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 **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных.**

О Т Ч Е Т

по лабораторной работе № 3

Вариант 6

Название: Арифметические операции

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-23М		В.А. Елисеев
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2022

Цель работы: получение навыков работы с классами Java, исследование механизмов наследования и полиморфизма.

Задание 1:

6. Определить класс Цепная дробь

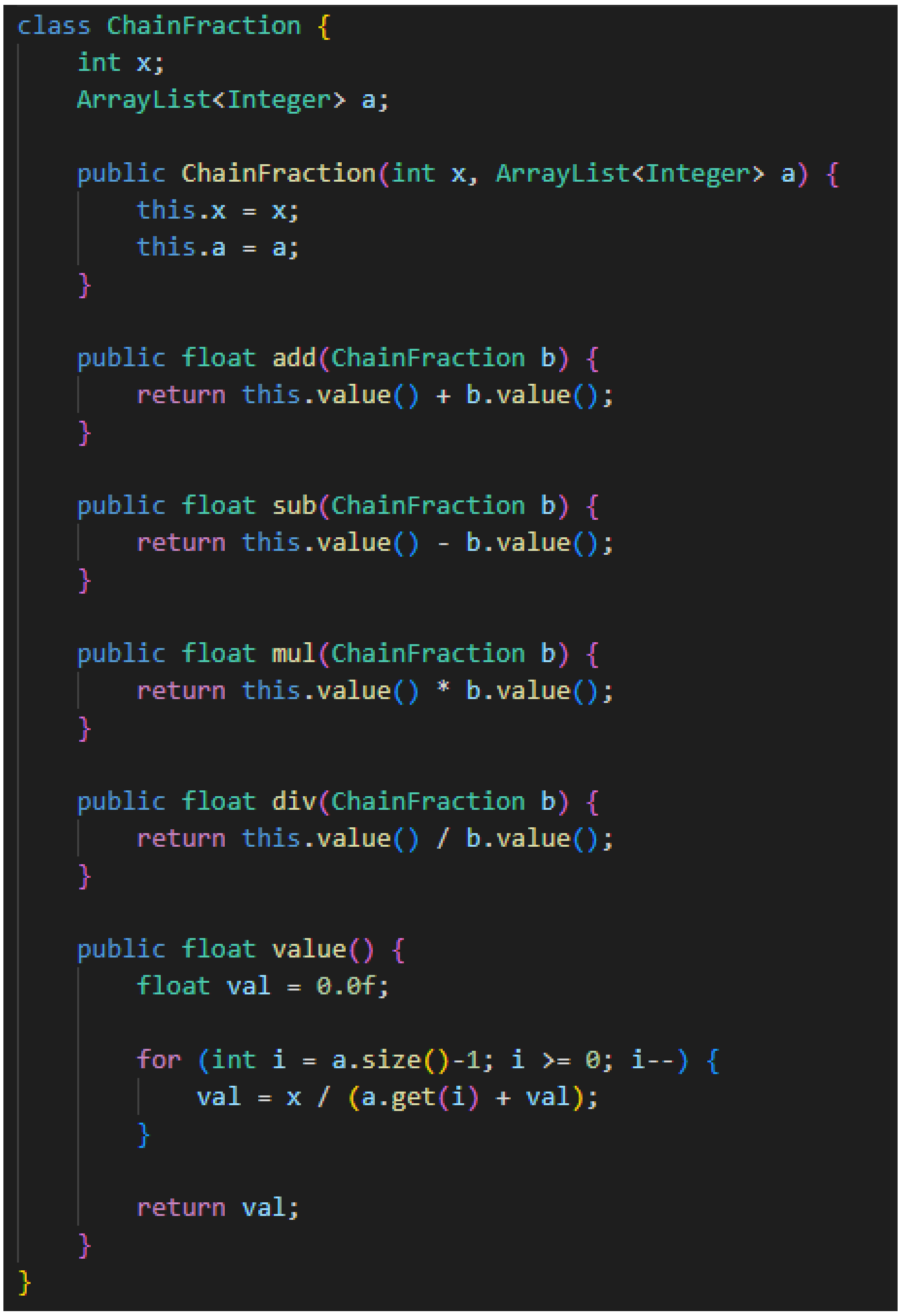
$$A = a_0 + \cfrac{x}{a_1 + \cfrac{x}{a_2 + \cfrac{x}{a_3 + \ldots}}}$$

Определить методы сложения, вычитания, умножения, деления.
Вычислить значение для заданного n, x, a[n].

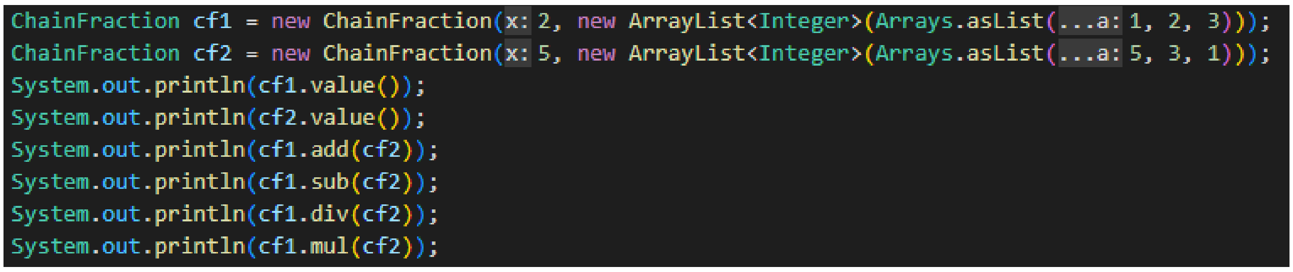
7. Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей, ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.

Подзадача 1.

Код программы:

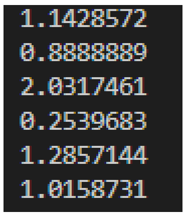


```java
class ChainFraction {
    int x;
    ArrayList<Integer> a;

    public ChainFraction(int x, ArrayList<Integer> a) {
        this.x = x;
        this.a = a;
    }

    public float add(ChainFraction b) {
        return this.value() + b.value();
    }

    public float sub(ChainFraction b) {
        return this.value() - b.value();
    }

    public float mul(ChainFraction b) {
        return this.value() * b.value();
    }

    public float div(ChainFraction b) {
        return this.value() / b.value();
    }

    public float value() {
        float val = 0.0f;

        for (int i = a.size()-1; i >= 0; i--) {
            val = x / (a.get(i) + val);
        }

        return val;
    }
}
```



```java
ChainFraction cf1 = new ChainFraction(x: 2, new ArrayList<Integer>(Arrays.asList(...a: 1, 2, 3)));
ChainFraction cf2 = new ChainFraction(x: 5, new ArrayList<Integer>(Arrays.asList(...a: 5, 3, 1)));
System.out.println(cf1.value());
System.out.println(cf2.value());
System.out.println(cf1.add(cf2));
System.out.println(cf1.sub(cf2));
System.out.println(cf1.div(cf2));
System.out.println(cf1.mul(cf2));
```

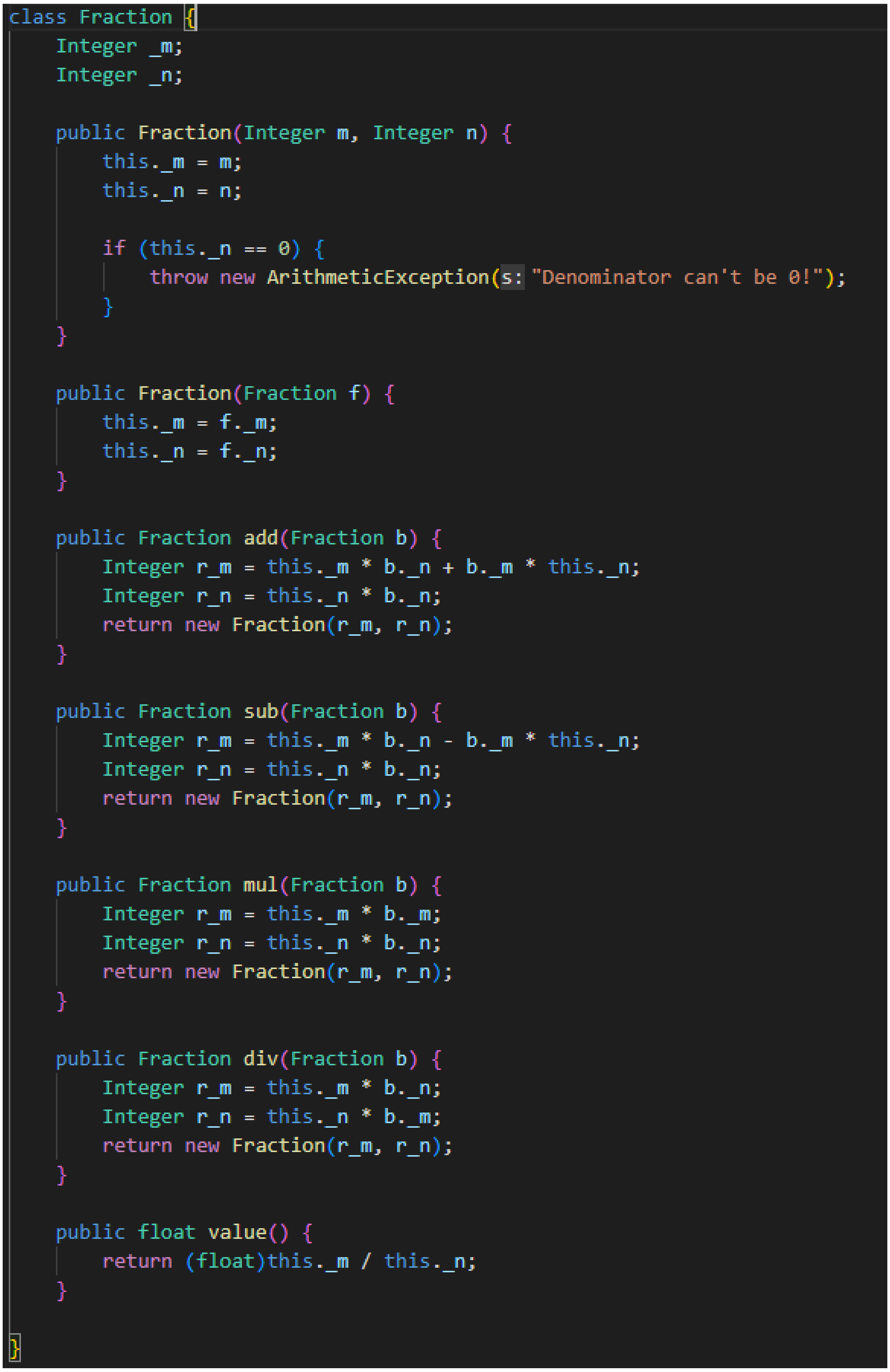
Процесс работы программы:



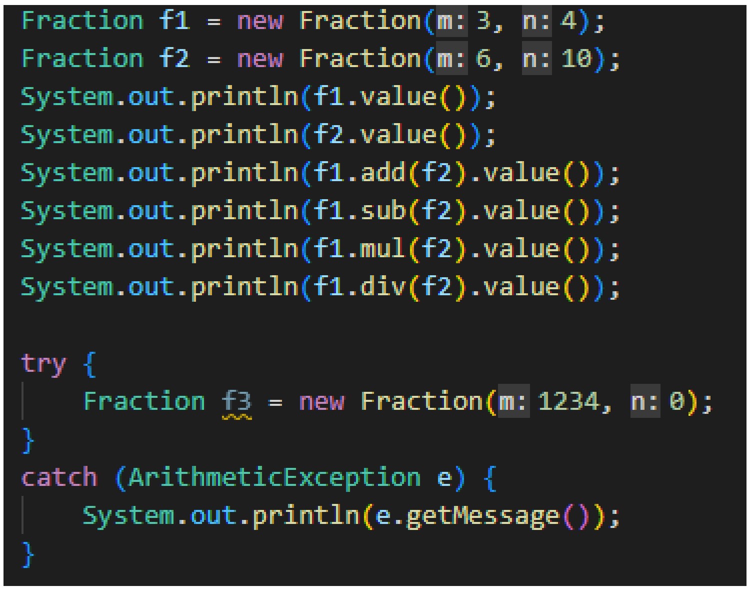
```
1.1428572
0.8888889
2.0317461
0.2539683
1.2857144
1.0158731
```

Подзадача 2.

Код программы:

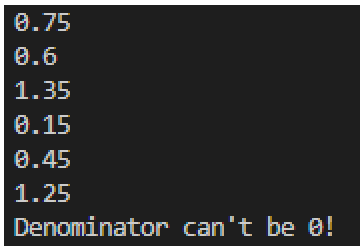


```java
class Fraction {
    Integer _m;
    Integer _n;

    public Fraction(Integer m, Integer n) {
        this._m = m;
        this._n = n;

        if (this._n == 0) {
            throw new ArithmeticException(s: "Denominator can't be 0!");
        }
    }

    public Fraction(Fraction f) {
        this._m = f._m;
        this._n = f._n;
    }

    public Fraction add(Fraction b) {
        Integer r_m = this._m * b._n + b._m * this._n;
        Integer r_n = this._n * b._n;
        return new Fraction(r_m, r_n);
    }

    public Fraction sub(Fraction b) {
        Integer r_m = this._m * b._n - b._m * this._n;
        Integer r_n = this._n * b._n;
        return new Fraction(r_m, r_n);
    }

    public Fraction mul(Fraction b) {
        Integer r_m = this._m * b._m;
        Integer r_n = this._n * b._n;
        return new Fraction(r_m, r_n);
    }

    public Fraction div(Fraction b) {
        Integer r_m = this._m * b._n;
        Integer r_n = this._n * b._m;
        return new Fraction(r_m, r_n);
    }

    public float value() {
        return (float)this._m / this._n;
    }

}
```



```java
Fraction f1 = new Fraction(m: 3, n: 4);
Fraction f2 = new Fraction(m: 6, n: 10);
System.out.println(f1.value());
System.out.println(f2.value());
System.out.println(f1.add(f2).value());
System.out.println(f1.sub(f2).value());
System.out.println(f1.mul(f2).value());
System.out.println(f1.div(f2).value());

try {
    Fraction f3 = new Fraction(m: 1234, n: 0);
}
catch (ArithmeticException e) {
    System.out.println(e.getMessage());
}
```

Процесс работы программы:



```
0.75
0.6
1.35
0.15
0.45
1.25
Denominator can't be 0!
```

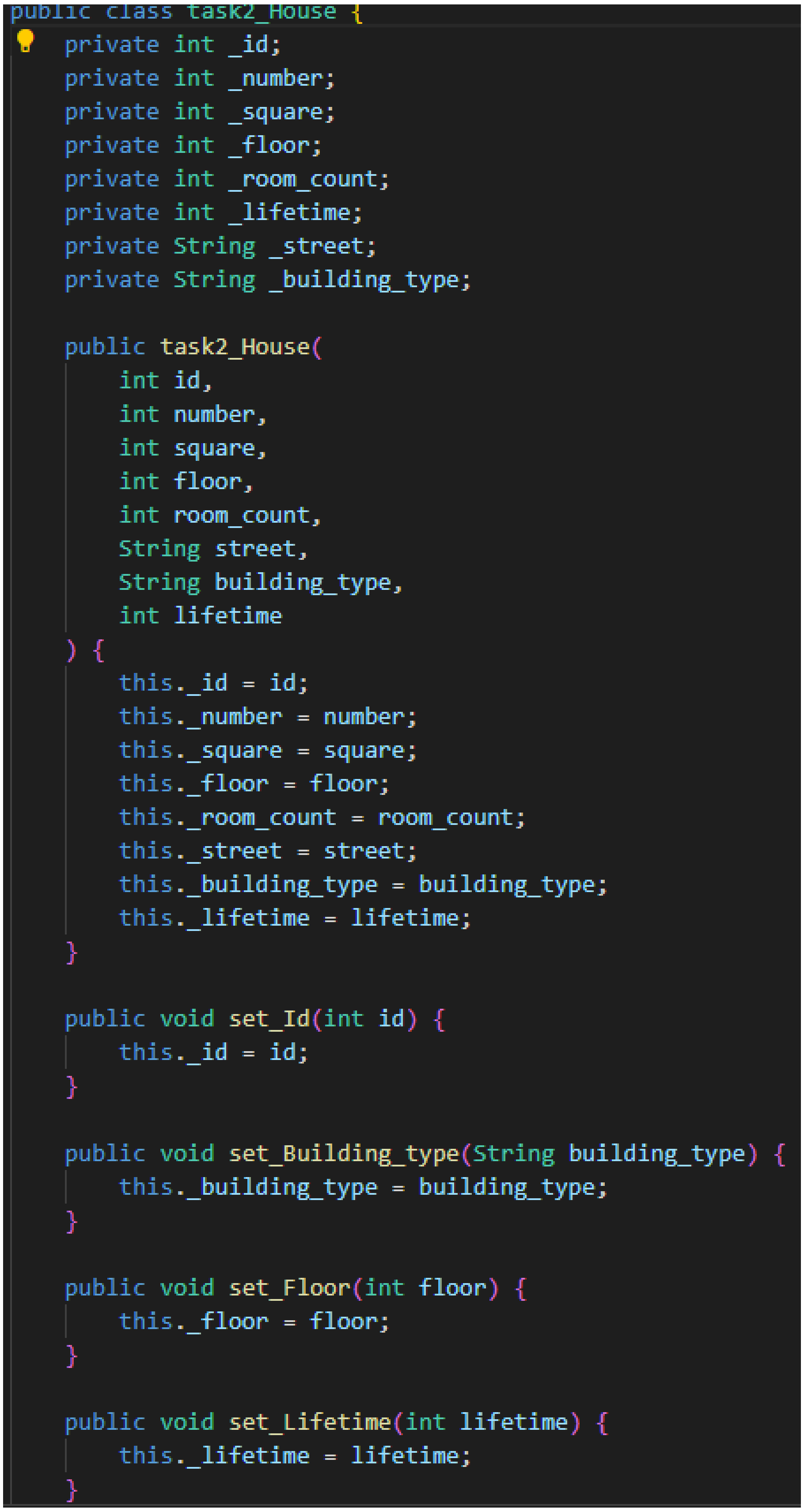
Задание 2:

6. House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: a) список квартир, имеющих заданное число комнат; b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; c) список квартир, имеющих площадь, превосходящую заданную.

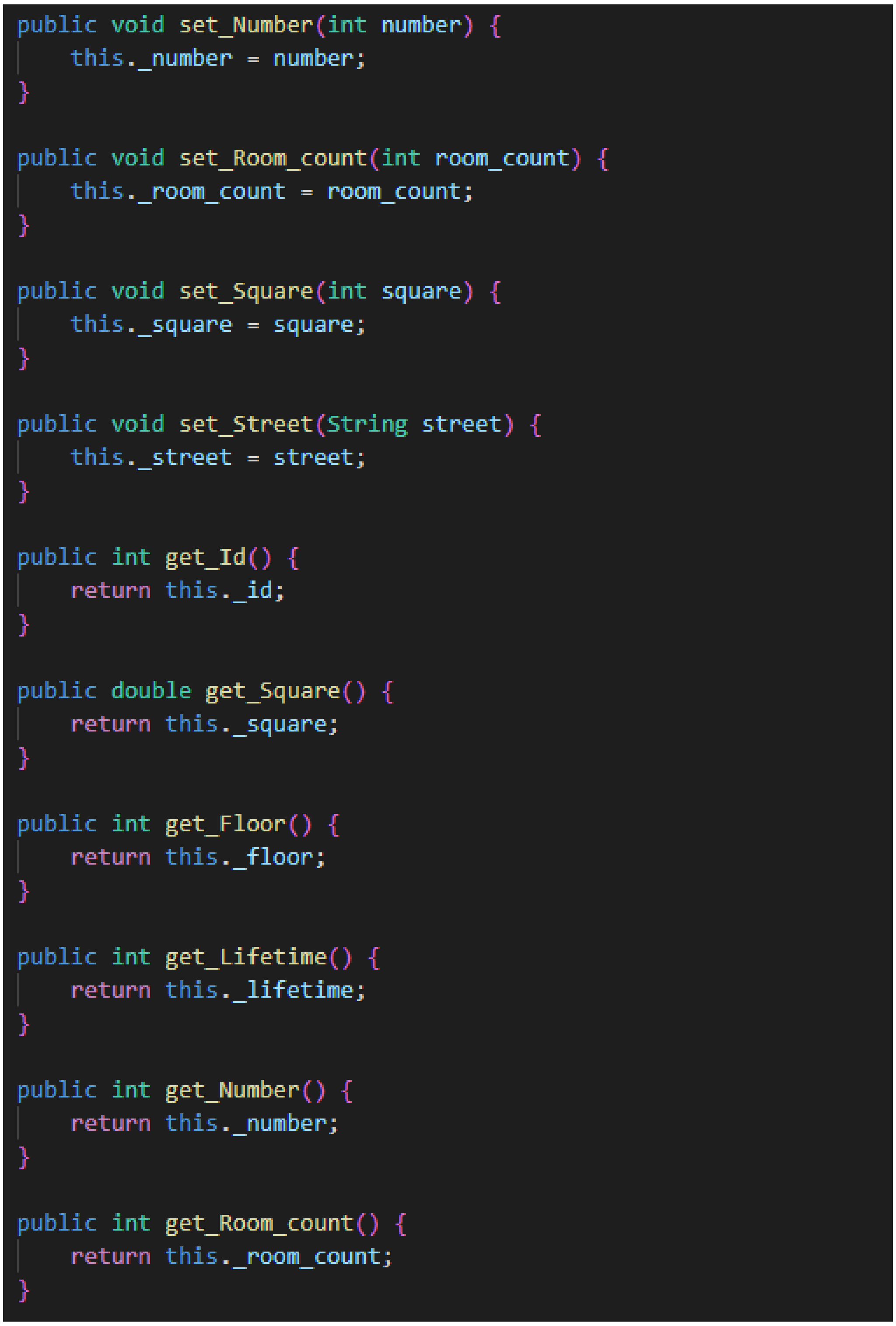
7. Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров. Создать массив объектов. Вывести: a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; b) сведения об абонентах, которые пользовались междугородной связью; c) сведения об абонентах в алфавитном порядке.

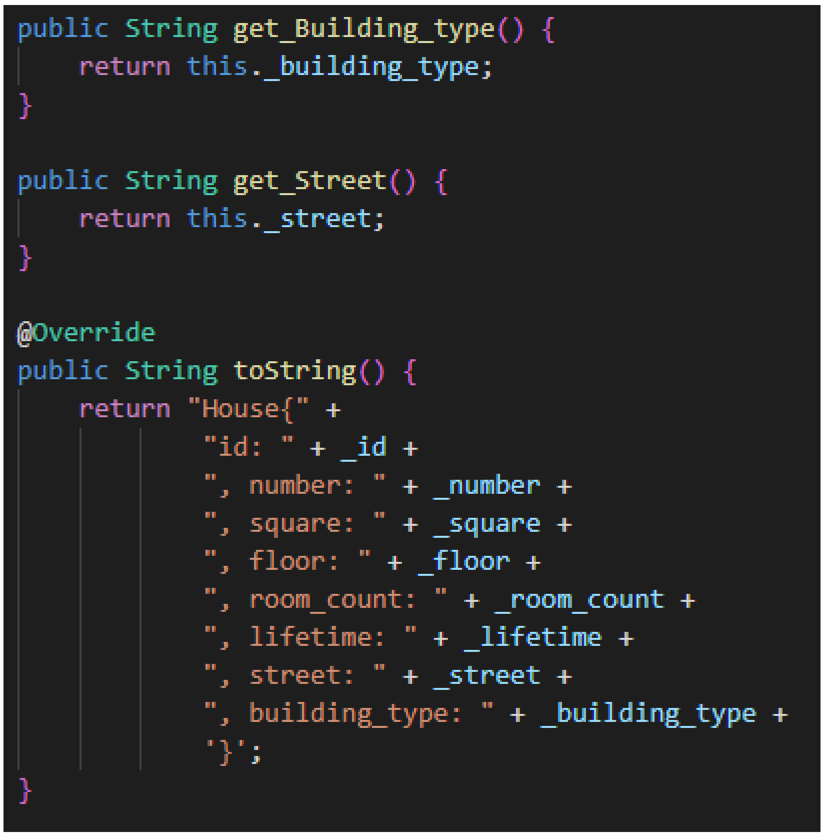
Подзадача 1.

Код программы:

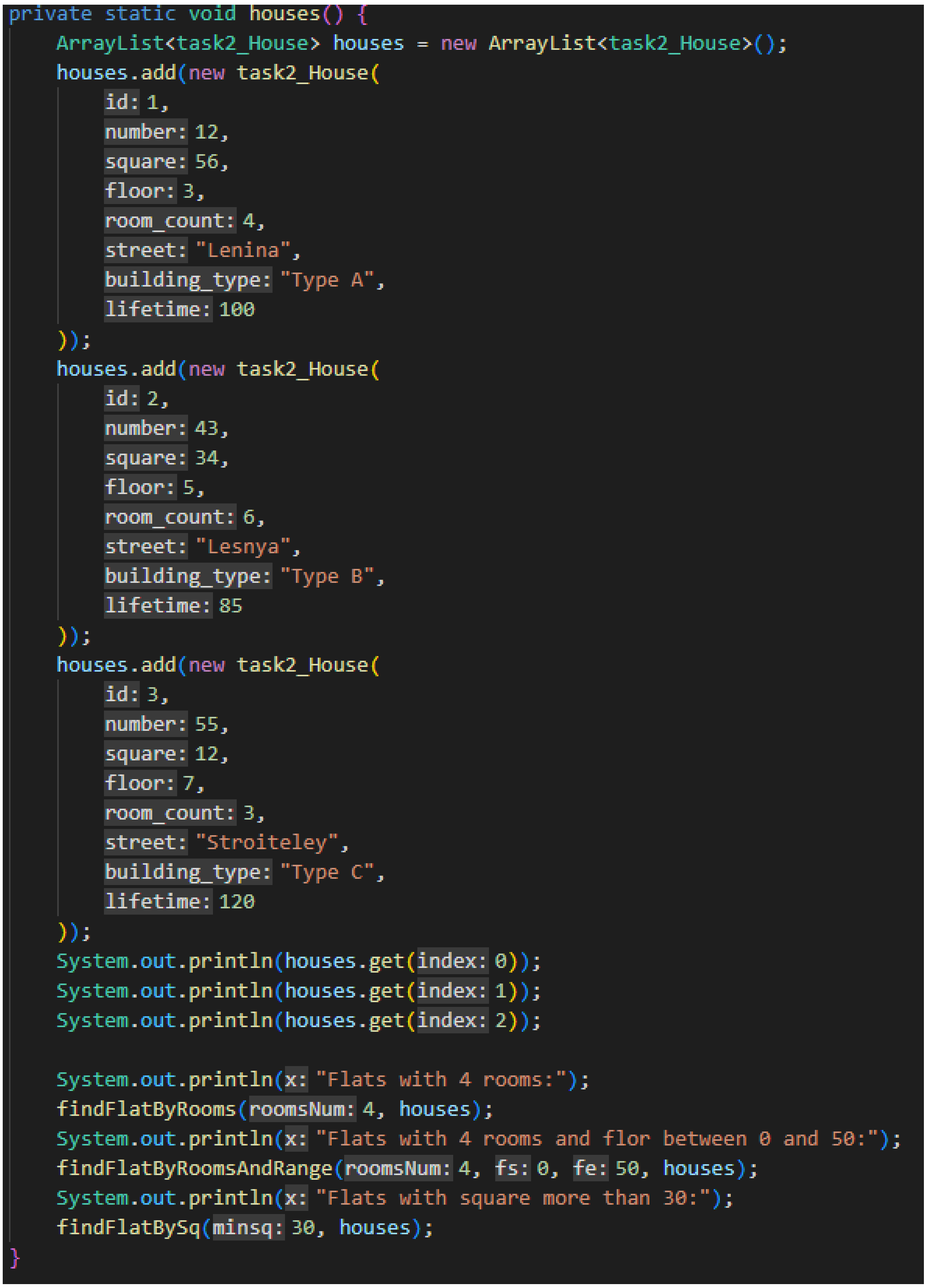


```
public class task2_House {
    private int _id;
    private int _number;
    private int _square;
    private int _floor;
    private int _room_count;
    private int _lifetime;
    private String _street;
    private String _building_type;

    public task2_House(
        int id,
        int number,
        int square,
        int floor,
        int room_count,
        String street,
        String building_type,
        int lifetime
    ) {
        this._id = id;
        this._number = number;
        this._square = square;
        this._floor = floor;
        this._room_count = room_count;
        this._street = street;
        this._building_type = building_type;
        this._lifetime = lifetime;
    }

    public void set_Id(int id) {
        this._id = id;
    }

    public void set_Building_type(String building_type) {
        this._building_type = building_type;
    }

    public void set_Floor(int floor) {
        this._floor = floor;
    }

    public void set_Lifetime(int lifetime) {
        this._lifetime = lifetime;
    }
```

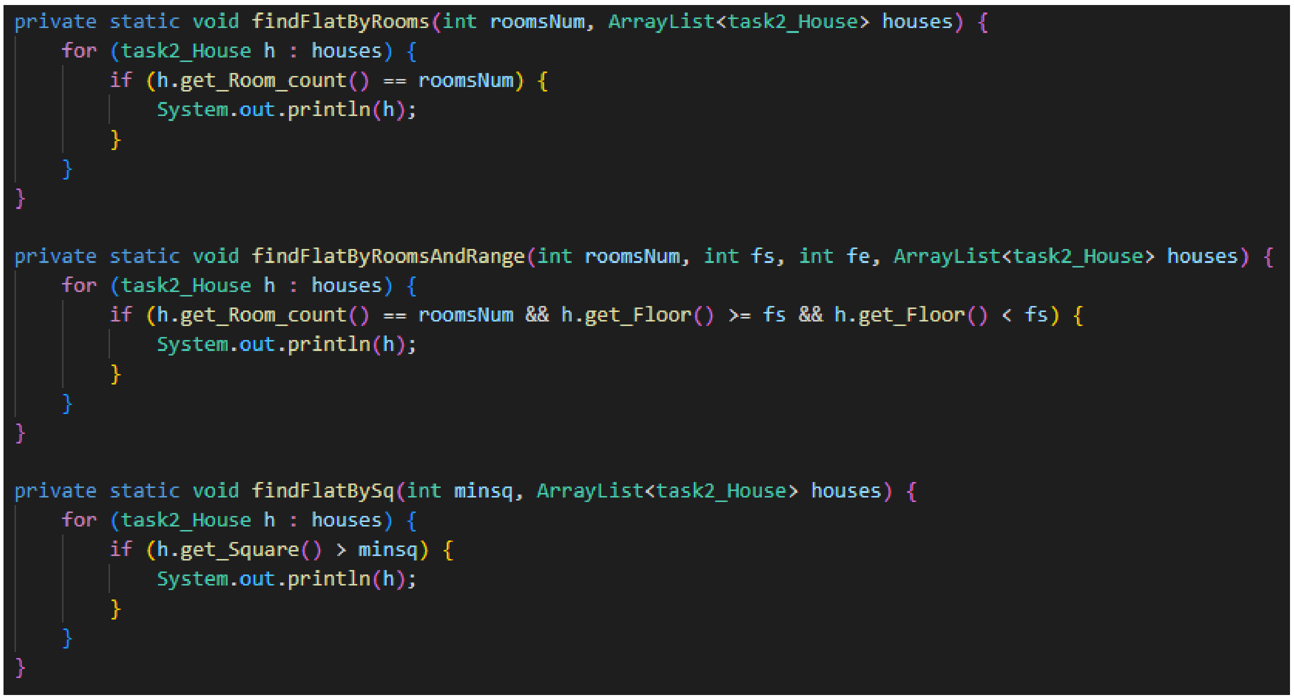

```
public void set_Number(int number) {
    this._number = number;
}

public void set_Room_count(int room_count) {
    this._room_count = room_count;
}

public void set_Square(int square) {
    this._square = square;
}

public void set_Street(String street) {
    this._street = street;
}

public int get_Id() {
    return this._id;
}

public double get_Square() {
    return this._square;
}

public int get_Floor() {
    return this._floor;
}

public int get_Lifetime() {
    return this._lifetime;
}

public int get_Number() {
    return this._number;
}

public int get_Room_count() {
    return this._room_count;
}
```



```java
public String get_Building_type() {
    return this._building_type;
}

public String get_Street() {
    return this._street;
}

@Override
public String toString() {
    return "House{" +
            "id: " + _id +
            ", number: " + _number +
            ", square: " + _square +
            ", floor: " + _floor +
            ", room_count: " + _room_count +
            ", lifetime: " + _lifetime +
            ", street: " + _street +
            ", building_type: " + _building_type +
            '}';
}
```

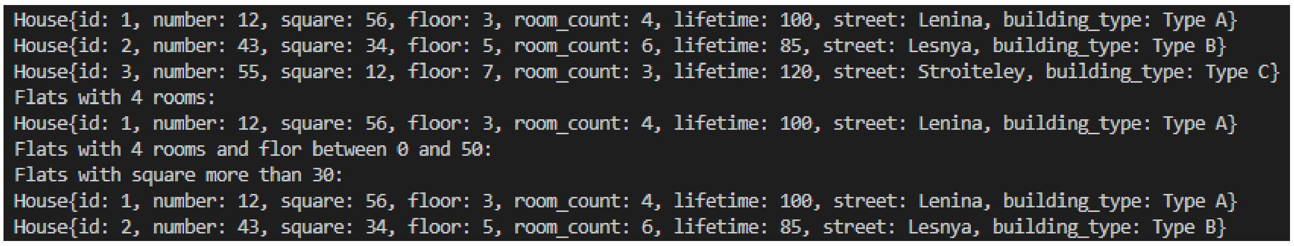


```
private static void houses() {
    ArrayList<task2_House> houses = new ArrayList<task2_House>();
    houses.add(new task2_House(
        id: 1,
        number: 12,
        square: 56,
        floor: 3,
        room_count: 4,
        street: "Lenina",
        building_type: "Type A",
        lifetime: 100
    ));
    houses.add(new task2_House(
        id: 2,
        number: 43,
        square: 34,
        floor: 5,
        room_count: 6,
        street: "Lesnya",
        building_type: "Type B",
        lifetime: 85
    ));
    houses.add(new task2_House(
        id: 3,
        number: 55,
        square: 12,
        floor: 7,
        room_count: 3,
        street: "Stroiteley",
        building_type: "Type C",
        lifetime: 120
    ));
    System.out.println(houses.get(index: 0));
    System.out.println(houses.get(index: 1));
    System.out.println(houses.get(index: 2));

    System.out.println(x: "Flats with 4 rooms:");
    findFlatByRooms(roomsNum: 4, houses);
    System.out.println(x: "Flats with 4 rooms and flor between 0 and 50:");
    findFlatByRoomsAndRange(roomsNum: 4, fs: 0, fe: 50, houses);
    System.out.println(x: "Flats with square more than 30:");
    findFlatBySq(minsq: 30, houses);
}
```



```java
private static void findFlatByRooms(int roomsNum, ArrayList<task2_House> houses) {
    for (task2_House h : houses) {
        if (h.get_Room_count() == roomsNum) {
            System.out.println(h);
        }
    }
}

private static void findFlatByRoomsAndRange(int roomsNum, int fs, int fe, ArrayList<task2_House> houses) {
    for (task2_House h : houses) {
        if (h.get_Room_count() == roomsNum && h.get_Floor() >= fs && h.get_Floor() < fs) {
            System.out.println(h);
        }
    }
}

private static void findFlatBySq(int minsq, ArrayList<task2_House> houses) {
    for (task2_House h : houses) {
        if (h.get_Square() > minsq) {
            System.out.println(h);
        }
    }
}
```

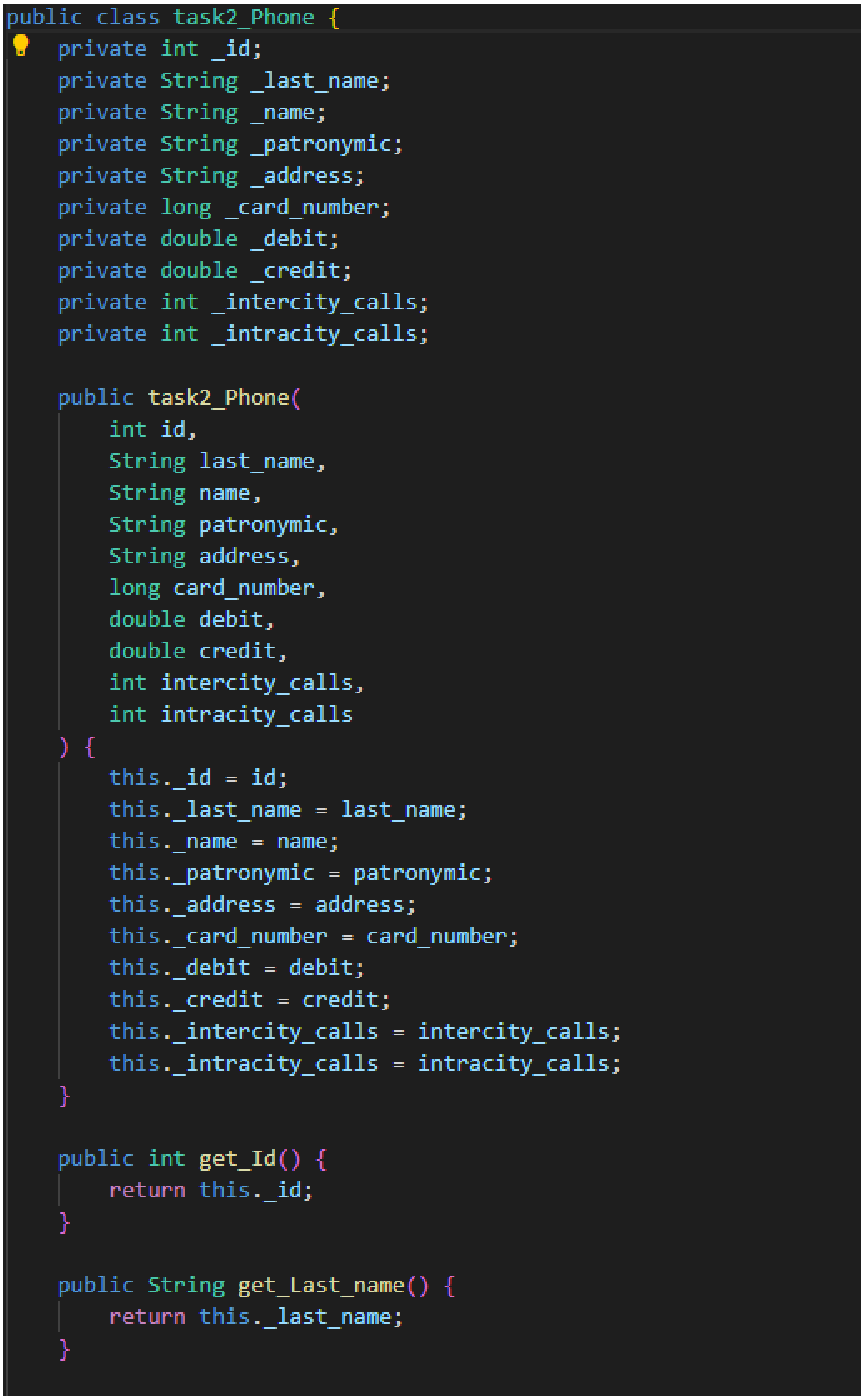
Процесс работы программы:



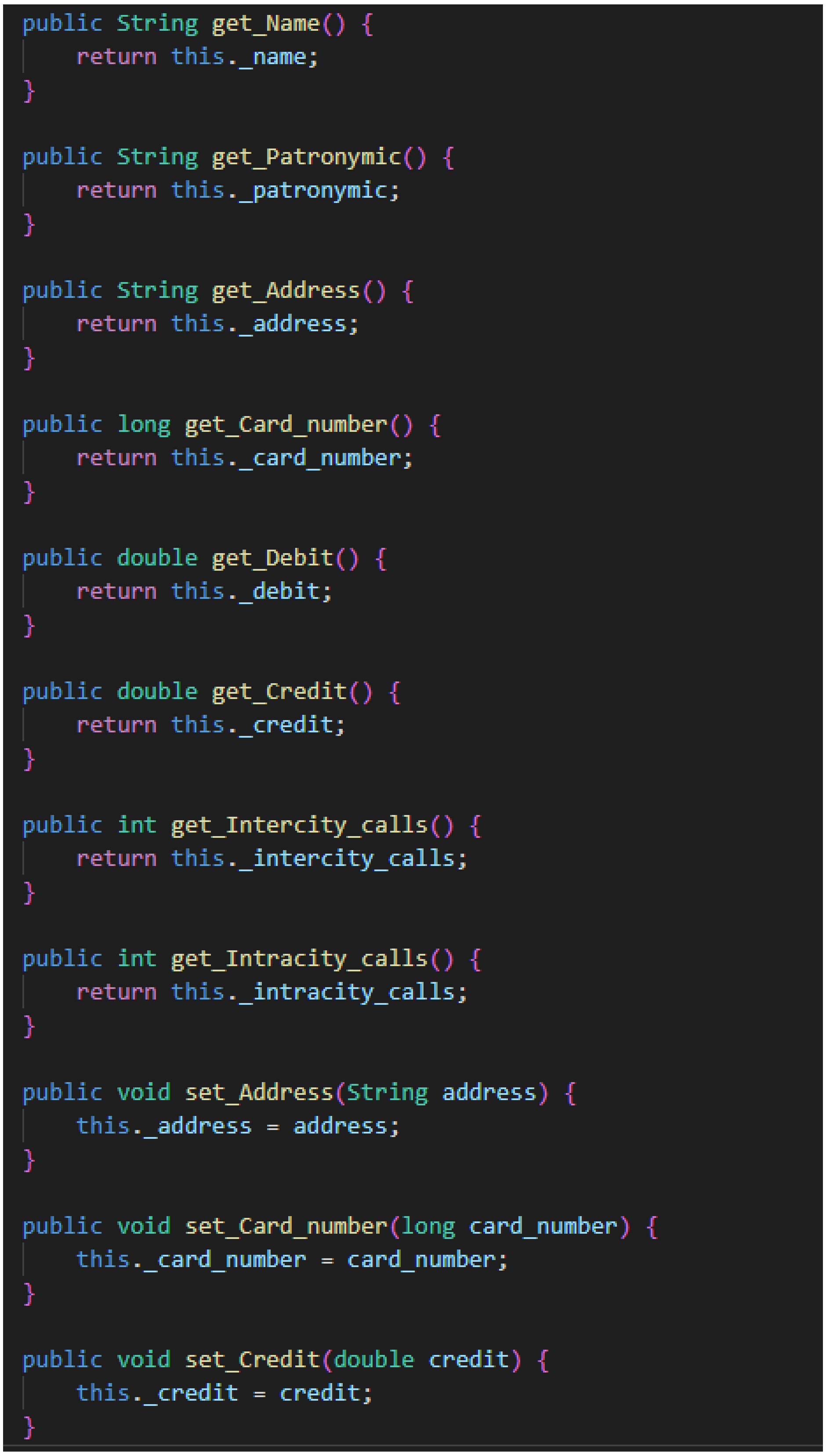
```
House{id: 1, number: 12, square: 56, floor: 3, room_count: 4, lifetime: 100, street: Lenina, building_type: Type A}
House{id: 2, number: 43, square: 34, floor: 5, room_count: 6, lifetime: 85, street: Lesnya, building_type: Type B}
House{id: 3, number: 55, square: 12, floor: 7, room_count: 3, lifetime: 120, street: Stroiteley, building_type: Type C}
Flats with 4 rooms:
House{id: 1, number: 12, square: 56, floor: 3, room_count: 4, lifetime: 100, street: Lenina, building_type: Type A}
Flats with 4 rooms and flor between 0 and 50:
Flats with square more than 30:
House{id: 1, number: 12, square: 56, floor: 3, room_count: 4, lifetime: 100, street: Lenina, building_type: Type A}
House{id: 2, number: 43, square: 34, floor: 5, room_count: 6, lifetime: 85, street: Lesnya, building_type: Type B}
```

Подзадача 2.

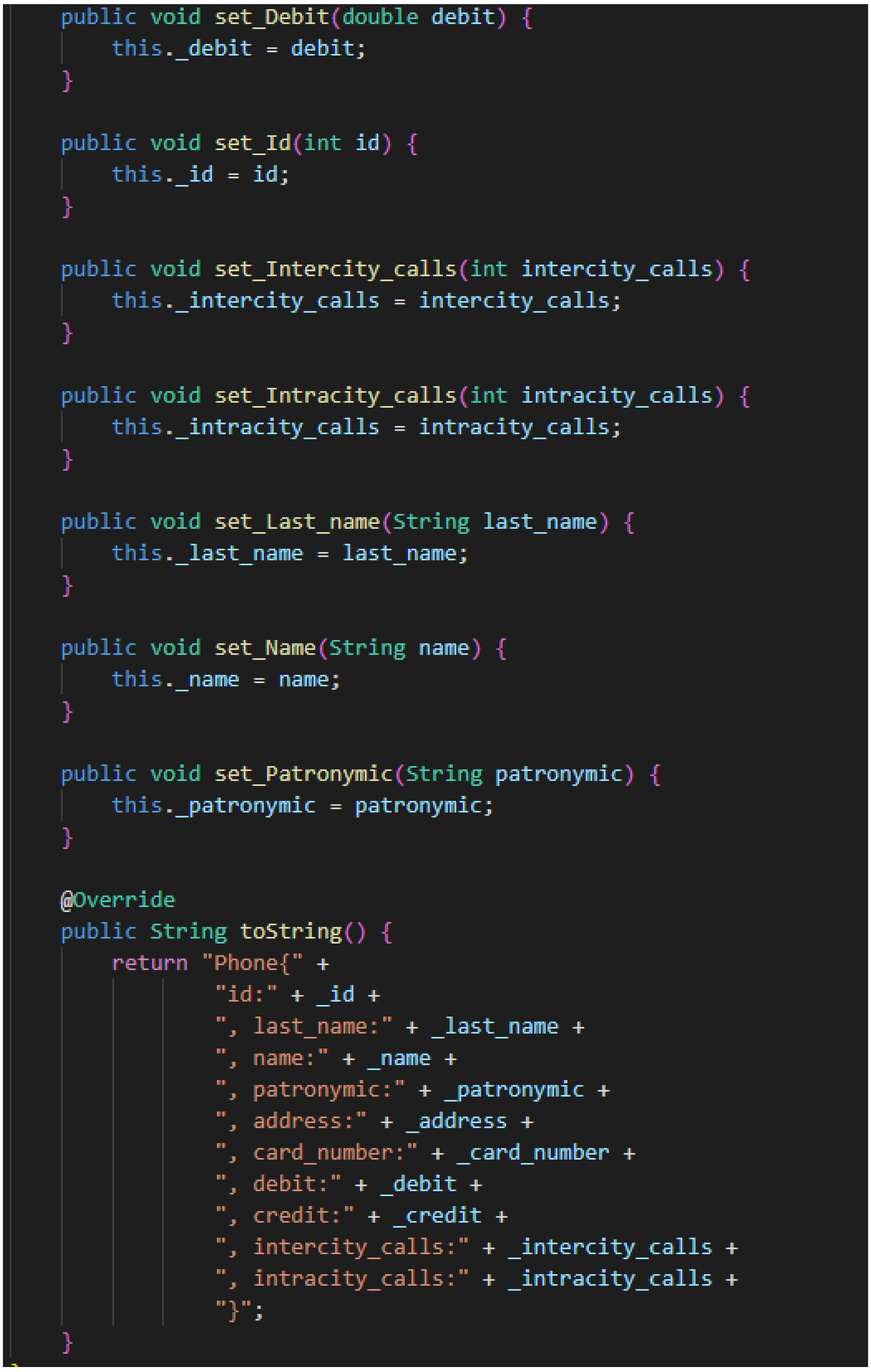
Код программы:



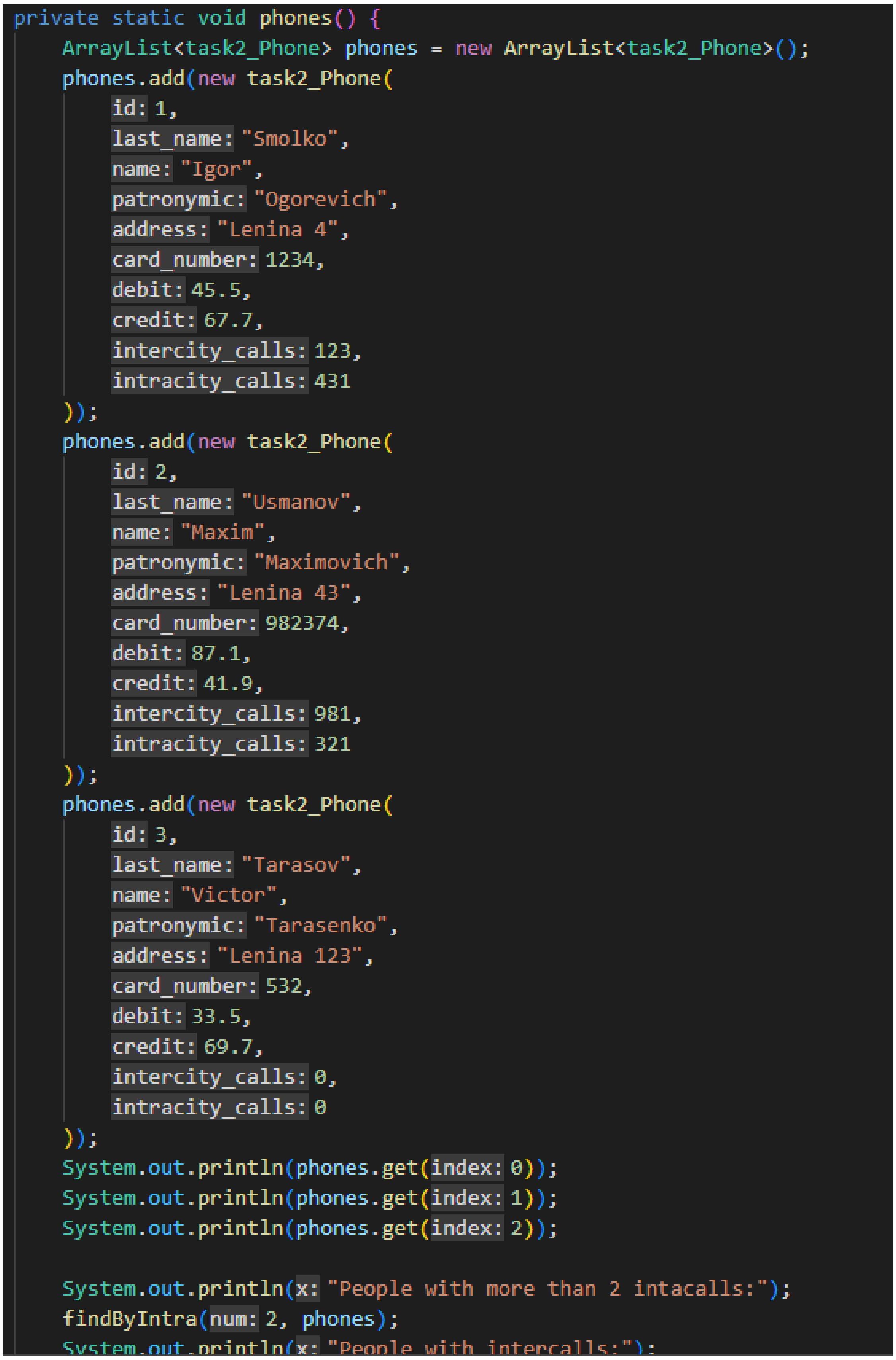
```java
public class task2_Phone {
    private int _id;
    private String _last_name;
    private String _name;
    private String _patronymic;
    private String _address;
    private long _card_number;
    private double _debit;
    private double _credit;
    private int _intercity_calls;
    private int _intracity_calls;

    public task2_Phone(
        int id,
        String last_name,
        String name,
        String patronymic,
        String address,
        long card_number,
        double debit,
        double credit,
        int intercity_calls,
        int intracity_calls
    ) {
        this._id = id;
        this._last_name = last_name;
        this._name = name;
        this._patronymic = patronymic;
        this._address = address;
        this._card_number = card_number;
        this._debit = debit;
        this._credit = credit;
        this._intercity_calls = intercity_calls;
        this._intracity_calls = intracity_calls;
    }

    public int get_Id() {
        return this._id;
    }

    public String get_Last_name() {
        return this._last_name;
    }
```



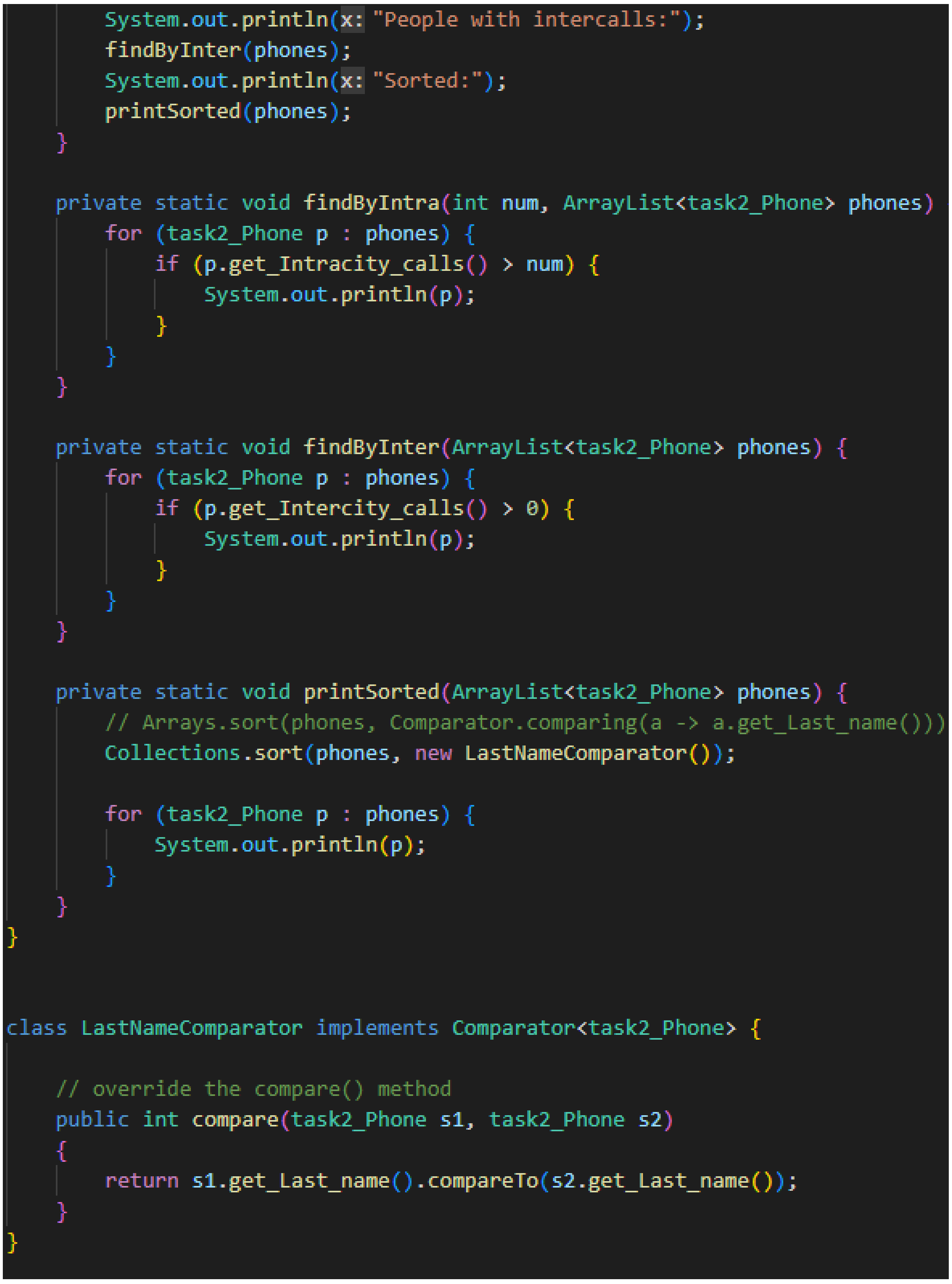
```
public String get_Name() {
    return this._name;
}

public String get_Patronymic() {
    return this._patronymic;
}

public String get_Address() {
    return this._address;
}

public long get_Card_number() {
    return this._card_number;
}

public double get_Debit() {
    return this._debit;
}

public double get_Credit() {
    return this._credit;
}

public int get_Intercity_calls() {
    return this._intercity_calls;
}

public int get_Intracity_calls() {
    return this._intracity_calls;
}

public void set_Address(String address) {
    this._address = address;
}

public void set_Card_number(long card_number) {
    this._card_number = card_number;
}

public void set_Credit(double credit) {
    this._credit = credit;
}
```



```
    public void set_Debit(double debit) {
        this._debit = debit;
    }

    public void set_Id(int id) {
        this._id = id;
    }

    public void set_Intercity_calls(int intercity_calls) {
        this._intercity_calls = intercity_calls;
    }

    public void set_Intracity_calls(int intracity_calls) {
        this._intracity_calls = intracity_calls;
    }

    public void set_Last_name(String last_name) {
        this._last_name = last_name;
    }

    public void set_Name(String name) {
        this._name = name;
    }

    public void set_Patronymic(String patronymic) {
        this._patronymic = patronymic;
    }

    @Override
    public String toString() {
        return "Phone{" +
                "id:" + _id +
                ", last_name:" + _last_name +
                ", name:" + _name +
                ", patronymic:" + _patronymic +
                ", address:" + _address +
                ", card_number:" + _card_number +
                ", debit:" + _debit +
                ", credit:" + _credit +
                ", intercity_calls:" + _intercity_calls +
                ", intracity_calls:" + _intracity_calls +
                "}";
    }
```

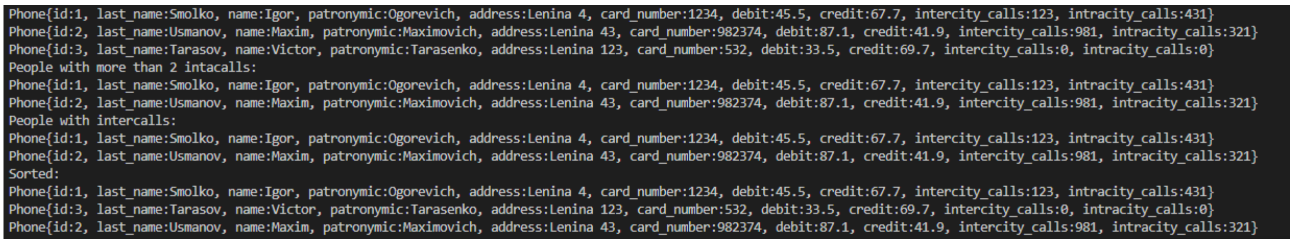


```
private static void phones() {
    ArrayList<task2_Phone> phones = new ArrayList<task2_Phone>();
    phones.add(new task2_Phone(
        id: 1,
        last_name: "Smolko",
        name: "Igor",
        patronymic: "Ogorevich",
        address: "Lenina 4",
        card_number: 1234,
        debit: 45.5,
        credit: 67.7,
        intercity_calls: 123,
        intracity_calls: 431
    ));
    phones.add(new task2_Phone(
        id: 2,
        last_name: "Usmanov",
        name: "Maxim",
        patronymic: "Maximovich",
        address: "Lenina 43",
        card_number: 982374,
        debit: 87.1,
        credit: 41.9,
        intercity_calls: 981,
        intracity_calls: 321
    ));
    phones.add(new task2_Phone(
        id: 3,
        last_name: "Tarasov",
        name: "Victor",
        patronymic: "Tarasenko",
        address: "Lenina 123",
        card_number: 532,
        debit: 33.5,
        credit: 69.7,
        intercity_calls: 0,
        intracity_calls: 0
    ));
    System.out.println(phones.get(index: 0));
    System.out.println(phones.get(index: 1));
    System.out.println(phones.get(index: 2));

    System.out.println(x: "People with more than 2 intacalls:");
    findByIntra(num: 2, phones);
    System.out.println(x: "People with intercalls:");
```



```java
        System.out.println(x: "People with intercalls:");
        findByInter(phones);
        System.out.println(x: "Sorted:");
        printSorted(phones);
    }

    private static void findByIntra(int num, ArrayList<task2_Phone> phones)
        for (task2_Phone p : phones) {
            if (p.get_Intracity_calls() > num) {
                System.out.println(p);
            }
        }
    }

    private static void findByInter(ArrayList<task2_Phone> phones) {
        for (task2_Phone p : phones) {
            if (p.get_Intercity_calls() > 0) {
                System.out.println(p);
            }
        }
    }

    private static void printSorted(ArrayList<task2_Phone> phones) {
        // Arrays.sort(phones, Comparator.comparing(a -> a.get_Last_name()))
        Collections.sort(phones, new LastNameComparator());

        for (task2_Phone p : phones) {
            System.out.println(p);
        }
    }
}


class LastNameComparator implements Comparator<task2_Phone> {

    // override the compare() method
    public int compare(task2_Phone s1, task2_Phone s2)
    {
        return s1.get_Last_name().compareTo(s2.get_Last_name());
    }
}
```

Процесс работы программы:



```
Phone{id:1, last_name:Smolko, name:Igor, patronymic:Ogorevich, address:Lenina 4, card_number:1234, debit:45.5, credit:67.7, intercity_calls:123, intracity_calls:431}
Phone{id:2, last_name:Usmanov, name:Maxim, patronymic:Maximovich, address:Lenina 43, card_number:982374, debit:87.1, credit:41.9, intercity_calls:981, intracity_calls:321}
Phone{id:3, last_name:Tarasov, name:Victor, patronymic:Tarasenko, address:Lenina 123, card_number:532, debit:33.5, credit:69.7, intercity_calls:0, intracity_calls:0}
People with more than 2 intacalls:
Phone{id:1, last_name:Smolko, name:Igor, patronymic:Ogorevich, address:Lenina 4, card_number:1234, debit:45.5, credit:67.7, intercity_calls:123, intracity_calls:431}
Phone{id:2, last_name:Usmanov, name:Maxim, patronymic:Maximovich, address:Lenina 43, card_number:982374, debit:87.1, credit:41.9, intercity_calls:981, intracity_calls:321}
People with intercalls:
Phone{id:1, last_name:Smolko, name:Igor, patronymic:Ogorevich, address:Lenina 4, card_number:1234, debit:45.5, credit:67.7, intercity_calls:123, intracity_calls:431}
Phone{id:2, last_name:Usmanov, name:Maxim, patronymic:Maximovich, address:Lenina 43, card_number:982374, debit:87.1, credit:41.9, intercity_calls:981, intracity_calls:321}
Sorted:
Phone{id:1, last_name:Smolko, name:Igor, patronymic:Ogorevich, address:Lenina 4, card_number:1234, debit:45.5, credit:67.7, intercity_calls:123, intracity_calls:431}
Phone{id:3, last_name:Tarasov, name:Victor, patronymic:Tarasenko, address:Lenina 123, card_number:532, debit:33.5, credit:69.7, intercity_calls:0, intracity_calls:0}
Phone{id:2, last_name:Usmanov, name:Maxim, patronymic:Maximovich, address:Lenina 43, card_number:982374, debit:87.1, credit:41.9, intercity_calls:981, intracity_calls:321}
```

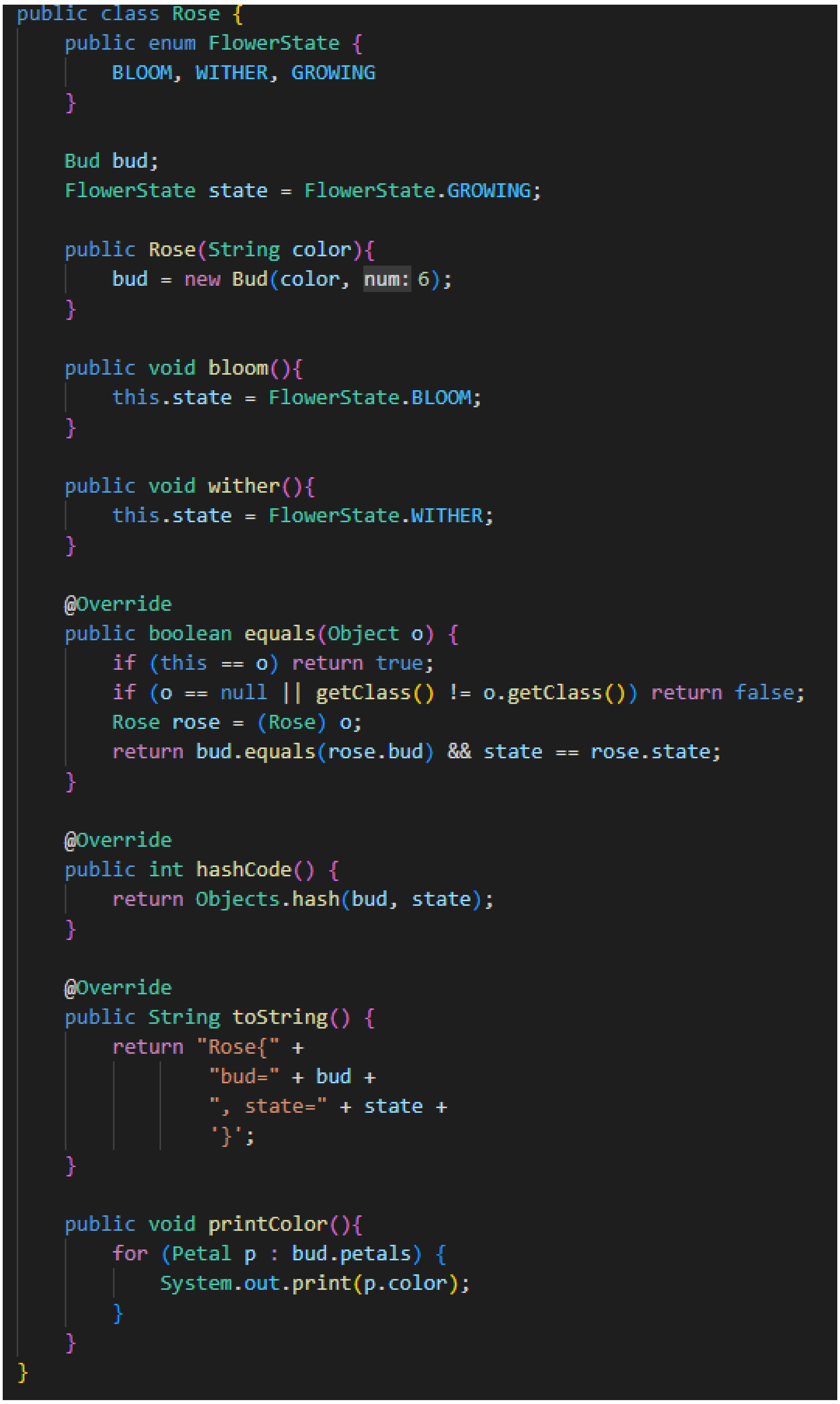
Задание 3:

6. Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завять, вывести на консоль цвет бутона.

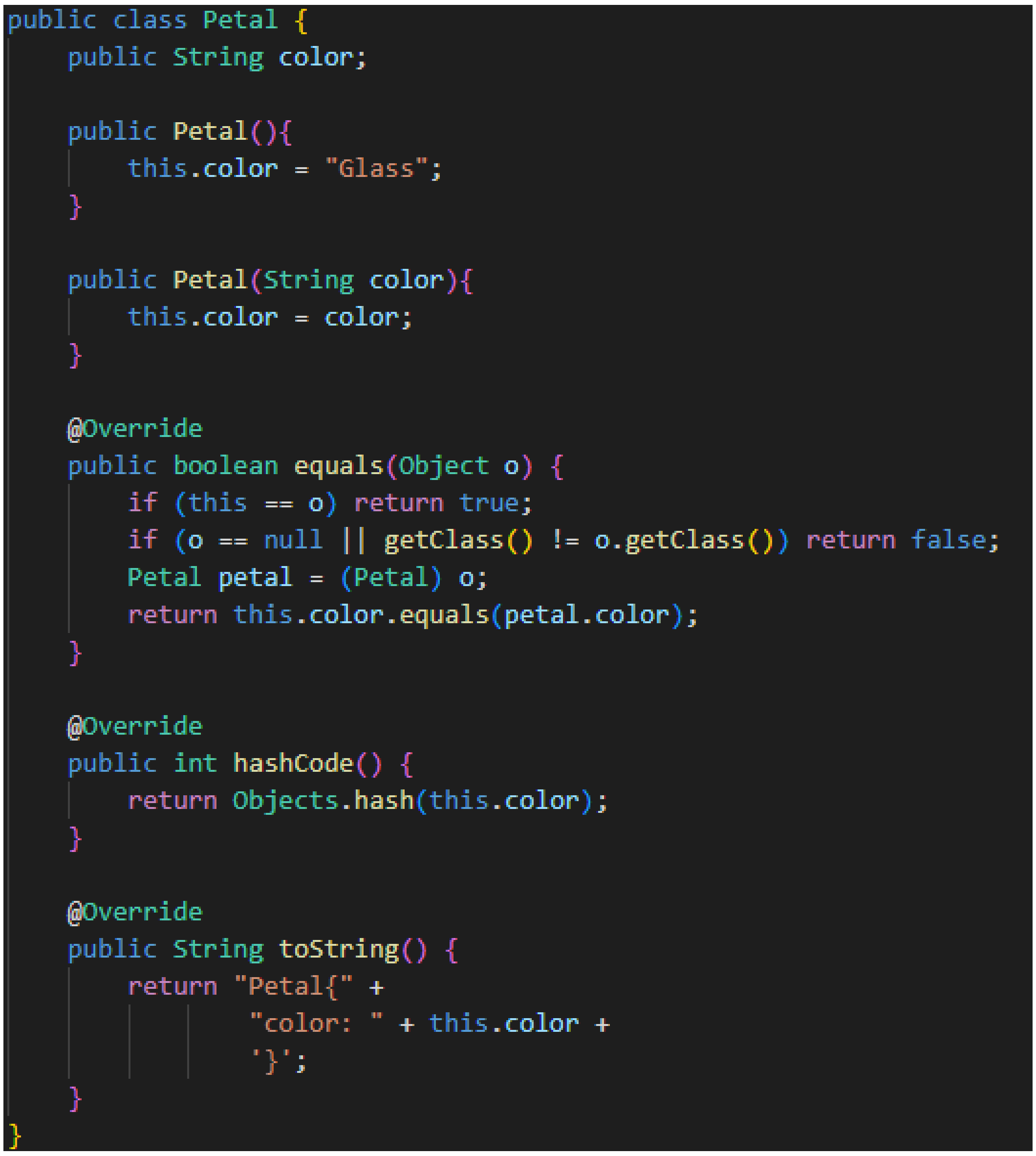
7. Создать объект класса Дерево, используя классы Лист. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.

Подзадача 1.

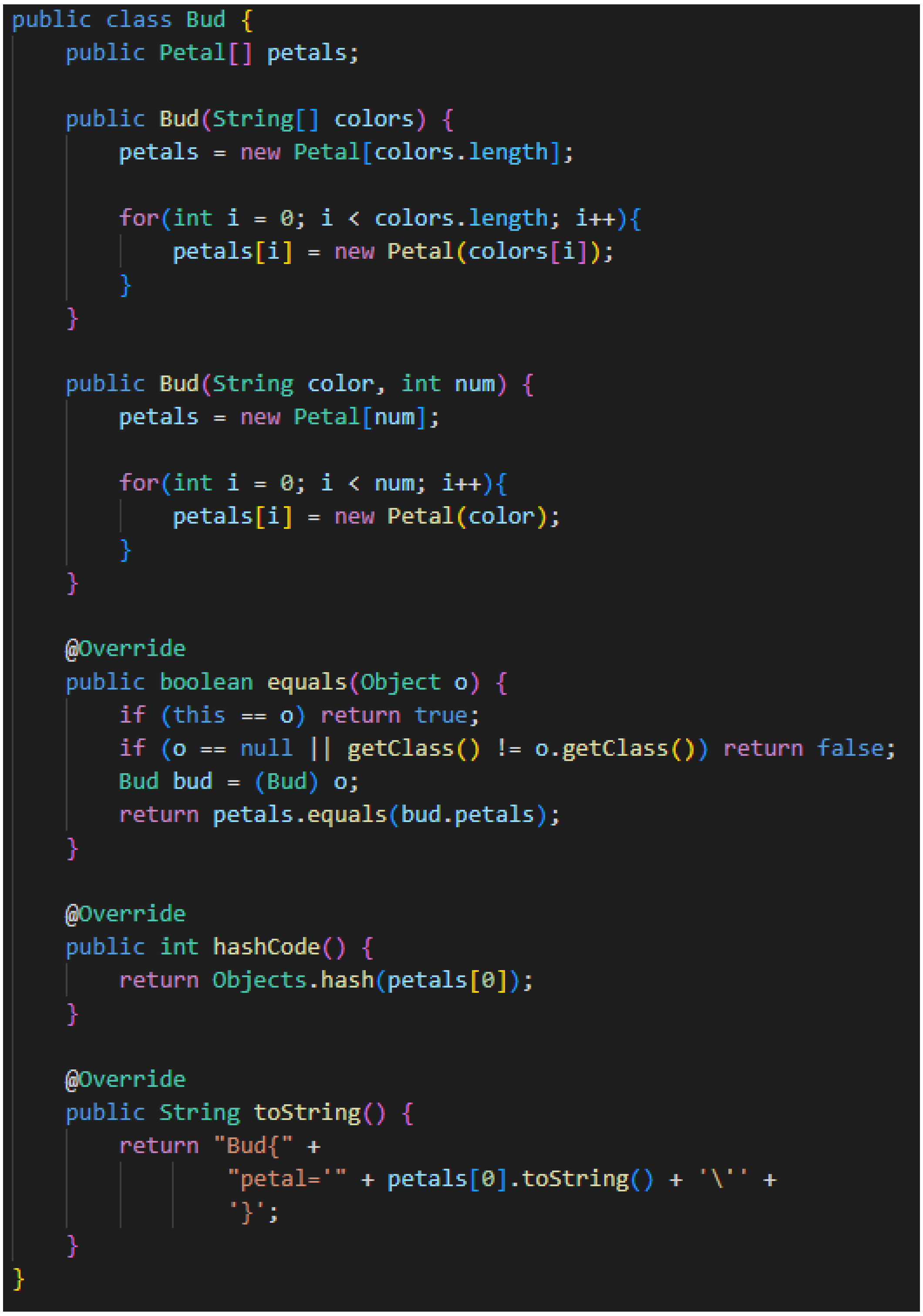
Код программы:



```java
public class Rose {
    public enum FlowerState {
        BLOOM, WITHER, GROWING
    }

    Bud bud;
    FlowerState state = FlowerState.GROWING;

    public Rose(String color){
        bud = new Bud(color, num: 6);
    }

    public void bloom(){
        this.state = FlowerState.BLOOM;
    }

    public void wither(){
        this.state = FlowerState.WITHER;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Rose rose = (Rose) o;
        return bud.equals(rose.bud) && state == rose.state;
    }

    @Override
    public int hashCode() {
        return Objects.hash(bud, state);
    }

    @Override
    public String toString() {
        return "Rose{" +
                "bud=" + bud +
                ", state=" + state +
                '}';
    }

    public void printColor(){
        for (Petal p : bud.petals) {
            System.out.print(p.color);
        }
    }
}
```



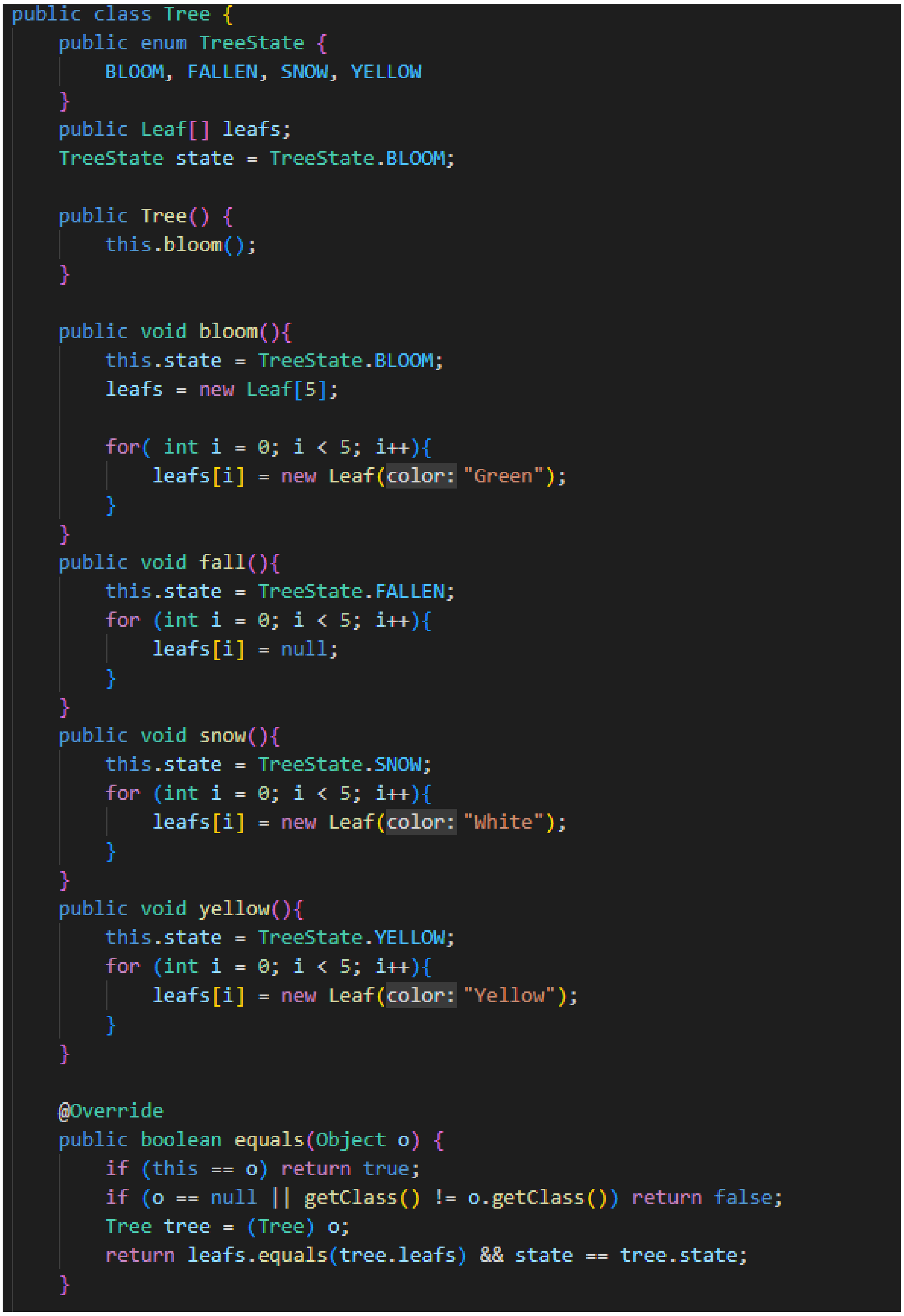
```java
public class Petal {
    public String color;

    public Petal(){
        this.color = "Glass";
    }

    public Petal(String color){
        this.color = color;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Petal petal = (Petal) o;
        return this.color.equals(petal.color);
    }

    @Override
    public int hashCode() {
        return Objects.hash(this.color);
    }

    @Override
    public String toString() {
        return "Petal{" +
                "color: " + this.color +
                '}';
    }
}
```



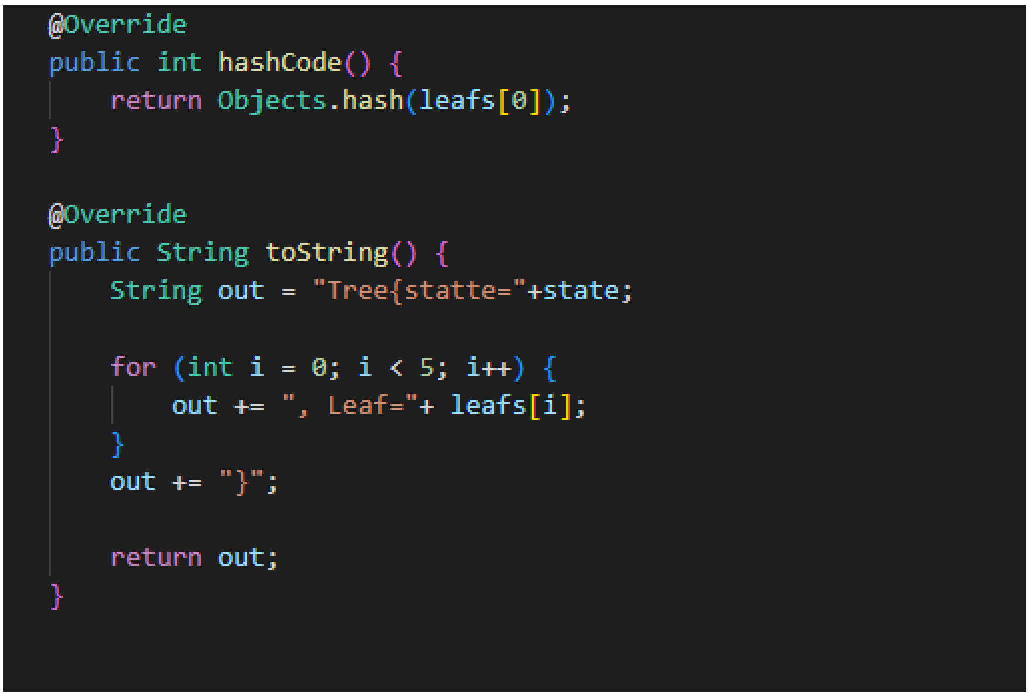
```java
public class Bud {
    public Petal[] petals;

    public Bud(String[] colors) {
        petals = new Petal[colors.length];

        for(int i = 0; i < colors.length; i++){
            petals[i] = new Petal(colors[i]);
        }
    }

    public Bud(String color, int num) {
        petals = new Petal[num];

        for(int i = 0; i < num; i++){
            petals[i] = new Petal(color);
        }
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Bud bud = (Bud) o;
        return petals.equals(bud.petals);
    }

    @Override
    public int hashCode() {
        return Objects.hash(petals[0]);
    }

    @Override
    public String toString() {
        return "Bud{" +
                "petal='" + petals[0].toString() + '\'' +
                '}';
    }
}
```

Подзадача 2.

Код программы:



```
public class Tree {
    public enum TreeState {
        BLOOM, FALLEN, SNOW, YELLOW
    }
    public Leaf[] leafs;
    TreeState state = TreeState.BLOOM;

    public Tree() {
        this.bloom();
    }

    public void bloom(){
        this.state = TreeState.BLOOM;
        leafs = new Leaf[5];

        for( int i = 0; i < 5; i++){
            leafs[i] = new Leaf(color: "Green");
        }
    }
    public void fall(){
        this.state = TreeState.FALLEN;
        for (int i = 0; i < 5; i++){
            leafs[i] = null;
        }
    }
    public void snow(){
        this.state = TreeState.SNOW;
        for (int i = 0; i < 5; i++){
            leafs[i] = new Leaf(color: "White");
        }
    }
    public void yellow(){
        this.state = TreeState.YELLOW;
        for (int i = 0; i < 5; i++){
            leafs[i] = new Leaf(color: "Yellow");
        }
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Tree tree = (Tree) o;
        return leafs.equals(tree.leafs) && state == tree.state;
    }
```



```
@Override
public int hashCode() {
    return Objects.hash(leafs[0]);
}

@Override
public String toString() {
    String out = "Tree{statte="+state;

    for (int i = 0; i < 5; i++) {
        out += ", Leaf="+ leafs[i];
    }
    out += "}";

    return out;
}
```

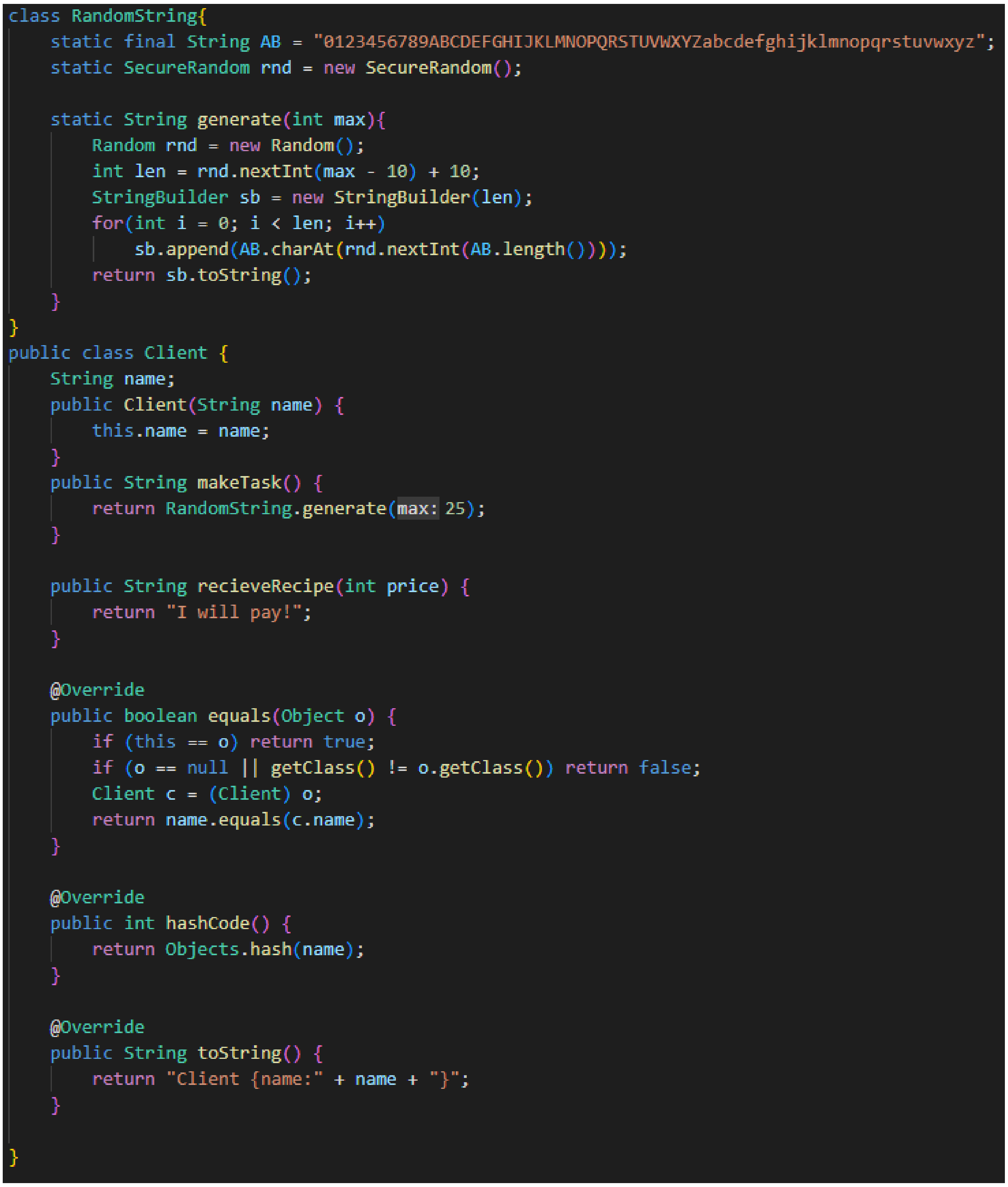
Задание 4:

6. Система Конструкторское бюро. Заказчик представляет Техническое Задание (ТЗ) на проектирование многоэтажного Дома. Конструктор регистрирует ТЗ, определяет стоимость проектирования и строительства, выставляет Заказчику Счет за проектирование и создает Бригаду Конструкторов для выполнения Проекта.

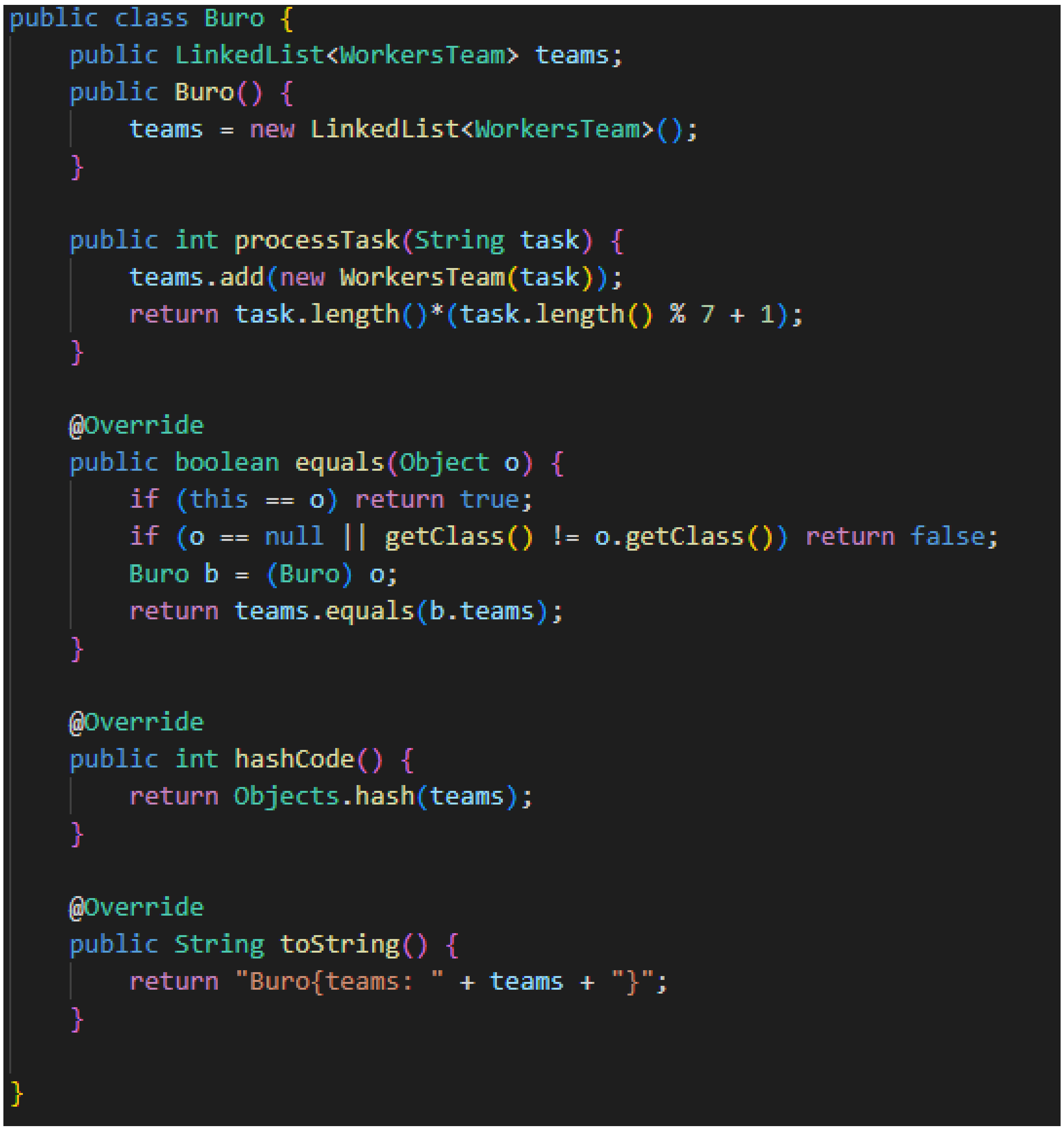
7. Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.

Подзадача 1.

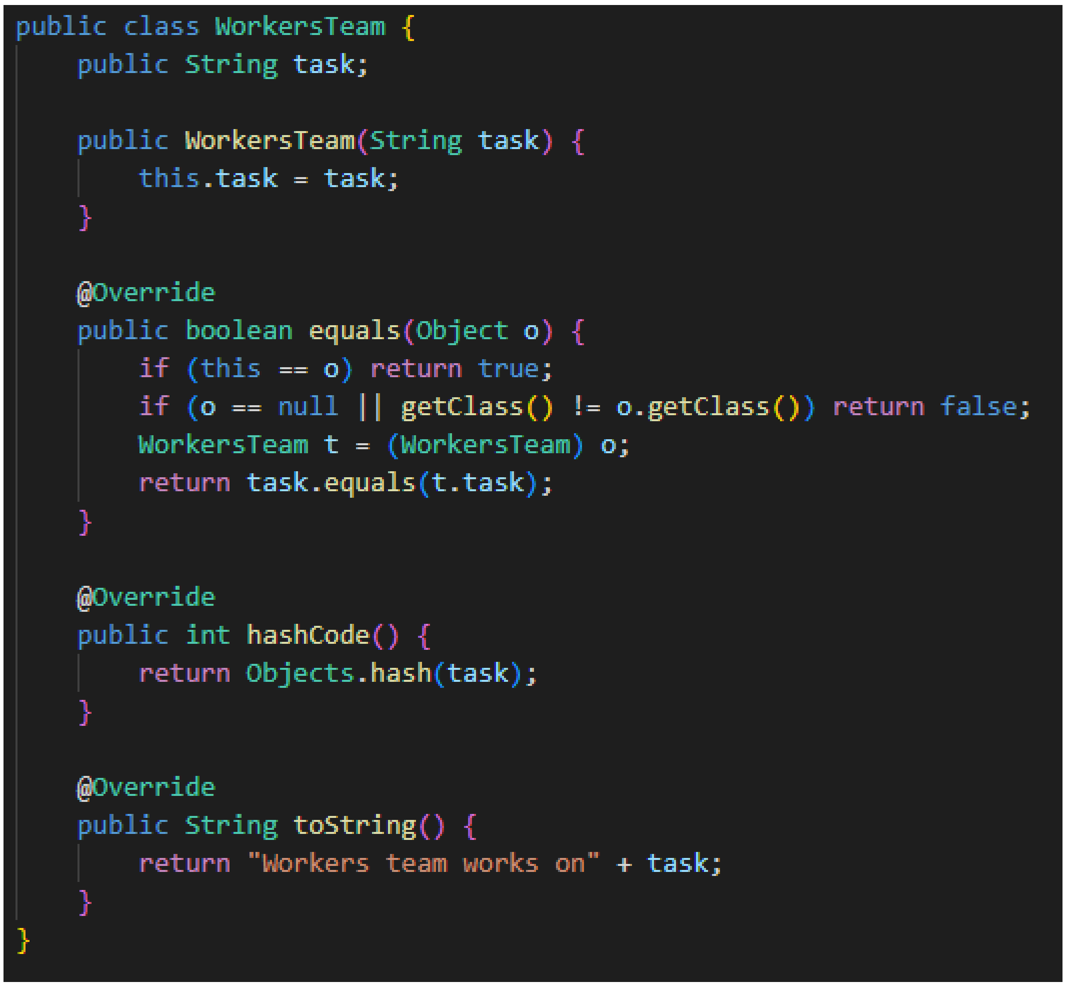
Код программы:



```java
class RandomString{
    static final String AB = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    static SecureRandom rnd = new SecureRandom();

    static String generate(int max){
        Random rnd = new Random();
        int len = rnd.nextInt(max - 10) + 10;
        StringBuilder sb = new StringBuilder(len);
        for(int i = 0; i < len; i++)
            sb.append(AB.charAt(rnd.nextInt(AB.length())));
        return sb.toString();
    }
}
public class Client {
    String name;
    public Client(String name) {
        this.name = name;
    }
    public String makeTask() {
        return RandomString.generate(max: 25);
    }

    public String recieveRecipe(int price) {
        return "I will pay!";
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Client c = (Client) o;
        return name.equals(c.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name);
    }

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

}
```

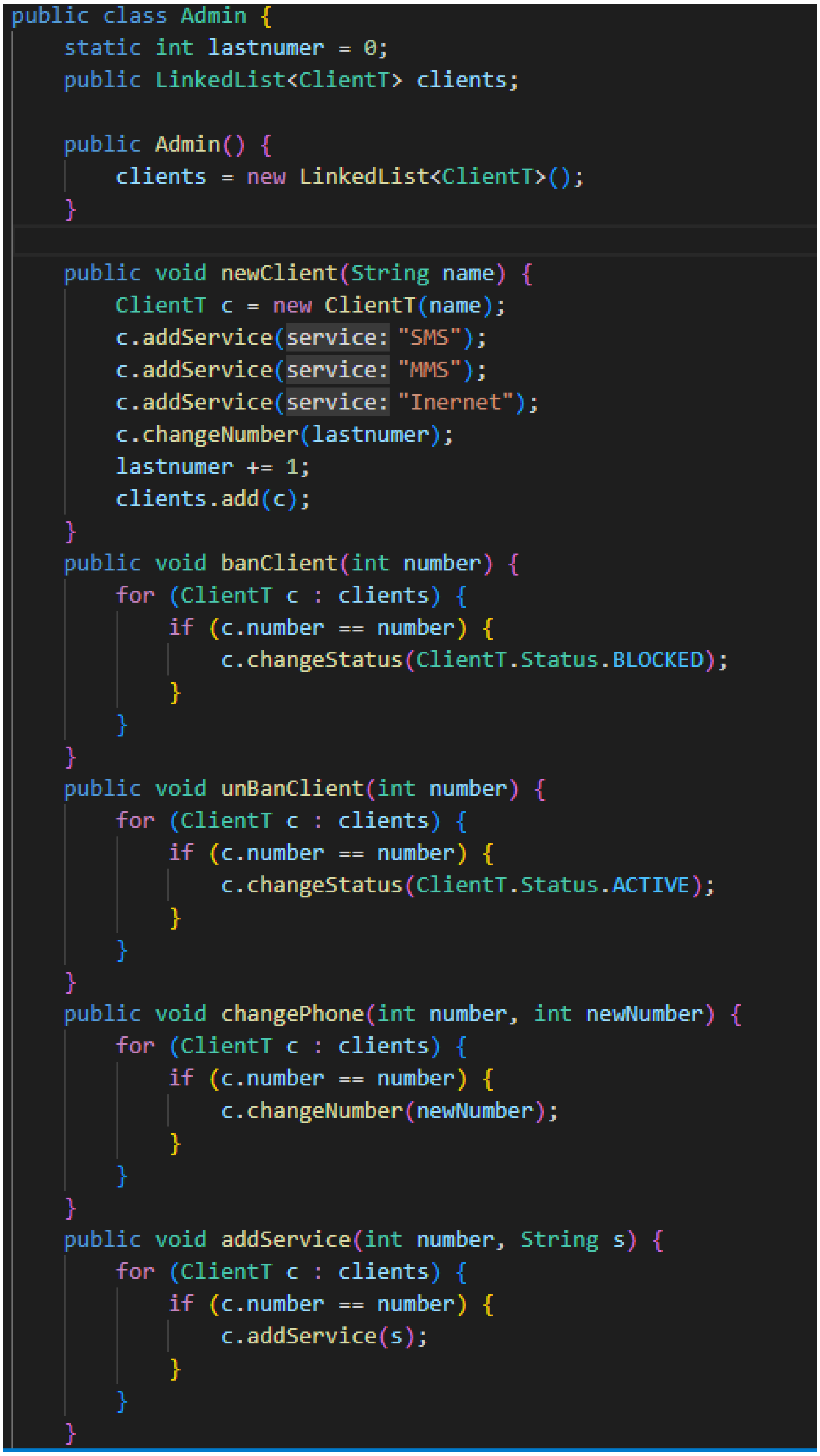


```
public class Buro {
    public LinkedList<WorkersTeam> teams;
    public Buro() {
        teams = new LinkedList<WorkersTeam>();
    }

    public int processTask(String task) {
        teams.add(new WorkersTeam(task));
        return task.length()*(task.length() % 7 + 1);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Buro b = (Buro) o;
        return teams.equals(b.teams);
    }

    @Override
    public int hashCode() {
        return Objects.hash(teams);
    }

    @Override
    public String toString() {
        return "Buro{teams: " + teams + "}";
    }

}
```

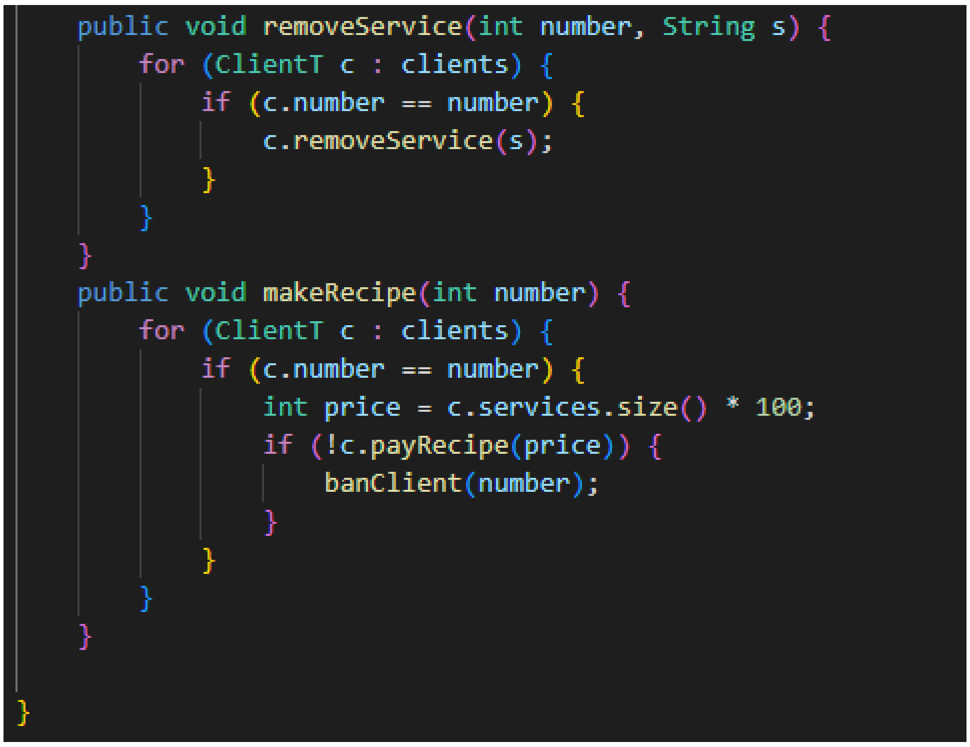

```java
public class WorkersTeam {
    public String task;

    public WorkersTeam(String task) {
        this.task = task;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        WorkersTeam t = (WorkersTeam) o;
        return task.equals(t.task);
    }

    @Override
    public int hashCode() {
        return Objects.hash(task);
    }

    @Override
    public String toString() {
        return "Workers team works on" + task;
    }
}
```

Подзадача 2.

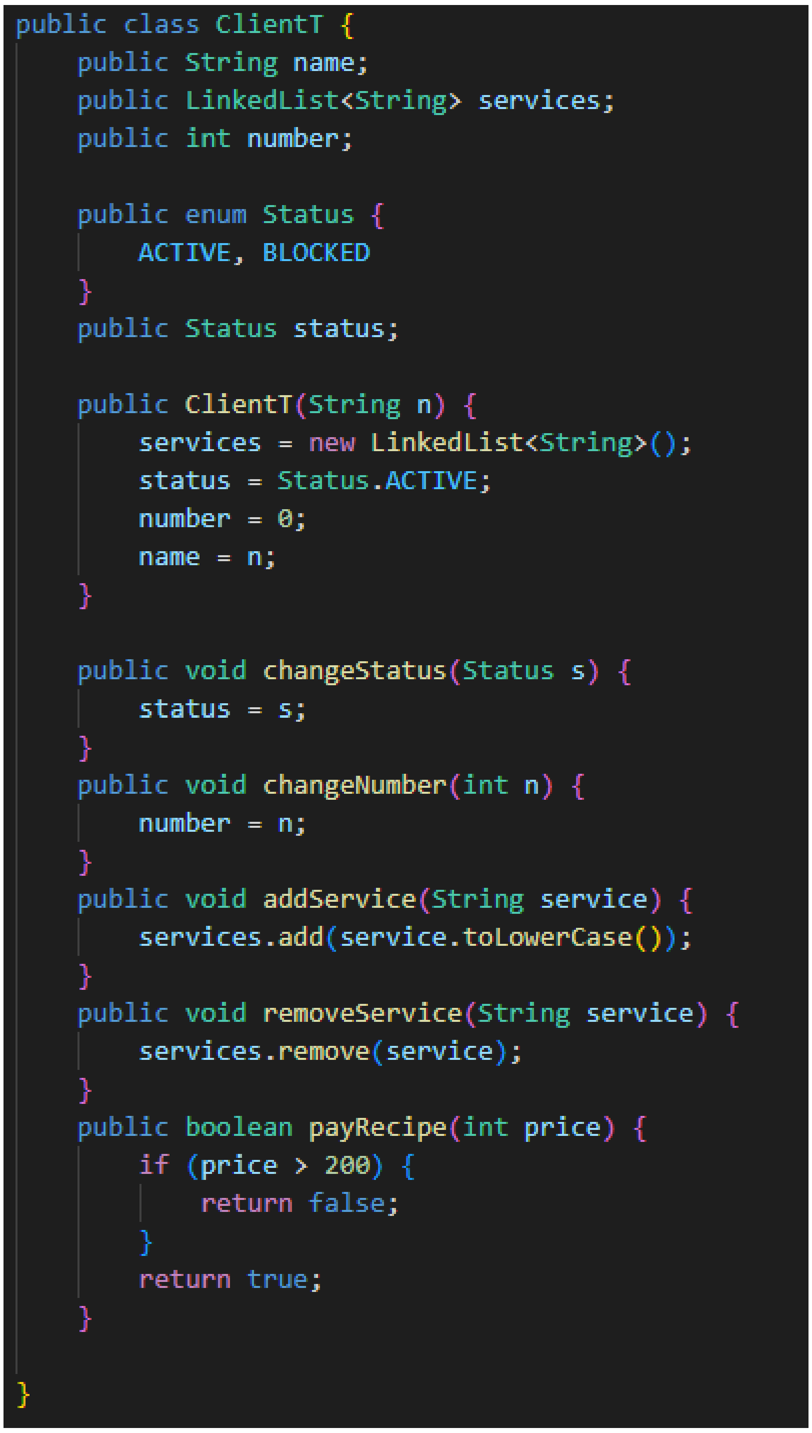
Код программы:



```java
public class Admin {
    static int lastnumer = 0;
    public LinkedList<ClientT> clients;

    public Admin() {
        clients = new LinkedList<ClientT>();
    }

    public void newClient(String name) {
        ClientT c = new ClientT(name);
        c.addService(service: "SMS");
        c.addService(service: "MMS");
        c.addService(service: "Inernet");
        c.changeNumber(lastnumer);
        lastnumer += 1;
        clients.add(c);
    }
    public void banClient(int number) {
        for (ClientT c : clients) {
            if (c.number == number) {
                c.changeStatus(ClientT.Status.BLOCKED);
            }
        }
    }
    public void unBanClient(int number) {
        for (ClientT c : clients) {
            if (c.number == number) {
                c.changeStatus(ClientT.Status.ACTIVE);
            }
        }
    }
    public void changePhone(int number, int newNumber) {
        for (ClientT c : clients) {
            if (c.number == number) {
                c.changeNumber(newNumber);
            }
        }
    }
    public void addService(int number, String s) {
        for (ClientT c : clients) {
            if (c.number == number) {
                c.addService(s);
            }
        }
    }
```



```
    public void removeService(int number, String s) {
        for (ClientT c : clients) {
            if (c.number == number) {
                c.removeService(s);
            }
        }
    }
    public void makeRecipe(int number) {
        for (ClientT c : clients) {
            if (c.number == number) {
                int price = c.services.size() * 100;
                if (!c.payRecipe(price)) {
                    banClient(number);
                }
            }
        }
    }

}
```



```java
public class ClientT {
    public String name;
    public LinkedList<String> services;
    public int number;

    public enum Status {
        ACTIVE, BLOCKED
    }
    public Status status;

    public ClientT(String n) {
        services = new LinkedList<String>();
        status = Status.ACTIVE;
        number = 0;
        name = n;
    }

    public void changeStatus(Status s) {
        status = s;
    }
    public void changeNumber(int n) {
        number = n;
    }
    public void addService(String service) {
        services.add(service.toLowerCase());
    }
    public void removeService(String service) {
        services.remove(service);
    }
    public boolean payRecipe(int price) {
        if (price > 200) {
            return false;
        }
        return true;
    }

}
```

Ссылка на программное решение:

https://github.com/ArMaxik/BigDataLanguages/tree/main/lr3

Вывод: в ходе лабораторной работы были получены навыки работы с классами Java, были исследованы механизмы наследования и полиморфизма языка программирования Java.