 4-этажных: 9420, 5-этажных: 9373
Бугульма: 1-этажных: 9326, 2-этажных: 9476, 3-этажных: 9540, 4-этажных: 9349, 5-этажных: 9472
Буйнакск: 1-этажных: 9481, 2-этажных: 9529, 3-этажных: 9310, 4-этажных: 9452, 5-этажных: 9391
Братск: 1-этажных: 9418, 2-этажных: 9558, 3-этажных: 9354, 4-этажных: 9411, 5-этажных: 9422
Батайск: 1-этажных: 9377, 2-этажных: 9382, 3-этажных: 9493, 4-этажных: 9384, 5-этажных: 9527
Балашов: 1-этажных: 9469, 2-этажных: 9538, 3-этажных: 9404, 4-этажных: 9333, 5-этажных: 9419
Архангельск: 1-этажных: 9404, 2-этажных: 9501, 3-этажных: 9250, 4-этажных: 9645, 5-этажных: 9363
Алексин: 1-этажных: 9416, 2-этажных: 9439, 3-этажных: 9488, 4-этажных: 9418, 5-этажных: 9402
Балахна: 1-этажных: 9438, 2-этажных: 9443, 3-этажных: 9405, 4-этажных: 9334, 5-этажных: 9543
Апатиты: 1-этажных: 9334, 2-этажных: 9338, 3-этажных: 9451, 4-этажных: 9560, 5-этажных: 9480
Видное: 1-этажных: 9552, 2-этажных: 9339, 3-этажных: 9451, 4-этажных: 9349, 5-этажных: 9472
Арзамас: 1-этажных: 9368, 2-этажных: 9523, 3-этажных: 9403, 4-этажных: 9421, 5-этажных: 9448
Березовский: 1-этажных: 9398, 2-этажных: 9642, 3-этажных: 9404, 4-этажных: 9419, 5-этажных: 9300
Ачинск: 1-этажных: 9691, 2-этажных: 9220, 3-этажных: 9472, 4-этажных: 9504, 5-этажных: 9276
Благовещенск: 1-этажных: 9381, 2-этажных: 9403, 3-этажных: 9526, 4-этажных: 9535, 5-этажных: 9318
Асбест: 1-этажных: 9343, 2-этажных: 9400, 3-этажных: 9417, 4-этажных: 9456, 5-этажных: 9547
Белореченск: 1-этажных: 9540, 2-этажных: 9453, 3-этажных: 9429, 4-этажных: 9236, 5-этажных: 9505
```

Рисунок 6 – Пример 2 работы программы

```
Белгород: 1-этажных: 9337, 2-этажных: 9463, 3-этажных: 9446, 4-этажных: 9461, 5-этажных: 9456
Барнаул: 1-этажных: 9486, 2-этажных: 9260, 3-этажных: 9517, 4-этажных: 9455, 5-этажных: 9445
Азов: 1-этажных: 9638, 2-этажных: 9307, 3-этажных: 9375, 4-этажных: 9454, 5-этажных: 9389
Брянск: 1-этажных: 9409, 2-этажных: 9544, 3-этажных: 9346, 4-этажных: 9555, 5-этажных: 9309
Александров: 1-этажных: 9493, 2-этажных: 9470, 3-этажных: 9552, 4-этажных: 9401, 5-этажных: 9247
Бердск: 1-этажных: 9347, 2-этажных: 9439, 3-этажных: 9414, 4-этажных: 9535, 5-этажных: 9428
Березники: 1-этажных: 9330, 2-этажных: 9504, 3-этажных: 9423, 4-этажных: 9468, 5-этажных: 9438

Время обработки: 7418 мс
```

Рисунок 7 – Пример 2 работы программы

Ниже на рисунках 8-9 приведены примеры команд на завершение программы (сочетание клавиш Ctrl+D/ввод «exit»).

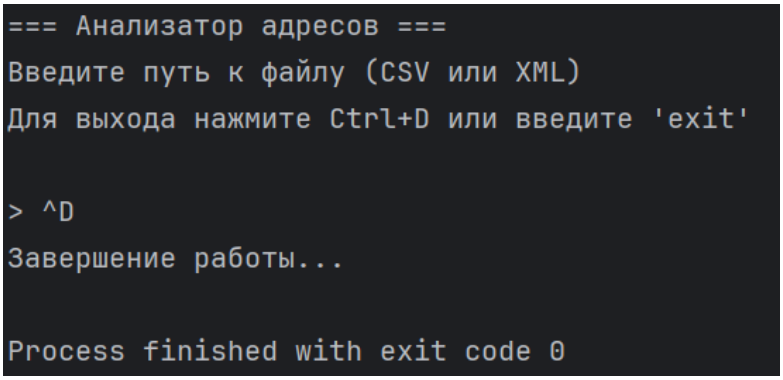

Рисунок 8 – Пример команды 1 (сочетание клавиш) на завершение работы

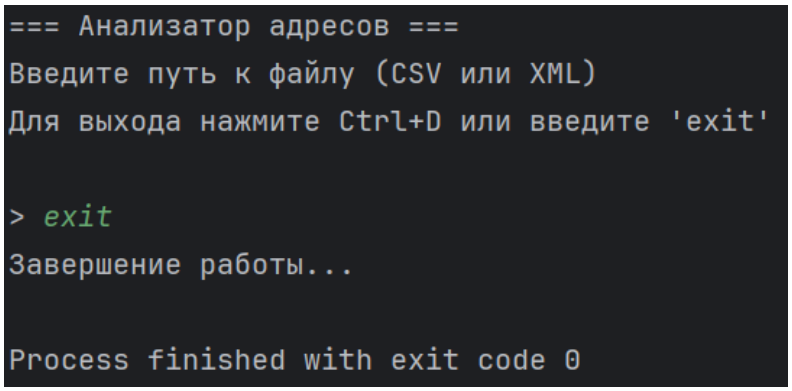

Рисунок 9 – Пример команды 2(ввод ‘exit’) на завершение работы

3. Описание интерфейса пользователя программы.

O1: "=== Анализатор адресов ==="

O2: "Введите путь к файлу (CSV или XML)"

O3: "Для выхода нажмите Ctrl+D или введите ‘exit’ "
" >"

I4: Ввод пути к файлу.

O5: Вывод результатов поиска:

"=== РЕЗУЛЬТАТЫ СТАТИСТИКИ ==="

"--- ДУБЛИРУЮЩИЕСЯ ЗАПИСИ --- "

"Город: … , Улица: … , Дом: … , Этажей: … - повторений: …"

"--- СТАТИСТИКА ПО ЭТАЖАМ ---"

"Название города 1: 1-этажных: …, 2-этажных: …, 3-этажных: …, 4-этажных: … , 5-этажных: … "

"Название города 2: 1-этажных: …, 2-этажных: …, 3-этажных: …, 4-этажных: … , 5-этажных: … "

…

O6: "Время обработки: … мс"

" >"

I7: Ввод пути к файлу/выход из программы

I8: Нажатие Ctrl+D или ввод ‘exit’

O9: "Завершение работы …"

Альтернативные сценарии:

Сценарий А (пустой запрос):

O3: ">"

I4: ""

O5: "Пустой запрос. Введите путь к файлу (CSV или XML) или ‘exit’ для выхода"

Сценарий Б (выход по комбинации клавиш):

O3: ">"

I4: Нажатие Ctrl+D

O5: "Завершение работы …"

Сценарий В (выход по вводу ‘exit’):

O3: ">"

I4: ‘exit’

O5: "Завершение работы …"

Сценарий Г (файл не найден или ошибка чтения):

O2: "Введите путь к файлу (CSV или XML)"

I4: adres.csv

O5: "Сообщение об ошибке (файл не найден/ошибка чтения файла)"

"Файл не содержит данных или произошла ошибка при чтении"

Сценарий Д (неподдерживаемый формат файла):

O2: "Введите путь к файлу (CSV или XML)"

I4: address.txt

O5: "Неподдерживаемый формат файла. Используйте .csv или .xml"

4. Текст программы

```
import java.io.*;
import java.util.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;

class Address {
    private String city;
    private String street;
    private String house;
    private int floor;

    public Address(String city, String street, String house, int floor) {
        this.city = city;
        this.street = street;
        this.house = house;
        this.floor = floor;
    }

    public String getCity() { return city; }
```

```
    public String getStreet() { return street; }
    public String getHouse() { return house; }
    public int getFloor() { return floor; }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Address address = (Address) o;
        return floor == address.floor &&
                Objects.equals(city, address.city) &&
                Objects.equals(street, address.street) &&
                Objects.equals(house, address.house);
    }

    @Override
    public int hashCode() {
        return Objects.hash(city, street, house, floor);
    }

    @Override
    public String toString() {
        return String.format("Город: %s, Улица: %s, Дом: %s, Этажей: %d",
                city, street, house, floor);
    }
}

abstract class FileParser {
    public abstract List<Address> parse(String filePath);
}
```

```
class CsvParser extends FileParser {
    @Override
    public List<Address> parse(String filePath) {
        List<Address> addresses = new ArrayList<>();

        try (BufferedReader br = new BufferedReader(new
FileReader(filePath))) {
            String line;
            boolean isFirstLine = true;

            while ((line = br.readLine()) != null) {
                if (isFirstLine) {
                    isFirstLine = false;
                    continue;
                }

                String[] parts = line.split("[,;]");
                if (parts.length >= 4) {
                    try {
                        String city = parts[0].trim().replace("\"", "");
                        String street = parts[1].trim().replace("\"", "");
                        String house = parts[2].trim();
                        int floor = Integer.parseInt(parts[3].trim());

                        addresses.add(new Address(city, street, house, floor));
                    } catch (NumberFormatException e) {
                        System.out.println("Пропущена некорректная строка: " +
line);
                    }
```

```
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println("Файл не найден: " + filePath);
    } catch (IOException e) {
        System.out.println("Ошибка чтения файла.");
    }

    return addresses;
  }
}

class XmlParser extends FileParser {
  @Override
  public List<Address> parse(String filePath) {
    List<Address> addresses = new ArrayList<>();

    try {
      DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();
      Document document = builder.parse(new File(filePath));

      NodeList itemNodes = document.getElementsByTagName("item");

      for (int i = 0; i < itemNodes.getLength(); i++) {
        Node itemNode = itemNodes.item(i);

        if (itemNode.getNodeType() == Node.ELEMENT_NODE) {
          Element itemElement = (Element) itemNode;
```

```
                try {
                    String city = itemElement.getAttribute("city");
                    String street = itemElement.getAttribute("street");
                    String house = itemElement.getAttribute("house");
                    int floor =
Integer.parseInt(itemElement.getAttribute("floor"));

                    addresses.add(new Address(city, street, house, floor));
                } catch (NumberFormatException e) {
                    System.out.println("Пропущена некорректная XML
запись");
                }
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println("Файл не найден: " + filePath);
    } catch (IOException e) {
        System.out.println("Ошибка чтения файла.");
    } catch (Exception e) {
        System.out.println("Ошибка при разборе XML файла.");
    }

    return addresses;
  }
}

class StatisticsCalculator {
  public Map<Address, Integer> findDuplicates(List<Address> addresses)
{
```

```
        Map<Address, Integer> addressCount = new HashMap<>();

        for (Address address : addresses) {
            addressCount.put(address, addressCount.getOrDefault(address, 0) +
1);
        }

        Map<Address, Integer> duplicates = new HashMap<>();
        for (Map.Entry<Address, Integer> entry : addressCount.entrySet()) {
            if (entry.getValue() > 1) {
                duplicates.put(entry.getKey(), entry.getValue());
            }
        }

        return duplicates;
    }

    public Map<String, int[]> calculateFloorStatistics(List<Address>
addresses) {
        Map<String, int[]> cityFloorStats = new HashMap<>();

        for (Address address : addresses) {
            String city = address.getCity();
            int floor = address.getFloor();

            int[] floors = cityFloorStats.getOrDefault(city, new int[5]);

            if (floor >= 1 && floor <= 5) {
                floors[floor - 1]++;
            }
```

```
            cityFloorStats.put(city, floors);
        }

        return cityFloorStats;
    }
}

public class AddressAnalyzer {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        StatisticsCalculator statsCalculator = new StatisticsCalculator();

        System.out.println("=== Анализатор адресов ===");
        System.out.println("Введите путь к файлу (CSV или XML)");
        System.out.println("Для выхода нажмите Ctrl+D или введите 'exit'");

        while (true) {
            System.out.print("\n> ");

            try {
                if (!scanner.hasNextLine()) {
                    System.out.println("Завершение работы...");
                    break;
                }

                String input = scanner.nextLine().trim();

                if (input.equalsIgnoreCase("exit")) {
                    System.out.println("Завершение работы...");
```

```
                break;
            }

            if (input.isEmpty()) {
                System.out.println("Пустой запрос. Введите путь к файлу
или 'exit' для выхода");
                continue;
            }

            long startTime = System.currentTimeMillis();

            List<Address> addresses = parseFile(input);

            if (!addresses.isEmpty()) {
                Map<Address, Integer> duplicates =
statsCalculator.findDuplicates(addresses);
                Map<String, int[]> floorStats =
statsCalculator.calculateFloorStatistics(addresses);

                System.out.println("\n=== РЕЗУЛЬТАТЫ СТАТИСТИКИ
===");

                System.out.println("\n--- ДУБЛИРУЮЩИЕСЯ ЗАПИСИ ---
");
                if (duplicates.isEmpty()) {
                    System.out.println("Дубликаты не найдены");
                } else {
                    for (Map.Entry<Address, Integer> entry :
duplicates.entrySet()) {
```

```
                    System.out.println(entry.getKey() + " - повторений: " +
entry.getValue());
                }
            }

            System.out.println("\n--- СТАТИСТИКА ПО ЭТАЖАМ ---");
            for (Map.Entry<String, int[]> entry : floorStats.entrySet()) {
                String city = entry.getKey();
                int[] floors = entry.getValue();

                System.out.printf("%s: 1-этажных: %d, 2-этажных: %d, 3-
этажных: %d, 4-этажных: %d, 5-этажных: %d%n",
                        city, floors[0], floors[1], floors[2], floors[3], floors[4]);
            }
        } else {
            System.out.println("Файл не содержит данных или
произошла ошибка при чтении");
        }

        long endTime = System.currentTimeMillis();
        System.out.println("\nВремя обработки: " + (endTime -
startTime) + " мс");

      } catch (NoSuchElementException e) {
        System.out.println("\nЗавершение работы...");
        break;
      }
    }

    scanner.close();
```

```
    }

    private static List<Address> parseFile(String filePath) {
        FileParser parser;

        if (filePath.toLowerCase().endsWith(".csv")) {
            parser = new CsvParser();
        } else if (filePath.toLowerCase().endsWith(".xml")) {
            parser = new XmlParser();
        } else {
            System.out.println("Неподдерживаемый формат файла.
Используйте .csv или .xml");
            return new ArrayList<>();
        }

        return parser.parse(filePath);
    }
}
```

5. Выводы

В результате выполнения практической работы было разработано консольное приложение для анализа адресных данных из CSV и XML файлов. Программа читает файлы с адресами в двух форматах: CSV и XML, выявляет полные дубликаты адресных записей с подсчетом количества повторений, строит детальную статистику по этажности зданий для каждого города (от 1 до 5 этажей). Также программа измеряет время обработки данных для оценки производительности.

В ходе разработки были изучены основные принципы объектно-ориентированного программирования для парсинга файлов, работа с файловой системой и обработка исключений при чтении данных, парсинг CSV-файлов с использованием потокового чтения и разбора строк,

обработка XML-документов с применением DOM-парсера и навигации по элементам.

В работе была применена коллекция Java (HashMap) для анализа данных. Результаты были выложены на Github.

https://github.com/wkinax