Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших данных в системах поддержки принятия решений.**

О Т Ч Е Т

по лабораторной работе № 4

Вариант № 11

Название: Внутренние классы, интерфейсы

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-23М (Группа) _____ (Подпись, дата) С.В.Мельников (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) П.В. Степанов (И.О. Фамилия)

Москва, 2023

Цель работы

Изучить принципы написания внутренних классов в языке программирования Java и рассмотреть интерфейсы.

Задание 1 (Вариант 1, Задание 1):

Создать класс City (город) с внутренним классом, с помощью объектов которого можно хранить информацию о проспектах, улицах, площадях.

Листинг программы:

Код класса City:

```
public class City {
    private final String name;
    private final ArrayList<Road> roads;

    public class Road {
        private final String name;
        private final String type;
        private final ArrayList<Pair<Road, Integer>> roads;

        public Road(String name, String type) {
            this.name = name;
            this.type = type;
            this.roads = new ArrayList<>();
        }

        public void addTurn(Road road, int length) {
            this.roads.add(new Pair<>(road, length));
        }

        @Override
        public String toString() {
            return type + ": " + name;
        }
    }

    public City(String name) {
        this.name = name;
        this.roads = new ArrayList<>();
    }

    public Road findRoad(String road_name) {
        for (Road road: this.roads) {
            if (road_name.equals(road.name)) {
                return road;
            }
        }
        return null;
    }

    public void addRoad(String road_name, String road_type, ArrayList<Pair<String,
Integer>> connection_list) {
        Road new_road = new Road(road_name, road_type);
        this.roads.add(new_road);
        if (connection_list != null) {
```

```
            for (Pair<String, Integer> connection: connection_list) {
                Road road = this.findRoad(connection.first);
                new_road.addTurn(road, connection.second);
                road.addTurn(new_road, connection.second);
            }
        }
    }

    @Override
    public String toString() {
        StringBuilder res = new StringBuilder("City: " + name + '\n');
        for (Road road: roads) {
            res.append(road.toString()).append('\n');
        }
        return res.toString();
    }

    public void findWay(String road_name_from, String road_name_to) {
        HashMap<Road, Pair<Integer, Road>> ways = new HashMap<>();
        for (Road road: this.roads) {
            ways.put(road, new Pair<>(100000, null));
        }
        ArrayList<Road> not_visited = new ArrayList<>(this.roads);
        Road start = this.findRoad(road_name_from);
        ways.replace(start, new Pair<>(0, null));
        while (!not_visited.isEmpty()) {
            Road curr_v = null;
            int cost = 100000;
            for (Map.Entry<Road, Pair<Integer, Road>> vertex: ways.entrySet()) {
                if ((vertex.getValue().first < cost || curr_v == null) &&
not_visited.contains(vertex.getKey())) {
                    cost = vertex.getValue().first;
                    curr_v = vertex.getKey();
                }
            }
            for (Pair<Road, Integer> connection: curr_v.roads) {
                if (not_visited.contains(connection.first) &&
ways.get(connection.first).first > (cost + connection.second)) {
                    ways.replace(connection.first, new Pair<>(cost +
connection.second, curr_v));
                }
            }
            not_visited.remove(curr_v);
        }
        ArrayList<Road> best_roads = new ArrayList<>();
        Road step = this.findRoad(road_name_to);
        while (step != null) {
            best_roads.add(step);
            step = ways.get(step).second;
        }
        StringBuilder result = new StringBuilder();
        for (int i = best_roads.size() - 1; i >= 0; i--) {
            result.append(best_roads.get(i));
            if (i != 0) {
                result.append(" -> ");
            }
        }
        System.out.println(result);
    }
}
```

Работа программы представлена на рисунке 1.

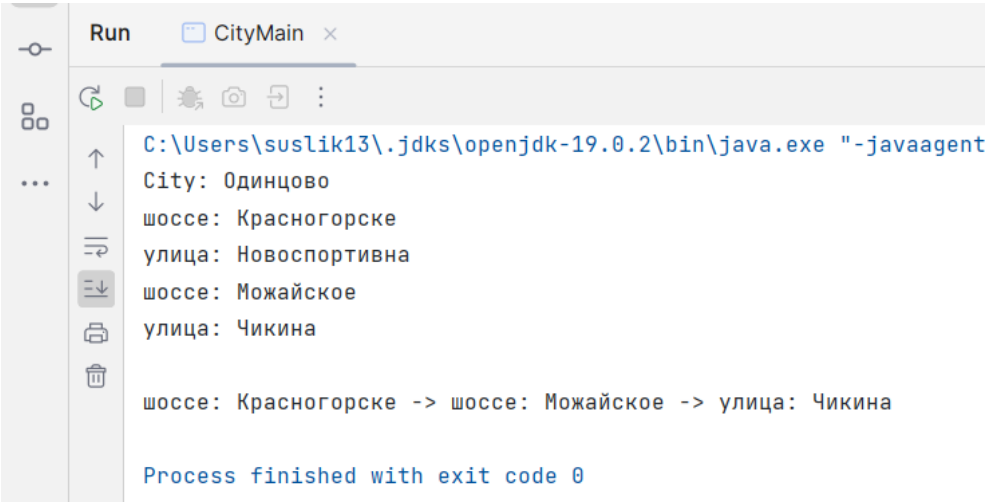

Рисунок 1 – Работа программы CityMain.java

Задание 2 (Вариант 1, Задание 2):

Создать класс CD (mp3-диск) с внутренним классом, с помощью объектов которого можно хранить информацию о каталогах, подкаталогах и записях. Листинг программы:

Код класса CD:

```
public class CD {
    private final String name;
    private final Document root;

    public class Document {
        private final boolean isFile;
        private final String name;
        private final ArrayList<Document> documents;

        public Document(String name, boolean isFile) {
            this.name = name;
            this.isFile = isFile;
            if (isFile) {
                this.documents = null;
            } else{
                this.documents = new ArrayList<>();
            }
        }

        public String getName() {
            return name;
        }
    }

    public CD(String name) {
        this.name = name;
        root = new Document(name, false);
    }
```

```
    public void addFile(String path) {
        String[] tokens = path.split("/");
        Document currDoc = root;
        for (int i = 0; i < tokens.length; i++) {
            if (!currDoc.isFile) {
                if (currDoc.documents.size() > 0) {
                    boolean found = false;
                    for (Document doc: currDoc.documents) {
                        if (doc.name.equals(tokens[i])) {
                            currDoc = doc;
                            found = true;
                            break;
                        }
                    }
                    if (!found) {
                        Document new_doc = new Document(tokens[i], i ==
tokens.length - 1);
                        currDoc.documents.add(new_doc);
                        currDoc = new_doc;
                    }
                } else {
                    Document new_doc = new Document(tokens[i], i == tokens.length
- 1);
                    currDoc.documents.add(new_doc);
                    currDoc = new_doc;
                }
            }
        }
    }

    public void tree() {
        System.out.println("CD: " + this.name + ":");
        tree_part(root, 0);
    }

    private void tree_part(Document doc, int tabs) {
        for (int i = 0; i < tabs; i++) {
            System.out.print("-");
        }
        if (doc.isFile) {
            System.out.println(doc.name);
        } else {
            System.out.println(doc.name);
            for (Document next: doc.documents) {
                tree_part(next,tabs + 2);
            }
        }
    }
}
```

Работа программы представлена на рисунке 2.

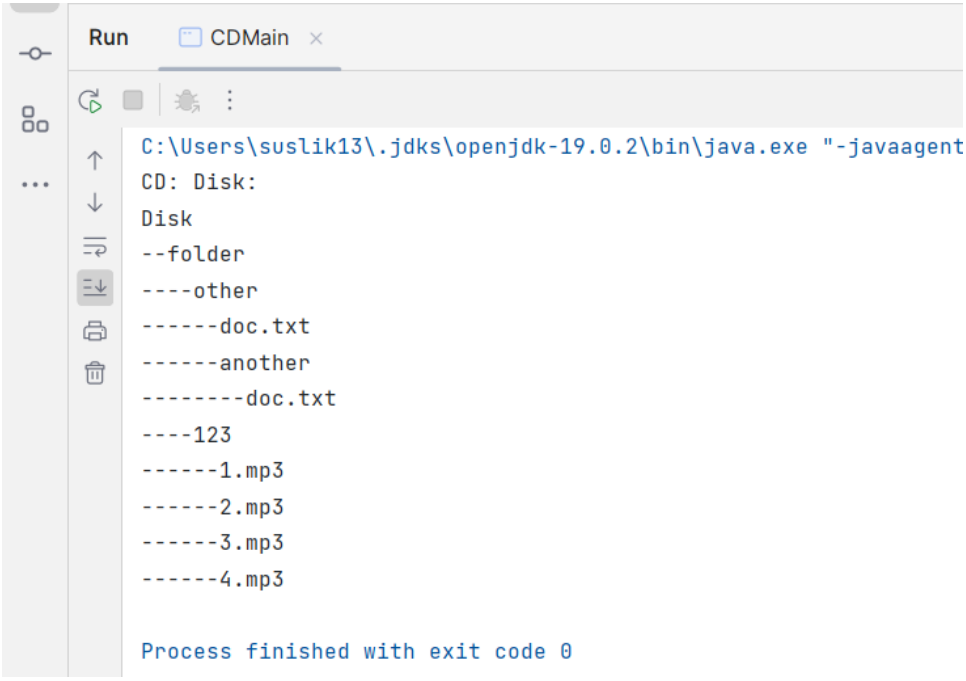

Рисунок 2 – Работа программы CDMain.java

Задание 3 (Вариант 2, Задание 1):

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

Абстрактный класс Книга (Шифр, Автор, Название, Год, Издательство). Подклассы Справочник и Энциклопедия.

Листинг программы:

Код класса Book:

```
public abstract class Book {
    protected final String cipher;
    protected final String author;
    protected final String name;
    protected final int year;
    protected final String publisher;

    protected Book(String cipher, String author, String name, int year, String
publisher) {
        this.cipher = cipher;
        this.author = author;
        this.name = name;
        this.year = year;
        this.publisher = publisher;
    }
```

```
    public abstract void printBook();
}
```

Код класса Encyclopedia:

```
public class Encyclopedia extends Book {
    public Encyclopedia(String cipher, String author, String name, int year,
String publisher) {
        super(cipher, author, name, year, publisher);
    }

    @Override
    public void printBook() {
        System.out.println("Это энциклопедия");
        System.out.println("        Шифр: " + this.cipher);
        System.out.println("       Автор: " + this.author);
        System.out.println("    Название: " + this.name);
        System.out.println("         Год: " + this.year);
        System.out.println("Издательство: " + this.publisher);
    }
}
```

Код класса Handbook:

```
public class Handbook extends Book {

    public Handbook(String cipher, String author, String name, int year, String
publisher) {
        super(cipher, author, name, year, publisher);
    }

    @Override
    public void printBook() {
        System.out.println("Это справочник");
        System.out.println("        Шифр: " + this.cipher);
        System.out.println("       Автор: " + this.author);
        System.out.println("    Название: " + this.name);
        System.out.println("         Год: " + this.year);
        System.out.println("Издательство: " + this.publisher);
    }
}
```

Работа программы представлена на рисунке 3.

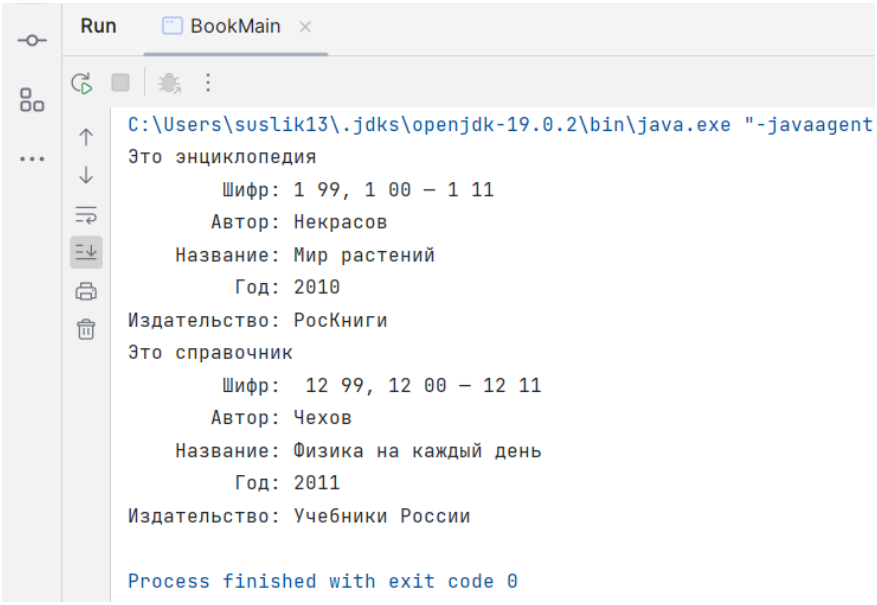

Рисунок 3 – Работа программы BookMain.java

Задание 4 (Вариант 2, Задание 2):

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов

interface Абитуриент <- abstract class Студент <- class Студент-Заочник.

Листинг программы:

Код класса Enrollee:

```
public interface Enrollee {
    void prepare_for_exams();
}
```

Код класса Student:

```
public abstract class Student implements Enrollee {
    protected final String name;
    protected final int age;

    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public abstract void sayHi();

    @Override
    public void prepare_for_exams() {
        System.out.println("Студент " + name + " готовится к экзамену");
    }
}
```

Код класса DistanceStudent:

```
public class DistanceStudent extends Student {
    public DistanceStudent(String name, int age) {
        super(name, age);
    }

    @Override
    public void sayHi() {
        System.out.println("Студент " + name + ", " + age + " лет. На заочном
обучении");
    }
}
```

Работа программы представлена на рисунке 4.

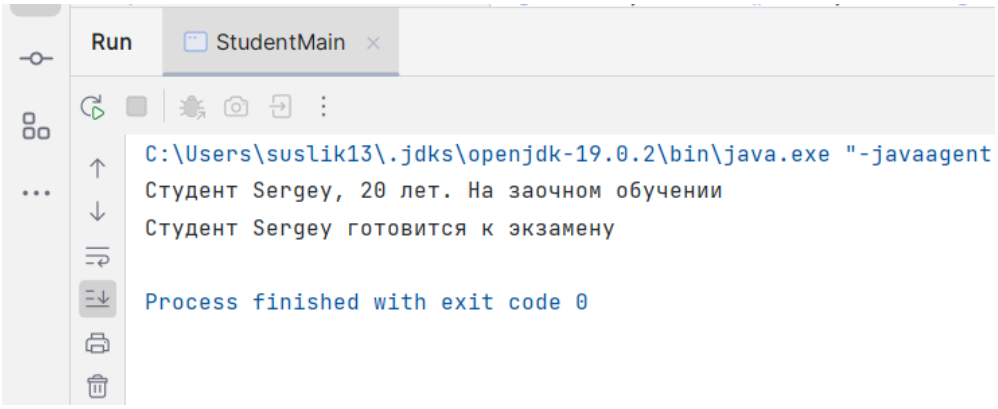

Рисунок 4 – Работа программы StudentMain.java

Вывод

В ходе выполнения лабораторной работы были изучены интерфейсы и вложенные классы в языке программирования Java.