



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

## ОТЧЕТ

по лабораторной работе № 3

Название: Классы, наследование, полиморфизм

Дисциплина: Языки программирования для работы с большими  
данными

Студент ИУ6-22М

Д. Р. Григорян

Преподаватель

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

Москва, 2023

## Цель работы:

Изучение понятия класса в объектно-ориентированном программировании, изучение структуры класса, изучение различий в статических и динамических вызовах методов

## Выполнение:

### Задача 1.1:

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.

### Листинг 1 программы main:

```
import java.util.Scanner;

//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.
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int nums[] = new int[4];
        System.out.println("Input one at the time int numbers: ");
        for (int i = 0; i < nums.length; i++) {
            nums[i] = in.nextInt();
        }
        Fraction frac1 = new Fraction(nums[0],nums[1]); // вызов второго
конструктора с одним параметром
        Fraction frac2 = new Fraction(nums[2],nums[3]);
        frac1.sum(frac1,frac2);
        frac1.sub(frac1,frac2);
        frac1.mul(frac1,frac2);
        frac1.div(frac1,frac2);

        int num = 0;
        while (true) {
            System.out.print("Input a number of Fractions for array: ");
            num = in.nextInt();
            if (num > 0) {
                break;
            }
        }
        int nums1[] = new int[num*2];
        System.out.println("Input one at the time int numbers: ");
```

```

        for (int i = 0; i < nums1.length; i++) {
            nums1[i] = in.nextInt();
        }
        Fraction[] arr = new Fraction[num];
        int k=0;
        System.out.println("The input array of fractions:");
        for (int i=0;i<nums1.length-1;i+=2) {
            int numerator = nums1[i];
            int determinator = nums1[i+1];
            arr[k] = new Fraction(numerator,determinator);
            System.out.println(arr[k].m);
            System.out.println(arr[k].n);
            System.out.println("---");
            k++;
        }
        Fraction frac3 = new Fraction();
        frac3.edit(arr);
        System.out.println("The edit array of fractions:");
        for (int i=0;i<arr.length;i++){
            System.out.println(arr[i].m);
            System.out.println(arr[i].n);
            System.out.println("---");
        }
    }
}

```

## Листинг 2 класса Fraction:

```

class Fraction{    // имя

    int m,n; // числитель и знаменатель
    public Fraction() {}

    public Fraction(int m, int n) {
        this.m = m;
        this.n = n;
    }

    private static int gcd(int a, int b) // Greatest Common Divisor
    {
        while (b > 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }

    private static int gcd(int[] input) { // Greatest Common Divisor
        int result = input[0];
        for (int i = 1; i < input.length; i++)
            result = gcd(result, input[i]);
        return result;
    }

    private static int lcm(int a,int b){
        return a / gcd(a,b) * b;
    }

    private static int lcm(int[] input) {
        int result = input[0];
        for (int i = 1; i < input.length; i++) {
            if ((input[i] == 0 || input[0] == 0)

```

```

        || (input[i] < 0 || input[0] < 0))
            break;
        result = lcm(result, input[i]);
    }
    return result;
}

void displayInfo(Fraction res,String operation){
    System.out.println(res.m);
    System.out.print(res.n);
    System.out.println(operation);
}

public Fraction sum(Fraction first, Fraction second){
    Fraction result = new Fraction();
    if (first.n==second.n) {
        result.m = first.m + second.m;
        result.n = first.n;
    }
    else {
        int LCM = lcm(first.n,second.n);
        result.m =first.m*(LCM/ first.n)+second.m*(LCM / second.n);
        result.n = LCM;
    }
    displayInfo(result," Sum");
    return result;
}

public void edit(Fraction[] array_for_edit){
    for (int i=0;i<array_for_edit.length-1;i++) {
        if (i % 2 == 0) {
            array_for_edit[i] =
sum(array_for_edit[i],array_for_edit[i+1]);
        }
    }
    //displayInfo(result," Sum");
}

public void sub(Fraction first, Fraction second){
    Fraction result = new Fraction();
    if (first.n==second.n) {
        result.m = first.m - second.m;
        result.n = first.n;
    }
    else {
        int LCM = lcm(first.n,second.n);
        result.m =first.m*(LCM/ first.n)-second.m*(LCM / second.n);
        result.n = LCM;
    }
    displayInfo(result," Sub");
}

public void mul(Fraction first, Fraction second){
    Fraction result = new Fraction();
    result.n= first.n* second.n;
    result.m= first.m* second.m;
    displayInfo(result," Mul/Div");
}

public void div(Fraction first, Fraction second){
    Fraction doubler = new Fraction();
    doubler.m = second.n;
    doubler.n = second.m;
    mul(first,doubler);
}
}

```

Результаты приведены на рисунках 1-2:

```
Input one at the time int numbers:
1
2
5
6
8
6 Sum
-2
6 Sub
5
12 Mul/Div
6
10 Mul/Div
```

Рисунок 1 – Результат выполнения программы

```
Input a number of Fractions for array: 2
Input one at the time int numbers:
1
2
5
6
The input array of fractions:
1
2
---
5
6
---
8
6 Sum
The edit array of fractions:
8
6
---
5
6
```

Рисунок 2 – Результат выполнения добавления в массив и замены дроби

### Задача 1.2:

Определить класс Комплекс. Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения, деления, присваивания комплексных чисел. Создать два вектора размерности  $n$  из комплексных координат. Передать их в метод, который выполнит их сложение.

### Листинг 3 программы main:

```
public class Main {
    public static void main(String[] args) {
```

```

        int n = 3;
        Complex[] vector1 = new Complex[n];
        Complex[] vector2 = new Complex[n];
        for (int i = 0; i < n; i++) {
            vector1[i] = new Complex(i, i+1);
            vector2[i] = new Complex(i + 1, i);
        }
        Complex[] result = new Complex[n];
        for (int i = 0; i < n; i++) {
            result[i] = vector1[i].add(vector2[i]);
        }
        System.out.println("Vector 1: ");
        for (int i = 0; i < n; i++) {
            System.out.println(vector1[i].toString());
        }
        System.out.println("Vector 2: ");
        for (int i = 0; i < n; i++) {
            System.out.println(vector2[i].toString());
        }
        System.out.println("Result: ");
        for (int i = 0; i < n; i++) {
            System.out.println(result[i].toString());
        }
    }
}

```

#### Листинг 4 класса Complex:

```

class Complex {
    private double real;
    private double imaginary;
    public Complex() {
        this.real = 0;
        this.imaginary = 0;
    }
    public Complex(double real1, double imaginary1) {
        this.real = real1;
        this.imaginary = imaginary1;
    }
    public Complex add(Complex other) {
        return new Complex(this.real + other.real, this.imaginary +
other.imaginary);
    }
    public Complex subtract(Complex other) {
        return new Complex(this.real - other.real, this.imaginary -
other.imaginary);
    }
    public Complex multiply(Complex other) {
        return new Complex(this.real * other.real - this.imaginary *
other.imaginary,
            this.real * other.imaginary + this.imaginary * other.real);
    }
    public Complex divide(Complex other) {
        double denominator = other.real * other.real + other.imaginary *
other.imaginary;
        return new Complex((this.real * other.real + this.imaginary *
other.imaginary) / denominator,
            (this.imaginary * other.real - this.real * other.imaginary) /
denominator);
    }
    public void setReal(double real) {
        this.real = real;
    }
}

```

```

    }
    public void setImaginary(double imaginary) {
        this.imaginary = imaginary;
    }
    public double getReal() {
        return this.real;
    }
    public double getImaginary() {
        return this.imaginary;
    }
    public String toString() {
        if (this.imaginary < 0) {
            return this.real + " - " + Math.abs(this.imaginary) + "i";
        } else {
            return this.real + " + " + this.imaginary + "i";
        }
    }
}

```

Результаты приведены на рисунках 3-4:

```

Vector 1:
0.0 + 1.0i
1.0 + 2.0i
2.0 + 3.0i
Vector 2:
1.0 + 0.0i
2.0 + 1.0i
3.0 + 2.0i
Result:
1.0 + 1.0i
3.0 + 3.0i
5.0 + 5.0i

```

Рисунок 3 – Результат выполнения программы

```

Vector 1:
0.0 + 2.0i
1.0 + 3.0i
2.0 + 4.0i
Vector 2:
2.0 + 0.0i
3.0 + 1.0i
4.0 + 2.0i
Result:
2.0 + 2.0i
4.0 + 4.0i
6.0 + 6.0i

```

Рисунок 4 – Результат выполнения программы

### Задача 2.1:

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы `setТип()`, `getТип()`, `toString()`. Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль

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

### Листинг 5 программы main:

```

//Определить конструкторы и методы setТип(), getТип(), toString().
// Определить дополнительно методы в классе, создающем массив объектов.
// Задать критерий выбора данных и вывести эти данные на консоль
// Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет,
Кредит, Время городских и
// междугородных разговоров.
// Создать массив объектов.
// Вывести: а) сведения об абонентах, у которых время внутригородских
разговоров превышает заданное;
//          б) сведения об абонентах, которые пользовались междугородной
связью;
//          с) сведения об абонентах в алфавитном порядке.
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Company result test1 = new Company result();

```



```

test1.add(new Phone_Operator("Grigoryan","David","Rubenovich",
    "Erevan",7592837992323L,545.0,1743.0,456,0));
test1.add(new Phone_Operator("Askerova","Nargiz","Alexandrovna",
    "Baku",7928734202441L,1200.0,1252.0,3446,1243));
test1.add(new Phone_Operator("Zamula","Margo","Viktorovna",
    "Tbilisi",8792032423167L,2000.0,1942.0,168,654));
test1.add(new Phone_Operator("Morozova","Nastya","Vladimirovna",
    "Kiev",1347290739242L,2412.0,1210.0,7952,2345));
test1.add(new Phone_Operator("Panfilkin","Artyom","Makarovich",
    "Minsk",1980923453455L,2342.0,2334.0,1236,254));
test1.add(new Phone_Operator("Ivanov","Vladimir","Igorevich",
    "Krasnodar",1989068304323L,6542.0,3534.0,6546,8765));

System.out.println("Unsorted list of clients");
test1.show();
System.out.println("Sorted list of clients by Last Name:");
test1.getSortByName();
test1.show();

System.out.println("Information about clients whose time of intra-
city calls exceeds the specified(intercity time > 1000 ms)");
ArrayList<Phone_Operator> listLocalLimit = new
ArrayList<Phone_Operator>();
listLocalLimit = test1.getClientsLocalLimitsTime(1000);
for (Phone_Operator client : listLocalLimit){
    System.out.println(client);
}

System.out.println("Information about clients who used long-distance
communication");
ArrayList<Phone_Operator> listInter = new
ArrayList<Phone_Operator>();
listInter = test1.getClientsByInternationTime(0);
for (Phone_Operator client : listInter){
    System.out.println(client);
}
}
}

```

## Листинг 6 класса Phone\_Operator:

```

class Phone_Operator { // ИМЯ
    private static int all_clients=0;
    private int id = all_clients++;
    private String Last_name, First_name, Patron, Address; // ФИО и адрес
    private long credit_card;
    private double debet,credit,town_time,intercity_time;

    public Phone_Operator() {
    }

    public Phone_Operator(String last_name, String first_name, String patron,
String address,
                        long credit_card, double debet, double credit,
double town_time, double intercity_time) {
        Last_name = last_name;
        First_name = first_name;
        Patron = patron;
        Address = address;
        this.credit_card = credit_card;
    }
}

```

```

        this.debet = debet;
        this.credit = credit;
        this.town_time = town_time;
        this.intercity_time = intercity_time;
    }

    public String getFirst_name() {
        return First_name;
    }

    public String getLast_name() {
        return Last_name;
    }

    public String getPatron() {
        return Patron;
    }

    public String getAddress() {
        return Address;
    }

    public double getId() {
        return id;
    }

    public double getDebet() {
        return debet;
    }

    public double getCredit() {
        return credit;
    }

    public double getTown_time() {
        return town_time;
    }

    public double getIntercity_time() {
        return intercity_time;
    }

    public long getCredit_card() {
        return credit_card;
    }

    public void setLast_name(String last_name) {
        Last_name = last_name;
    }

    public void setFirst_name(String first_name) {
        First_name = first_name;
    }

    public void setPatron(String patron) {
        Patron = patron;
    }

    public void setAddress(String address) {
        Address = address;
    }

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

```

```

    }

    public void setDebet(double debet) {
        this.debet = debet;
    }

    public void setCredit(double credit) {
        this.credit = credit;
    }

    public void setTown_time(double town_time) {
        this.town_time = town_time;
    }

    public void setIntercity_time(double intercity_time) {
        this.intercity_time = intercity_time;
    }

    public void setCredit_card(long credit_card) {
        this.credit_card = credit_card;
    }

    @Override
    public String toString() {
        return "Phone_Operator{" +
            "all_clients=" + all_clients +
            ", id=" + id +
            ", Last_name='" + Last_name + '\'' +
            ", First_name='" + First_name + '\'' +
            ", Patron='" + Patron + '\'' +
            ", Address='" + Address + '\'' +
            ", credit_card=" + credit_card +
            ", debet=" + debet +
            ", credit=" + credit +
            ", town_time=" + town_time +
            ", intercity_time=" + intercity_time +
            '}';
    }
}

```

### Листинг 7 класса Company\_result:

```

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Company_result {
    private ArrayList<Phone_Operator> list_result = new ArrayList<>();
    void add(Phone_Operator phone) {
        list_result.add(phone);
    }
    void show() {
        for (Phone_Operator client : list_result){
            System.out.println(client);
        }
    }
    public ArrayList<Phone_Operator> getSortByName() {
        ArrayList<Phone_Operator> tempList = new ArrayList<Phone_Operator>();
        Comparator<Phone_Operator> countryModelsComparator
            = Comparator.comparing(Phone_Operator::getLast_name);

        Collections.sort(list_result, countryModelsComparator);
    }
}

```

```

        return tempList;
    }
    public ArrayList<Phone_Operator> getClientsLocalLimitsTime(int time) {
        ArrayList<Phone_Operator> tempList = new ArrayList<Phone_Operator>();
        for (Phone_Operator client : list_result){
            if (client.getIntercity_time() > time){
                tempList.add(client);
            }
        }
        return tempList;
    }
    public ArrayList<Phone_Operator> getClientsByInternationTime(int
inter_time) {
        ArrayList<Phone_Operator> tempList = new ArrayList<Phone_Operator>();
        for (Phone_Operator client : list_result){
            if (client.getIntercity_time() > inter_time){
                tempList.add(client);
            }
        }
        return tempList;
    }
}

```

Результат приведен на рисунке 5:

```

Unsorted list of clients
Phone_Operator{all_clients=6, id=0, Last_name='Grigoryan', First_name='David', Patron='Rubenovich', Address='Erevan', credit_card=7592837992323, debet=545.0, credit=1743.0, town_
Phone_Operator{all_clients=6, id=1, Last_name='Askerova', First_name='Nargiz', Patron='Alexandrovna', Address='Baku', credit_card=7928734202441, debet=1200.0, credit=1252.0, town_
Phone_Operator{all_clients=6, id=2, Last_name='Zamula', First_name='Margo', Patron='Viktorovna', Address='Tbilisi', credit_card=8792032423167, debet=2000.0, credit=1942.0, town_
Phone_Operator{all_clients=6, id=3, Last_name='Morozova', First_name='Nastya', Patron='Vladimirovna', Address='Kiev', credit_card=1347290739242, debet=2412.0, credit=1210.0, town_
Phone_Operator{all_clients=6, id=4, Last_name='Panfilkin', First_name='Artyom', Patron='Makarovich', Address='Minsk', credit_card=1980923453455, debet=2342.0, credit=2334.0, town_
Phone_Operator{all_clients=6, id=5, Last_name='Ivanov', First_name='Vladimir', Patron='Igorovich', Address='Krasnodar', credit_card=1989068394323, debet=6542.0, credit=3534.0, town_
Sorted list of clients by Last Name:
Phone_Operator{all_clients=6, id=1, Last_name='Askerova', First_name='Nargiz', Patron='Alexandrovna', Address='Baku', credit_card=7928734202441, debet=1200.0, credit=1252.0, town_
Phone_Operator{all_clients=6, id=0, Last_name='Grigoryan', First_name='David', Patron='Rubenovich', Address='Erevan', credit_card=7592837992323, debet=545.0, credit=1743.0, town_
Phone_Operator{all_clients=6, id=5, Last_name='Ivanov', First_name='Vladimir', Patron='Igorovich', Address='Krasnodar', credit_card=1989068394323, debet=6542.0, credit=3534.0, town_
Phone_Operator{all_clients=6, id=3, Last_name='Morozova', First_name='Nastya', Patron='Vladimirovna', Address='Kiev', credit_card=1347290739242, debet=2412.0, credit=1210.0, town_
Phone_Operator{all_clients=6, id=4, Last_name='Panfilkin', First_name='Artyom', Patron='Makarovich', Address='Minsk', credit_card=1980923453455, debet=2342.0, credit=2334.0, town_
Phone_Operator{all_clients=6, id=2, Last_name='Zamula', First_name='Margo', Patron='Viktorovna', Address='Tbilisi', credit_card=8792032423167, debet=2000.0, credit=1942.0, town_
Information about clients whose time of intra-city calls exceeds the specified(intercity time > 1000 ms)
Phone_Operator{all_clients=6, id=1, Last_name='Askerova', First_name='Nargiz', Patron='Alexandrovna', Address='Baku', credit_card=7928734202441, debet=1200.0, credit=1252.0, town_
Phone_Operator{all_clients=6, id=5, Last_name='Ivanov', First_name='Vladimir', Patron='Igorovich', Address='Krasnodar', credit_card=1989068394323, debet=6542.0, credit=3534.0, town_
Phone_Operator{all_clients=6, id=3, Last_name='Morozova', First_name='Nastya', Patron='Vladimirovna', Address='Kiev', credit_card=1347290739242, debet=2412.0, credit=1210.0, town_
Information about clients who used long-distance communication
Phone_Operator{all_clients=6, id=1, Last_name='Askerova', First_name='Nargiz', Patron='Alexandrovna', Address='Baku', credit_card=7928734202441, debet=1200.0, credit=1252.0, town_
Phone_Operator{all_clients=6, id=5, Last_name='Ivanov', First_name='Vladimir', Patron='Igorovich', Address='Krasnodar', credit_card=1989068394323, debet=6542.0, credit=3534.0, town_
Phone_Operator{all_clients=6, id=3, Last_name='Morozova', First_name='Nastya', Patron='Vladimirovna', Address='Kiev', credit_card=1347290739242, debet=2412.0, credit=1210.0, town_
Phone_Operator{all_clients=6, id=4, Last_name='Panfilkin', First_name='Artyom', Patron='Makarovich', Address='Minsk', credit_card=1980923453455, debet=2342.0, credit=2334.0, town_
Phone_Operator{all_clients=6, id=2, Last_name='Zamula', First_name='Margo', Patron='Viktorovna', Address='Tbilisi', credit_card=8792032423167, debet=2000.0, credit=1942.0, town_

```

Рисунок 5 – Результат выполнения программы

### Задача 2.2:

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы setType(), getType(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. Создать массив объектов. Вывести: а) список автомобилей заданной марки; б) список автомобилей заданной модели, которые эксплуатируются

больше n лет; с) список автомобилей заданного года выпуска, цена которых больше указанной.

#### Листинг 8 программы main:

```
public class Main {
    public static void main(String[] args) {
        Car[] cars = {
            new Car(1, "Toyota", "Camry", 2018, "Silver", 25000,
"ABC123"),
            new Car(2, "Honda", "Civic", 2019, "Red", 20000, "DEF456"),
            new Car(3, "Toyota", "Corolla", 2017, "Black", 22000,
"GHI789"),
            new Car(4, "Ford", "Mustang", 2015, "Yellow", 30000,
"JKL012"),
            new Car(5, "Chevrolet", "Camaro", 2016, "White", 28000,
"MNO345")
        };

        CarManager cm = new CarManager(cars);

        // Get cars by brand
        Car[] toyotas = cm.getCarsByBrand("Toyota");
        System.out.println("Cars by brand:");
        for (Car car : toyotas) {
            System.out.println(car.toString());
        }

        // Get cars by model and age
        Car[] oldCivics = cm.getCarsByModelAndAge("Civic", 3);
        System.out.println("\nOld cars by model:");
        for (Car car : oldCivics) {
            System.out.println(car.toString());
        }

        // Get cars by year and price
        Car[] expensive2016 = cm.getCarsByYearAndPrice(2016, 25000);
        System.out.println("\nExpensive cars by year:");
        for (Car car : expensive2016) {
            System.out.println(car.toString());
        }
    }
}
```

#### Листинг 9 класса Car:

```
public class Car {
    private int id;
    private String brand;
    private String model;
    private int year;
    private String color;
    private double price;
    private String regNumber;

    public Car(int id, String brand, String model, int year, String color,
double price, String regNumber) {
        this.id = id;
        this.brand = brand;
        this.model = model;
    }
}
```

```

        this.year = year;
        this.color = color;
        this.price = price;
        this.regNumber = regNumber;
    }

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

    public int getId() {
        return id;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getBrand() {
        return brand;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public String getModel() {
        return model;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public int getYear() {
        return year;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public String getColor() {
        return color;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public double getPrice() {
        return price;
    }

    public void setRegNumber(String regNumber) {
        this.regNumber = regNumber;
    }

    public String getRegNumber() {
        return regNumber;
    }

    public String toString() {
        return "Car{" +

```

```

        "id=" + id +
        ", brand='" + brand + '\'' +
        ", model='" + model + '\'' +
        ", year=" + year +
        ", color='" + color + '\'' +
        ", price=" + price +
        ", regNumber='" + regNumber + '\'' +
        '}' +
    };
}
}

```

### Листинг 10 класса CarManager:

```

class CarManager {
    private Car[] cars;

    public CarManager(Car[] cars) {
        this.cars = cars;
    }

    public Car[] getCars() {
        return cars;
    }

    public void setCars(Car[] cars) {
        this.cars = cars;
    }

    public Car[] getCarsByBrand(String brand) {
        int count = 0;
        for (Car car : cars) {
            if (car.getBrand().equals(brand)) {
                count++;
            }
        }
        Car[] result = new Car[count];
        int index = 0;
        for (Car car : cars) {
            if (car.getBrand().equals(brand)) {
                result[index] = car;
                index++;
            }
        }
        return result;
    }

    public Car[] getCarsByModelAndAge(String model, int age) {
        int count = 0;
        for (Car car : cars) {
            if (car.getModel().equals(model) && car.getYear() <= (2023 -
age)) {
                count++;
            }
        }
        Car[] result = new Car[count];
        int index = 0;
        for (Car car : cars) {
            if (car.getModel().equals(model) && car.getYear() <= (2023 -
age)) {
                result[index] = car;
                index++;
            }
        }
    }
}

```

```

    }
    return result;
}

public Car[] getCarsByYearAndPrice(int year, double price) {
    int count = 0;
    for (Car car : cars) {
        if (car.getYear() == year && car.getPrice() > price) {
            count++;
        }
    }
    Car[] result = new Car[count];
    int index = 0;
    for (Car car : cars) {
        if (car.getYear() == year && car.getPrice() > price) {
            result[index] = car;
            index++;
        }
    }
    return result;
}
}

```

Результат приведен на рисунке 6:

```

Cars by brand:
Car{id=1, brand='Toyota', model='Camry', year=2018, color='Silver', price=25000.0, regNumber='ABC123'}
Car{id=3, brand='Toyota', model='Corolla', year=2017, color='Black', price=22000.0, regNumber='GHI789'}

Old cars by model:
Car{id=2, brand='Honda', model='Civic', year=2019, color='Red', price=20000.0, regNumber='DEF456'}

Expensive cars by year:
Car{id=5, brand='Chevrolet', model='Camaro', year=2016, color='White', price=28000.0, regNumber='MN0345'}

```

Рисунок 6 – Результат выполнения программы

### Задача 3.1:

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

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

### Листинг 11 программы main:

```

//Создать приложение, удовлетворяющее требованиям, приведенным в задании.
//Аргументировать принадлежность
//классу каждого создаваемого метода и корректно переопределить для
//каждого класса
//методы equals(), hashCode(), toString().
//7. Создать объект класса Дерево, используя классы Лист.
//Методы: зацвести, опать листьям, покрыться инеем, пожелтеть

```



```

ЛИСТЬЯМ
import java.util.*;
public class Main {
    public static void main(String[] args) {
        List<Leaf> leaves = new ArrayList<>();
        Leaf leaf1 = new Leaf("green", "round");
        Leaf leaf2 = new Leaf("green", "oval");
        Leaf leaf3 = new Leaf("green", "round");
        leaves.add(leaf1);
        leaves.add(leaf2);
        leaves.add(leaf3);
        Tree tree = new Tree(leaves);

        System.out.println(leaf1.equals(leaf2));
        System.out.println(leaf1.equals(leaf3));
        System.out.println(leaf1.hashCode());
        System.out.println(leaf3.hashCode());
        System.out.println(leaf2.hashCode());
        System.out.println(tree.equals(leaf1));
        System.out.println(tree.equals(tree));
        System.out.println(leaf1.hashCode());

        System.out.println(tree.toString());
        tree.bloom();
        System.out.println(tree.toString());
        tree.frost();
        System.out.println(tree.toString());
        tree.yellowLeaves();
        System.out.println(tree.toString());
        tree.fallLeaves();
        System.out.println(tree.toString());
    }
}

```

## Листинг 12 класса Leaf:

```

import java.util.Objects;

class Leaf {
    private String color;
    private String shape;
    public Leaf(String color, String shape) {
        this.color = color;
        this.shape = shape;
    }
    public String getColor() {
        return color;
    }
    public String getShape() {
        return shape;
    }
    public void setColor(String color) {
        this.color = color;
    }
    public void setShape(String shape) {
        this.shape = shape;
    }
    @Override
    public String toString() {
        return "Leaf{" +
            "color='" + color + '\'' +
            ", shape='" + shape + '\'' +

```

```

    }
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof Leaf)) return false;
    Leaf leaf = (Leaf) o;
    return Objects.equals(color, leaf.color) &&
        Objects.equals(shape, leaf.shape);
}

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

```

### Листинг 13 класса Tree:

```

import java.util.List;
import java.util.Objects;

class Tree {
    private List<Leaf> leaves;
    private boolean isBlooming; // зацвести
    private boolean hasFrost; // иней
    private boolean isYellowing; // Пожелтеть
    public Tree(List<Leaf> leaves) {
        this.leaves = leaves;
        this.isBlooming = false;
        this.hasFrost = false;
        this.isYellowing = false;
    }
    public void bloom() {
        this.isBlooming = true;
    }
    public void fallLeaves() {
        this.leaves.clear();
    }
    public void frost() {
        this.hasFrost = true;
    }
    public void yellowLeaves() {
        this.isYellowing = true;
    }
    public List<Leaf> getLeaves() {
        return leaves;
    }
    public boolean isBlooming() {
        return isBlooming;
    }
    public boolean hasFrost() {
        return hasFrost;
    }
    public boolean isYellowing() {
        return isYellowing;
    }
    @Override
    public String toString() {
        return "Tree{" +
            "leaves=" + leaves +
            ", isBlooming=" + isBlooming +
            ", hasFrost=" + hasFrost +

```

```

        ", isYellowing=" + isYellowing +
        " '}'";
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Tree)) return false;
        Tree tree = (Tree) o;
        return isBlooming == tree.isBlooming &&
            hasFrost == tree.hasFrost &&
            isYellowing == tree.isYellowing &&
            Objects.equals(leaves, tree.leaves);
    }
    @Override
    public int hashCode() {
        return Objects.hash(leaves, isBlooming, hasFrost, isYellowing);
    }
}

```

Результат приведен на рисунке 7:

```

false
true
-1129068884
-1129068884
-1234349712
false
true
-1129068884
Tree{leaves=[Leaf{color='green', shape='round'}, Leaf{color='green', shape='oval'}, Leaf{color='green', shape='round'}], isBlooming=false, hasFrost=false, isYellowing=false}
Tree{leaves=[Leaf{color='green', shape='round'}, Leaf{color='green', shape='oval'}, Leaf{color='green', shape='round'}], isBlooming=true, hasFrost=false, isYellowing=false}
Tree{leaves=[Leaf{color='green', shape='round'}, Leaf{color='green', shape='oval'}, Leaf{color='green', shape='round'}], isBlooming=true, hasFrost=true, isYellowing=false}
Tree{leaves=[Leaf{color='green', shape='round'}, Leaf{color='green', shape='oval'}, Leaf{color='green', shape='round'}], isBlooming=true, hasFrost=true, isYellowing=true}
Tree{leaves=[], isBlooming=true, hasFrost=true, isYellowing=true}

```

Рисунок 7 – Результат выполнения программы

### Задача 3.2:

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

Создать объект класса Пианино, используя класс Клавиша. Методы: настроить, играть на пианино, нажимать клавишу.

### Листинг 14 программы main:

```

import java.util.*;
//Создать объект класса Пианино, используя класс Клавиша. Методы: настроить,
играть на пианино, нажимать клавишу.
public class Main {
    public static void main(String[] args) {
        List<Key> keys = new ArrayList<>();
        Key key1 = new Key(1, false, false);
        Key key2 = new Key(2, false, true);
        keys.add(key1);
        keys.add(key2);
        Piano piano = new Piano(keys);
        System.out.println(piano.toString());
    }
}

```

```

        piano.tune();
        System.out.println(piano.toString());
        piano.play();
        piano.pressKey(key1);
        piano.pressKey(key2);
    }
}

```

## Листинг 15 класса Key:

```

import java.util.Objects;

class Key {
    private int number;
    private boolean isPressed;
    private boolean isBroken;

    public Key(int number, boolean isPressed, boolean isBroken) {
        this.number = number;
        this.isPressed = isPressed;
        this.isBroken = isBroken;
    }

    public void press() {
        this.isPressed = true;
        System.out.println("Key #" + this.number + " is pressed");
    }

    public int getNumber() {
        return number;
    }

    public void setNumber(int number) {
        this.number = number;
    }

    public boolean isPressed() {
        return isPressed;
    }

    public void setPressed(boolean pressed) {
        isPressed = pressed;
    }

    public boolean isBroken() {
        return isBroken;
    }

    public void setBroken(boolean broken) {
        isBroken = broken;
    }

    @Override
    public String toString() {
        return "Key{" +
            "number=" + number +
            ", isPressed=" + isPressed +
            ", isBroken=" + isBroken +
            '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
    }
}

```

```

        if (o == null || getClass() != o.getClass()) return false;
        Key key = (Key) o;
        return number == key.number &&
            isPressed == key.isPressed &&
            isBroken == key.isBroken;
    }
    @Override
    public int hashCode() {
        return Objects.hash(number, isPressed, isBroken);
    }
}

```

### Листинг 16 класса Piano:

```

import java.util.List;
import java.util.Objects;

class Piano {
    private List<Key> keys;
    private boolean isTuned;

    public Piano(List<Key> keys) {
        this.keys = keys;
        this.isTuned = false;
    }

    public void tune() {
        this.isTuned = true;
    }

    public void play() {
        if (!this.isTuned) {
            System.out.println("Piano is not tuned");
        } else {
            System.out.println("Playing the piano");
        }
    }

    public void pressKey(Key key) {
        if (!this.isTuned) {
            System.out.println("Piano is not tuned");
        } else if (key.isBroken()) {
            System.out.println("Key is broken");
        } else {
            key.press();
        }
    }

    public List<Key> getKeys() {
        return keys;
    }

    public boolean isTuned() {
        return isTuned;
    }
    @Override
    public String toString() {
        return "Piano{" +
            "keys=" + keys +
            ", isTuned=" + isTuned +
            '}';
    }
}

```

```

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Piano piano = (Piano) o;
    return isTuned == piano.isTuned &&
        Objects.equals(keys, piano.keys);
}
@Override
public int hashCode() {
    return Objects.hash(keys, isTuned);
}
}

```

Результат приведен на рисунке 8:

```

Piano{keys=[Key{number=1, isPressed=false, isBroken=false}, Key{number=2, isPressed=false, isBroken=true}], isTuned=false}
Piano{keys=[Key{number=1, isPressed=false, isBroken=false}, Key{number=2, isPressed=false, isBroken=true}], isTuned=true}
Playing the piano
Key #1 is pressed
Key is broken

```

Рисунок 8 – Результат выполнения программы

#### Задача 4.1:

Построить модель программной системы.

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.

#### Листинг 17 программы main:

```

import java.util.ArrayList;

//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.
public class Main {
    public static void main(String[] args) {
        Admin ADMIN = new Admin();
        ArrayList<Customer> List2 = new ArrayList<Customer>();
        ArrayList<Service> ServicesForClient1 = new ArrayList<Service>();
        ServicesForClient1.add(new Service("Inet", 500));
        ServicesForClient1.add(new Service("TV", 350));
        ServicesForClient1.add(new Service("Phone", 170));
        Customer Tom = new Customer("Grigoryan", "David", 567482,
            919964, 567, true, 0);
        Tom.AddServices(ServicesForClient1);
        List2.add(Tom);

        ArrayList<Service> ServicesForClient2 = new ArrayList<Service>();
        ServicesForClient2.add(new Service("Inet", 500));
    }
}

```

```

        ServicesForClient2.add(new Service("Phone", 170));
        Customer Andy = new Customer("Ivanov", "Alex", 100,
            991264, 127, true, 0);
        Andy.AddServices(ServicesForClient2);
        List2.add(Andy);
        Andy.GetAdmin (ADMIN, ServicesForClient2);

        ADMIN.getallclients(List2);
        Andy.PayTheBill();
        System.out.println(Andy.on);
    }
}

```

### Листинг 18 класса Service:

```

public class Service {
    String service;
    int price;

    public Service(String service, int price) {
        this.service = service;
        this.price = price;
    }
}

```

### Листинг 19 класса Admin:

```

import java.util.ArrayList;
import java.util.Random;

public class Admin {
    private static ArrayList<Customer> list_result = new ArrayList<>();
    public int ChangeNumber(int num) {
        Random rand = new Random();
        int newnumber = (int) (Math.random() * 900000) + 100000;
        return newnumber;
    }
    public void RefuseSevices(ArrayList<Service> list) {
        list.clear();
    }
    public void NonPayment(int a) {
        for (Customer client : list_result) {
            if (client.number == a) {
                client.on = false;
                client.time = 1440;
            }
        }
    }
    public void getallclients(ArrayList<Customer> list5) {
        list_result = list5;
    }
}

```

### Листинг 20 класса Customer:

```

import java.util.ArrayList;
import java.util.Random;

public class Customer {
    String FName, LName;
}

```

```

int account, number, phoneConversations;
boolean on;
int time;
int TheFinalPrice;

public Customer(String FName, String LName, int account, int number, int
phoneConversations, boolean on, int time) {
    this.FName = FName;
    this.LName = LName;
    this.account = account;
    this.number = number;
    this.phoneConversations = phoneConversations;
    this.on = on;
    this.time = time;
}

public Customer() {
}

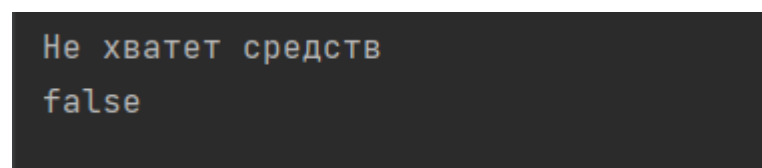
public void AddServices(ArrayList<Service> list) {
    ArrayList<Service> ServicesCust = new ArrayList<Service>();
    ServicesCust = list;
    for (Service serv : list){
        TheFinalPrice = TheFinalPrice + serv.price;
    }
}

public void PayTheBill() {
    if (this.account > TheFinalPrice) {
        this.account = this.account-TheFinalPrice;
        System.out.printf("Счет оплачен, уважаемый ",this.FName);
        System.out.println("Текущий счет:");
        System.out.println(this.account);
    }
    else {
        System.out.printf("Не хватает средств",this.FName);
        Admin AD = new Admin();
        AD.NonPayment(this.number);
    }
}

public void GetAdmin(Admin name,ArrayList<Service> list) {
    this.number = name.ChangeNumber(this.number);
    name.RefuseSevices(list);
}
}

```

Результат приведен на рисунке 9:



```

Не хватает средств
false

```

Рисунок 9 – Результат выполнения программы



#### Задача 4.2:

Построить модель программной системы.

Система Автобаза. Диспетчер распределяет заявки на Рейсы между Водителями и назначает для этого Автомобиль. Водитель может сделать заявку на ремонт. Диспетчер может отстранить Водителя от работы. Водитель делает отметку о выполнении Рейса и состоянии Автомобиля.

#### Листинг 21 программы main:

```
import java.util.ArrayList;

//Система Автобаза. Диспетчер распределяет заявки на Рейсы между Водителями и
//назначает для этого Автомобиль.
// Водитель может сделать заявку на ремонт.
// Диспетчер может отстранить Водителя от работы. Водитель делает отметку о
//выполнении Рейса и состоянии Автомобиля
public class Main {
    public static void main(String[] args) {
        ArrayList<Auto> allAuto = new ArrayList<Auto>();
        ArrayList<Driver> allDrivers = new ArrayList<Driver>();
        ArrayList<Trip> alltrips = new ArrayList<Trip>();
        Operator Dispatch = new Operator();

        Auto Ford = new Auto("Ford",true,true,false,100,123456);
        Auto Toyota = new Auto("Toyota",true,true,false,40,123412);
        Auto Mini = new Auto("Mini",false,false,false,10,765847);
        Auto BMW = new Auto("BMW",false,true,false,0,456792);
        Auto Lada = new Auto("Lada",true,true,false,180,827346);
        allAuto.add(Ford);
        allAuto.add(Toyota);
        allAuto.add(Mini);
        allAuto.add(BMW);
        allAuto.add(Lada);

        Driver alex = new Driver("Alex","Grigoryan",34,456792,true,true);
        allDrivers.add(new Driver("Alex","Grigoryan",34,456792,true,true));
        allDrivers.add(new Driver("Eren","Rayan",24,0,false,true));
        allDrivers.add(new Driver("David","Mosin",43,0,true,true));
        allDrivers.add(new Driver("Slava","Lobok",50,0,true,true));

        alltrips.add(new Trip("Moscow","Ramenskoe","",100,0,false));
        alltrips.add(new Trip("Minsk","Kaliningrad","",480,0,false));
        alltrips.add(new Trip("Erevan","Zelenograd","",100,0,false));
        alltrips.add(new Trip("Vilnius","Warsava","",780,0,false));
        //Dispatch.SuspendDriver("Mosin",allDrivers);
        for (Driver uber : allDrivers){
            System.out.printf("%s: ",uber.FirstName);
            System.out.println(uber.active);
        }
        Dispatch.DistributeRequests(allDrivers,allAuto,alltrips);
        alex.TicketForRepair(456792,allAuto);
        alex.ReportOfAuto(BMW);
    }
}
```

### Листинг 22 класса Auto:

```
public class Auto {
    String car_model;
    boolean ready,breaks,OnRepair;
    int fuel,carID;

    public Auto(String car_model, boolean ready, boolean breaks, boolean
onRepair, int fuel, int carID) {
        this.car_model = car_model;
        this.ready = ready;
        this.breaks = breaks;
        OnRepair = onRepair;
        this.fuel = fuel;
        this.carID = carID;
    }
}
```

### Листинг 23 класса Driver:

```
import java.util.ArrayList;

public class Driver {
    String FirstName,LastName;
    int age,carID;
    boolean active,free;

    public Driver(String firstName, String lastName, int age, int carID,
boolean active, boolean free) {
        FirstName = firstName;
        LastName = lastName;
        this.age = age;
        this.carID = carID;
        this.active = active;
        this.free = free;
    }

    public void TicketForRepair(int carID, ArrayList<Auto> list2) {
        for (Auto car : list2){
            if (car.carID==carID) {
                car.ready=false;
                car.OnRepair = true;
            }
        }
    }

    public void ReportOfAuto(Auto car) {
        System.out.printf("Report about the car:%s
%d\n",car.car_model,carID);
        if (car.ready) {
            System.out.println("This car is fine and ready to go!: ");
        } else if (!car.breaks) {
            System.out.println("The brakes of this car are not working!
");
        } else if (car.fuel==0) {
            System.out.println("Gasoline is empty: ");
        }
    }
}
```

### Листинг 24 класса Trip:

```

public class Trip {
    String departure,destination,NameDriver;
    int distance,carID;
    boolean pending;

    public Trip(String departure, String destination, String nameDriver, int
distance, int carID, boolean pending) {
        this.departure = departure;
        this.destination = destination;
        NameDriver = nameDriver;
        this.distance = distance;
        this.carID = carID;
        this.pending = pending;
    }
}

```

### Листинг 25 класса Operator:

```

import java.util.ArrayList;

public class Operator {

    public void SuspendDriver(String name,ArrayList<Driver> list2) {
        for (Driver uber : list2){
            if (uber.LastName.equals(name)) {
                uber.active=false;
            }
        }
        System.out.printf("The driver %s suspended!\n",name);
    }

    public void DistributeRequests(ArrayList<Driver>
list_drivers,ArrayList<Auto> list_cars,ArrayList<Trip> list_trips) {
//        var f = list_trips.stream().filter(trip -> !trip.pending);
//        var f2 = list_drivers.stream().filter(driver -> driver.active &&
driver.carID == 0 && driver.free);
//        f.forEach( trip -> {
//            f2.forEach(driver -> {
//                var filtered_cars = list_cars.stream().filter(car ->
car.ready && trip.carID == 0);
//                filtered_cars.forEach();
//            });
//        });
        for (Trip trip: list_trips) {
            System.out.printf("The TRIP from %s to %s
\n",trip.departure,trip.destination);
            if (!trip.pending) {
                for (Driver uber : list_drivers) {
                    if (uber.active && uber.carID == 0 && uber.free) {
                        for (Auto car : list_cars) {
                            if (car.ready && trip.carID==0) {
                                trip.carID= car.carID;
                                trip.NameDriver= uber.FirstName;
                                uber.free=false;
                                uber.carID = car.carID;
                                car.ready = false;
                                System.out.printf("This TRIP will do the
driver %s with car %s !\n",uber.FirstName,car.car_model);
                                break;
                            }
                        }
                    }
                }
            }
            if (trip.carID==0) {

```

