**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**КАФЕДРА САПР**

**Практическая работа №2**

**по дисциплине**

**«Объектно-ориентированное программирование»**

**Тема: «Консольная программа для работы с CSV и XML файлами»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3354 |  | Филимонова.Е.С. |
| Преподаватель |  | Кулагин М.В. |

Санкт-Петербург

2024

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.

1. **Спецификация программы**

**Основной класс** — City:

* Управляет процессом взаимодействия с пользователем.
* Обрабатывает ввод пути к файлу.
* Выбирает соответствующий обработчик (CSV или XML)
* Выводит статистику.

**Обработка Csv файлов** — CsvFile:

* Читает данные о зданиях (город, улица, дом, этажность).
* Передает их в объект CityStatistics.

**Обработка Xml файлов** — XmlFile:

* Читает данные о зданиях, извлекая их из элементов XML.
* Передает их в объект CityStatistics.

**Хранение статистики** — CityStatistics:

* Подсчитывает дублирующиеся записи (по ключу: город, улица, дом).
* Ведет учет количества зданий по этажам в каждом городе.

**Форматирование и вывод статистики** — CityStatisticsPrinter:

* Выводит список дублирующихся записей.
* Выводит количество зданий по этажам в каждом городе.

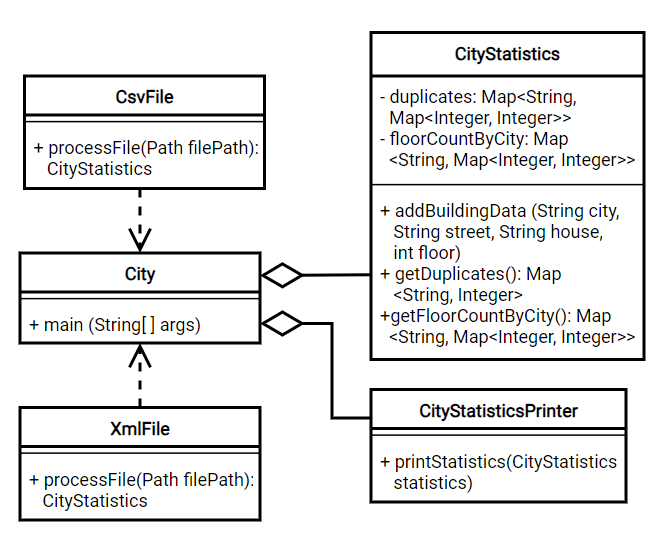


Рисунок 1. UML-диаграмма классов

**3. Описание интерфейса пользователя программы**

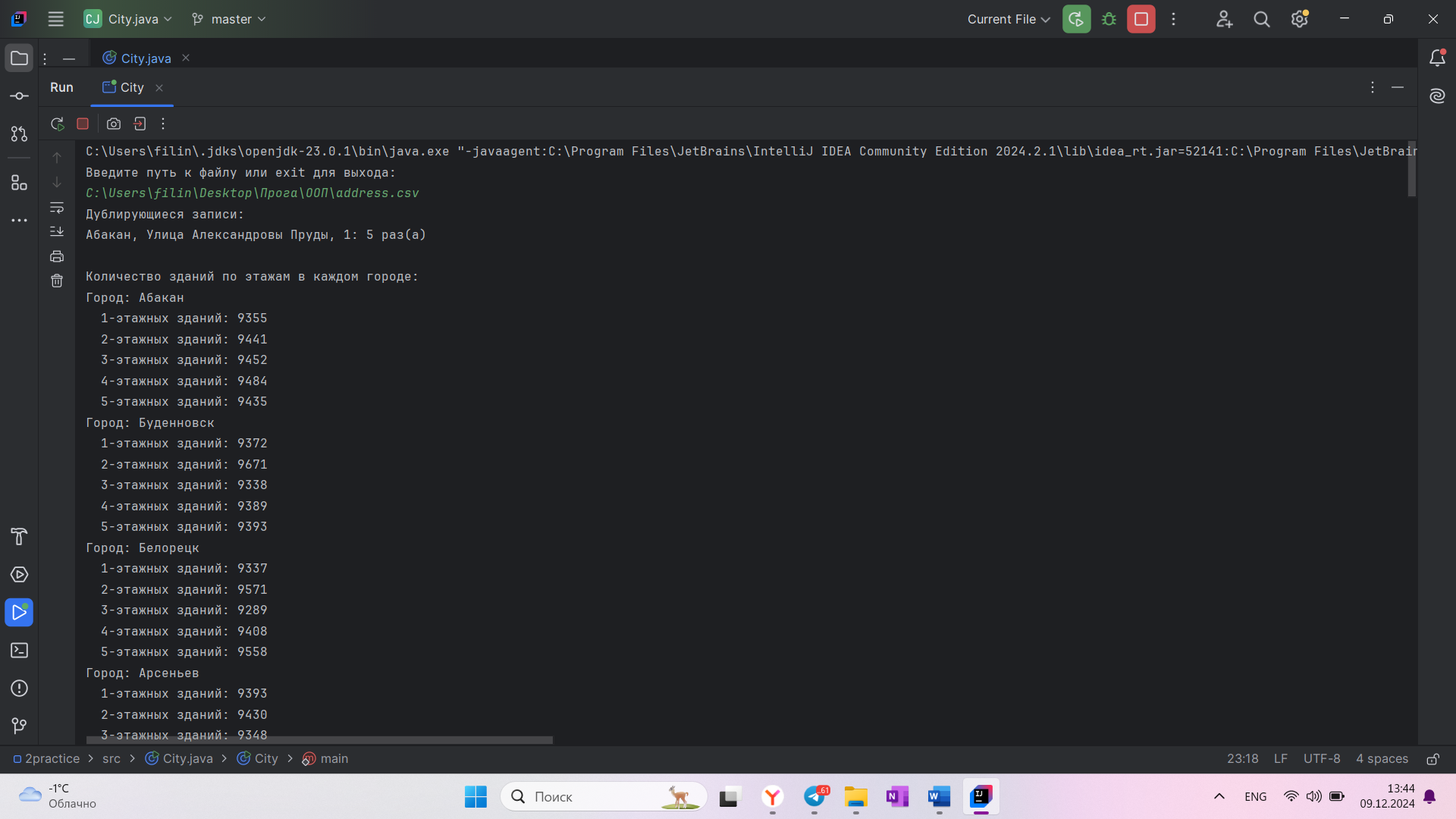
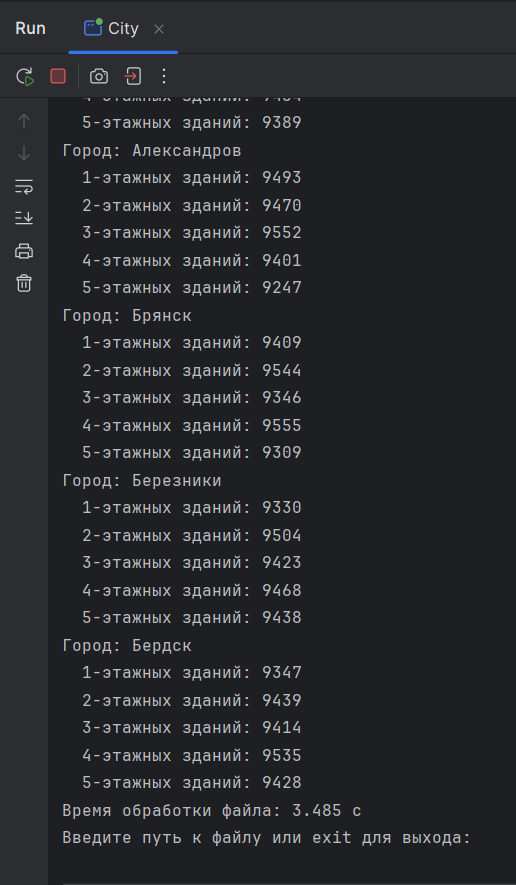
**** ****

Рисунок 2. Результат работы программы для csv файла

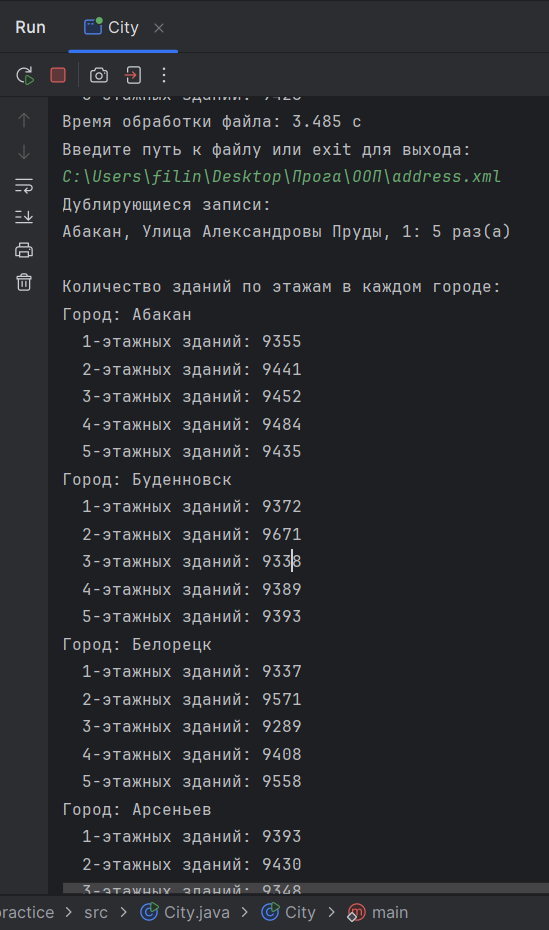
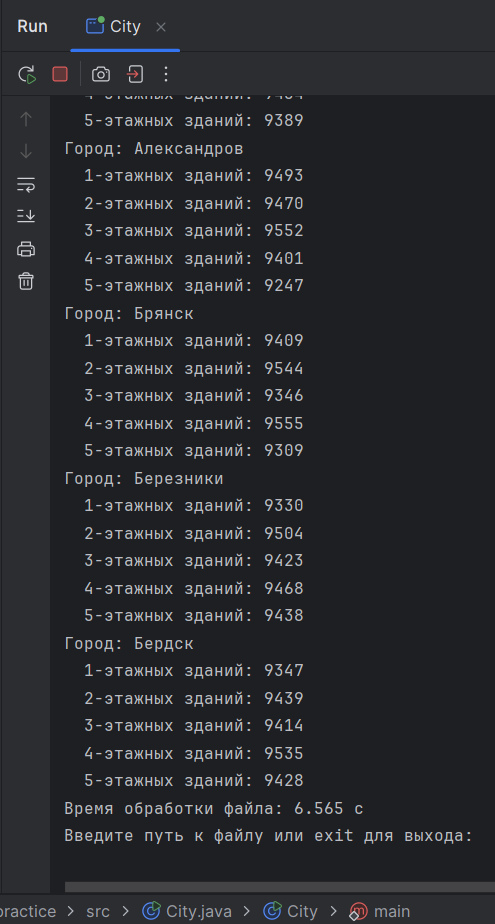
 

Рисунок 3. Результат работы программы для xml файла

Программа начинает выполнение с вывода сообщения: «Введите путь к файлу или exit для выхода:». После чего **пользователь вводит строку**, например, "C:\Users\filin\Desktop\Прога\ООП\address.xml" (при вводе «exit» программа завершит работу). После чего программа проверяет существует ли файл и соответствует ли его формат поддерживаемым типам. Для формата .csv используется CsvFile для чтения данных, для .xml – XmlFile. Извлекаются данные о городе, улице, доме и этажности зданий, статистика передается в объект CityStatistics. После обработки программа выводит: дублирующиеся записи зданий и статистику этажности по городам. После завершения обработки программа снова запрашивает путь к файлу.

В процессе пользователь может наткнуться на ошибки выполнения программы:

* Если введённая строка оказалась пустой, программа снова запросит ввести путь к файлу.
* Если введён путь к несуществующему файлу, то программа выдаст сообщение «Файл не найден, попробуйте снова.»
* Введён путь с неподдерживаемым расширением, последует сообщение «Неверный формат файла.»
* Если в файле некорректные данные, то программа не может их обработать и выдаст «Ошибка при обработке файла: <текст ошибки>».

**4. Текст программы**

import java.io.\*;

import java.util.\*;

import java.nio.file.\*;

import javax.xml.stream.\*;

import java.io.FileInputStream;

public class City {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Введите путь к файлу или exit для выхода:");

while (true) {

String input = scanner.nextLine().trim().toLowerCase().replace("\"", "");

if (input.equalsIgnoreCase("exit")) {

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

break;

}

Path filePath = Paths.get(input);

if (!Files.exists(filePath)) {

System.out.println("Файл не найден, попробуйте снова.");

continue;

}

long startTime = System.currentTimeMillis();

try {

if (input.endsWith(".xml")) {

XmlFile processor = new XmlFile();

CityStatistics statistics = processor.processFile(filePath);

CityStatisticsPrinter.printStatistics(statistics);

} else if (input.endsWith(".csv")) {

CsvFile processor = new CsvFile();

CityStatistics statistics = processor.processFile(filePath);

CityStatisticsPrinter.printStatistics(statistics);

}

else {

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

}

} catch (Exception e) {

System.out.println("Ошибка при обработке файла: " + e.getMessage());

}

long endTime = System.currentTimeMillis();

System.out.println("Время обработки файла: " + (endTime - startTime)/1000. + " с");

System.out.println("Введите путь к файлу или exit для выхода:");

}

scanner.close();

}

}

class CsvFile {

public CityStatistics processFile(Path filePath) throws Exception {

CityStatistics statistics = new CityStatistics();

try (BufferedReader br = Files.newBufferedReader(filePath)) {

String line=br.readLine();

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

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

String city = values[0].replace("\"", "").trim();

String street = values[1].replace("\"", "").trim();

String house = values[2].replace("\"", "").trim();

int floor = Integer.parseInt(values[3].replace("\"", "").trim());

statistics.addBuildingData(city, street, house, floor);

}

}

return statistics;

}

}

class XmlFile {

public CityStatistics processFile(Path filePath) throws Exception {

CityStatistics statistics = new CityStatistics();

XMLInputFactory factory = XMLInputFactory.newInstance();

try (FileInputStream fis = new FileInputStream(filePath.toFile())) {

XMLStreamReader reader = factory.createXMLStreamReader(fis);

while (reader.hasNext()) {

int event = reader.next();

if (event == XMLStreamConstants.START\_ELEMENT && "item".equals(reader.getLocalName())) {

String city = reader.getAttributeValue(null, "city");

String street = reader.getAttributeValue(null, "street");

String house = reader.getAttributeValue(null, "house");

int floor = Integer.parseInt(reader.getAttributeValue(null, "floor"));

statistics.addBuildingData(city, street, house, floor);

}

}

reader.close();

}

return statistics;

}

}

class CityStatistics {

private final Map<String, Integer> duplicates = new HashMap<>();

private final Map<String, Map<Integer, Integer>> floorCountByCity = new HashMap<>();

public void addBuildingData(String city, String street, String house, int floor) {

String buildingKey = city + ", " + street + ", " + house;

duplicates.put(buildingKey, duplicates.getOrDefault(buildingKey, 0) + 1);

floorCountByCity.computeIfAbsent(city, k -> new HashMap<>())

.merge(floor, 1, Integer::sum);

}

public Map<String, Integer> getDuplicates() {

return duplicates;

}

public Map<String, Map<Integer, Integer>> getFloorCountByCity() {

return floorCountByCity;

}

}

class CityStatisticsPrinter {

public static void printStatistics(CityStatistics statistics) {

System.out.println("Дублирующиеся записи:");

statistics.getDuplicates().entrySet().stream()

.filter(entry -> entry.getValue() > 1)

.forEach(entry -> System.out.println(entry.getKey() + ": " + entry.getValue() + " раз(а)"));

System.out.println("\nКоличество зданий по этажам в каждом городе:");

statistics.getFloorCountByCity().forEach((city, floorCounts) -> {

System.out.println("Город: " + city);

for (int i = 1; i <= 5; i++) {

int count = floorCounts.getOrDefault(i, 0);

System.out.println(" " + i + "-этажных зданий: " + count);

}

});

}

1. **Выводы**

**Класс City**:

Метод main(String[] args): Для начала создает объект Scanner для ввода данных пользователем, после чего выводит приглашение ввести путь к файлу или команду exit для завершения работы. Пользователь вводит строку, программа проверяет ввод: если путь пустой или некорректный, программа запрашивает ввод снова, если путь верный, то программа определяет, существует ли файл и соответствует ли его формат возможным. Если файл .xml, вызывается XmlFile.processFile(). Если файл .csv, вызывается CsvFile.processFile(). Далее результат передается в printStatistics() для вывода. А также на протяжении всей обработки файла замеряется время, которая после выводится в секундах.

**Класс CsvFile**:

Метод processFile(Path filePath): читает файл построчно с помощью BufferedReader, где для каждой строки файла: разделяет данные на город, улицу, дом и этажность. Далее преобразует эти данные в нужный формат и добавляет их в объект CityStatistics с помощью addBuildingData(). Если файл содержит некорректные данные, вызывается исключение Exception.

**Класс XmlFile**:

Метод processFile(Path filePath): создает объект XMLInputFactory, читает файл с помощью XMLStreamReader. Для каждого элемента извлекает атрибуты: city, street, house, floor; преобразует данные и добавляет их в объект CityStatistics.

**Класс CityStatistics**:

duplicates используется для подсчета количества повторяющихся записей.

floorCountByCity - для подсчета зданий по этажам в каждом городе.

Методы addBuildingData(String city, String street, String house, int floor): создает ключ для записи здания, увеличивает счетчик повторений для этого ключа и счетчик зданий с заданным количеством этажей для города.

Метод getDuplicates(): возвращает дублирующиеся записи.

Метод getFloorCountByCity(): возвращает количество зданий по этажам в каждом городе.

**Класс CityStatisticsPrinter**:

Метод printStatistics(CityStatistics statistics): фильтрует записи, встречающиеся более одного раза, и выводит их на экран. Перебирает города и выводит количество зданий с 1 по 5 этажей.

Результаты были выложены на Github:

[https://github.com/Filimonova-Ekaterina/OOP\_Practice\_2/blob/master/src/City.java](https://vk.com/away.php?to=https%3A%2F%2Fgithub.com%2FFilimonova-Ekaterina%2FOOP_Practice_2%2Fblob%2Fmaster%2Fsrc%2FCity.java&utf=1)