



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 лабораторной работы:** Классы. Наследование. Полиморфизм

**Дисциплина:** Языки программирования для работы с большими данными

Студент гр. ИУ6-22М

\_\_\_\_\_  
(Подпись, дата)

**И.А. Горшков**  
(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

**П.В. Степанов**  
(И.О. Фамилия)

Москва, 2023

## **Введение**

Целью лабораторной работы является приобретение навыков работы с классами, наследованием и полиморфизмом на языке программирования Java.

## Практическая часть

### Задание 1

Определить класс Цепная дробь

$$A = a_0 + \frac{x}{a_1 + \frac{x}{a_2 + \frac{x}{a_3 + \dots}}}$$

Определить методы сложения, вычитания, умножения, деления.

Вычислить значение для заданного n, x, a[n].

Код написанной программы представлен в листинге 1.

Листинг 1 – Программа для первого задания

```
package com.java.lab;

import java.util.Scanner;

// Определить класс Цепная дробь.
// Определить методы сложения, вычитания, умножения, деления.
// Вычислить значение для заданного n, x, a[n].
public class Main {
    public static void main(String[] args) {

        ContinuedFraction firstContinuedFraction =
makeChainFraction();
        System.out.println("Total value: " +
firstContinuedFraction.getValue());
        ContinuedFraction secondContinuedFraction =
makeChainFraction();
        System.out.println("Total value: " +
secondContinuedFraction.getValue());

        System.out.println("sum: " +
firstContinuedFraction.add(secondContinuedFraction));
        System.out.println("subtract: " +
firstContinuedFraction.subtract(secondContinuedFraction));
        System.out.println("multiply: " +
firstContinuedFraction.multiply(secondContinuedFraction));
        System.out.println("divide: " +
firstContinuedFraction.divide(secondContinuedFraction));
    }

    public static ContinuedFraction makeChainFraction() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("number of coefficients: ");
        var n = scanner.nextInt();

        System.out.print("continued fraction numerator: ");
```

```

        var x = scanner.nextInt();

        var a = new double[n];
        for (var i = 0; i < n; i++) {
            System.out.print("Enter coefficient No. " + (i) + " : ");
            a[i] = scanner.nextInt();
        }

        return new ContinuedFraction(x, n, a);
    }
}

class ContinuedFraction {
    private final double[] a;
    private final int n;
    private final int x;

    ContinuedFraction(int x, int n, double[] a) {
        this.x = x;
        this.n = n;
        this.a = a;
    }

    double getValue() {
        var value = a[n-1];
        for (var i = n - 2; i >= 0; i--) {
            value = a[i] + x / value;
        }
        return value;
    }

    double add(ContinuedFraction continuedFraction) {
        return this.getValue() + continuedFraction.getValue();
    }

    double subtract(ContinuedFraction continuedFraction) {
        return this.getValue() - continuedFraction.getValue();
    }

    double multiply(ContinuedFraction continuedFraction) {
        return this.getValue() * continuedFraction.getValue();
    }

    double divide(ContinuedFraction continuedFraction) {
        return this.getValue() / continuedFraction.getValue();
    }
}

```

Результат выполнения программы показан на рисунке 1.

```
number of coefficients: 2
continued fraction numerator: 3
Enter coefficient No. 0 : 4
Enter coefficient No. 1 : 5
Total value: 4.6
number of coefficients: 6
continued fraction numerator: 7
Enter coefficient No. 0 : 8
Enter coefficient No. 1 : 7
Enter coefficient No. 2 : 6
Enter coefficient No. 3 : 5
Enter coefficient No. 4 : 4
Enter coefficient No. 5 : 3
Total value: 8.877248677248677
sum: 13.477248677248676
subtract: -4.277248677248677
multiply: 40.835343915343906
divide: 0.5181785671712957
```

Рисунок 1 – Результат выполнения программы

## Задание 2

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.

Код написанной программы представлен в листинге 2.

## Листинг 2 – Программа для второго задания

```
package com.java.lab;

import java.util.Scanner;
```

```

class Fraction {
    int m, n;

    public Fraction() {
        this.m = 0;
        this.n = 1;
    }

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

    public Fraction add(Fraction addedFraction) {
        var sum = new Fraction();
        sum.n = this.n * addedFraction.n;
        sum.m = m * addedFraction.n + addedFraction.m * n;
        return sum;
    }

    public Fraction multiply(Fraction multipliedFraction) {
        return new Fraction(
            m * multipliedFraction.m,
            n * multipliedFraction.n
        );
    }

    public Fraction subtract(Fraction subtractedFraction) {
        Fraction invertedFraction = new Fraction(-
subtractedFraction.m, subtractedFraction.n);
        return add(invertedFraction);
    }

    public Fraction divide(Fraction dividedFraction) {
        Fraction switchedFraction = new Fraction(dividedFraction.n,
dividedFraction.m);
        return this.multiply(switchedFraction);
    }

    public String toString() {
        return m + "/" + n;
    }
}

```

// 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) {
        System.out.println("k: ");

        Scanner console = new Scanner(System.in);
        var k = console.nextInt();

        Fraction[] fractions = new Fraction[k];
    }
}

```

```

        for (var i = 0; i < k; i++) {
            System.out.println("input m for Fraction No. " + (i + 1) +
":");
            var m = console.nextInt();
            System.out.println("input n for Fraction No. " + (i + 1) +
":");
            var n = console.nextInt();

            fractions[i] = new Fraction(m, n);
        }

        print(fractions);
        System.out.println("sum:");

System.out.println((fractions[0].add(fractions[1]).toString()));
        System.out.println("subtract:");

System.out.println((fractions[0].subtract(fractions[1]).toString()));
        System.out.println("multiply:");

System.out.println((fractions[0].multiply(fractions[1]).toString()));
        System.out.println("divide:");

System.out.println((fractions[0].divide(fractions[1]).toString()));
        changeFraction(fractions);
        print(fractions);
    }

    private static void changeFraction(Fraction[] fractions) {
        for (int i = 0; i < fractions.length - 1; i += 2) {
            fractions[i] = fractions[i].add(fractions[i + 1]);
        }
    }

    private static void print(Fraction[] array) {
        System.out.println("-----");
        for (var i = 0; i < array.length; i++) {
            System.out.print(array[i].toString());
            if (i != array.length - 1) {
                System.out.print(" ");
            }
        }
        System.out.println("\n-----");
    }
}

```

Результат выполнения программы показан на рисунке 2.

```

k:
4
input m for Fraction No. 1:
1
input n for Fraction No. 1:
2
input m for Fraction No. 2:
3
input n for Fraction No. 2:
4
input m for Fraction No. 3:
5
input n for Fraction No. 3:
6
input m for Fraction No. 4:
7
input n for Fraction No. 4:
8
-----
1/2 3/4 5/6 7/8
-----
sum:
10/8
subtract:
-2/8
multiply:
3/8
divide:
4/6
-----
10/8 3/4 82/48 7/8
-----

```

Рисунок 2 – Результат выполнения программы

### Задание 3

House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: а) список квартир, имеющих заданное число комнат; б) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в



заданном промежутке; с) список квартир, имеющих площадь, превосходящую заданную.

Код написанной программы представлен в листинге 3.

### Листинг 3 – Программа для третьего задания

```
package com.java.lab;

import java.util.Scanner;

// House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица,
// Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: а)
// список квартир, имеющих заданное число комнат; б) список квартир,
// имеющих заданное число комнат и расположенных на этаже, который
// находится в заданном промежутке; с) список квартир, имеющих площадь,
// превосходящую заданную.
public class Main {

    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("n: ");
        var n = scanner.nextInt();
        var arr = new Flat[n];
        for(var i = 0; i < n; i++) {
            arr[i] = new Flat();
            System.out.print("input id for " + (i+1) + " flat: ");
            arr[i].setId(scanner.nextInt());
            System.out.print("input flatNum for " + (i+1) + " flat: ");
            arr[i].setFlatNum(scanner.nextInt());
            System.out.print("input S for " + (i+1) + " flat: ");
            arr[i].setS(scanner.nextDouble());
            System.out.print("input flour for " + (i+1) + " flat: ");
            arr[i].setFlour(scanner.nextInt());
            System.out.print("input roomNum for " + (i+1) + " flat: ");
            arr[i].setRoomNum(scanner.nextInt());
            System.out.print("input street for " + (i+1) + " flat: ");
            arr[i].setStreet(scanner.next());
            System.out.print("input type for " + (i+1) + " flat: ");
            arr[i].setType(scanner.next());
            System.out.print("input duration for " + (i+1) + " flat: ");
            arr[i].setDuration(scanner.nextInt());
        }

        System.out.print("count roomNum: ");
        var countRoomNum = scanner.nextInt();
        for (var i = 0; i < n; i++) {
            if (arr[i].getRoomNum() == countRoomNum) {
                System.out.println();
                arr[i].printFlat();
                System.out.println();
            }
        }
    }
}
```

```

        System.out.println();
        System.out.print("count roomNum and range flour");
        countRoomNum = scanner.nextInt();
        var rangeMin = scanner.nextInt();
        var rangeMax = scanner.nextInt();
        for (var i = 0; i < n; i ++) {
            if (arr[i].getRoomNum() == countRoomNum) {
                if (arr[i].getFlour() >= rangeMin && arr[i].getFlour()
<= rangeMax) {
                    System.out.println();
                    arr[i].printFlat();
                    System.out.println();
                }
            }
        }

        System.out.print("S: ");
        var S = scanner.nextDouble();
        for (int i = 0; i < n; i++) {
            if (arr[i].getS() >= S) {
                System.out.println();
                arr[i].printFlat();
                System.out.println();
            }
        }
    }
}

class Flat {
    private int id;
    private int flatNum;
    private double S;
    private int flour;
    private int roomNum;
    private String street;
    private String type;
    private int duration;

    public Flat() {
        id = 0;
        flatNum = 0;
        S = 0;
        flour = 0;
        roomNum = 0;
        street = "";
        type = "";
        duration = 0;
    }

    public Flat(int id, int flatNum, double S, int flour, int roomNum,
String street, String type, int duration) {
        this.id = id;
        this.flatNum = flatNum;
        this.S = S;
        this.flour = flour;
        this.roomNum = roomNum;
        this.street = street;
    }
}

```

```

        this.type = type;
        this.duration = duration;
    }

    public void printFlat() {
        System.out.println("id: " + id);
        System.out.println("flatNum: " + flatNum);
        System.out.println("S: " + S);
        System.out.println("flour: " + flour);
        System.out.println("roomNum: " + roomNum);
        System.out.println("street: " + street);
        System.out.println("type: " + type);
        System.out.println("duration: " + duration);

    }

    public double getS() {
        return S;
    }

    public void setS(double s) {
        S = s;
    }

    public int getFlour() {
        return flour;
    }

    public void setFlour(int flour) {
        this.flour = flour;
    }

    public int getRoomNum() {
        return roomNum;
    }

    public void setRoomNum(int roomNum) {
        this.roomNum = roomNum;
    }

    public String getStreet() {
        return street;
    }

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

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public int getDuration() {
        return duration;
    }

```

```

    }

    public void setDuration(int duration) {
        this.duration = duration;
    }

    public int getId() {
        return id;
    }

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

    public int getFlatNum() {
        return flatNum;
    }

    public void setFlatNum(int flatNum) {
        this.flatNum = flatNum;
    }
}

```

Результат выполнения программы показан на рисунке 3.

```

Project
Main
n: 1
input id for 1 flat: 1
input flatNum for 1 flat: 2
input S for 1 flat: 3
input flour for 1 flat: 4
input roomNum for 1 flat: 5
input street for 1 flat: st
input type for 1 flat: 2
input duration for 1 flat: 3
count roomNum: 5

id: 1
flatNum: 2
S: 3.0
flour: 4
roomNum: 5
street: st
type: 2
duration: 3

count roomNum and range flour
1
4

id: 1
flatNum: 2
S: 3.0
flour: 4
roomNum: 5
street: st
type: 2
duration: 3

S: 1

id: 1
flatNum: 2
S: 3.0
flour: 4
roomNum: 5
street: st
type: 2
duration: 3

```

Рисунок 3 – Результат выполнения программы

#### Задание 4

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; в) сведения об абонентах в алфавитном порядке.

Код написанной программы представлен в листинге 4.

#### Листинг 4 – Программа для четвертого задания

```
package com.java.lab;

import java.util.Arrays;
import java.util.Comparator;
import java.util.Scanner;

// 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; в) сведения об
// абонентах в алфавитном порядке.
public class Main {

    public static void main(String[] args) {
        var scanner = new Scanner(System.in);
        System.out.print("n: ");
        var n = scanner.nextInt();
        var arr = new Phone[n];
        for(var i = 0; i < n; i++) {
            arr[i] = new Phone();
            System.out.print("input id for " + (i+1) + " Phone: ");
            arr[i].setId(scanner.nextInt());
            System.out.print("input surname for " + (i+1) + " Phone: ");
            arr[i].setSurname(scanner.next());
            System.out.print("input name for " + (i+1) + " Phone: ");
            arr[i].setName(scanner.next());
            System.out.print("input lastname for " + (i+1) + " Phone: ");
            arr[i].setLastname(scanner.next());
            System.out.print("input address for " + (i+1) + " Phone: ");
            arr[i].setLastname(scanner.next());
            System.out.print("input cardNumber for " + (i+1) + " Phone: ");
            arr[i].setCardNumber(scanner.nextLong());
            System.out.print("input debit for " + (i+1) + " Phone: ");
            arr[i].setDebit(scanner.nextInt());
```

```

        System.out.print("input credit for " + (i+1) + " Phone:
");
        arr[i].setCredit(scanner.nextInt());
        System.out.print("input cityTime for " + (i+1) + " Phone:
");
        arr[i].setCityTime(scanner.nextInt());
        System.out.print("input outsideCityTime for " + (i+1) + "
Phone: ");
        arr[i].setOutsideCityTime(scanner.nextInt());
    }

    System.out.print("direction cityTime >: ");
    var direction = scanner.nextInt();
    for (var i = 0; i < n; i++) {
        if (arr[i].getCityTime() > direction) {
            System.out.println();
            arr[i].printPhone();
            System.out.println();
        }
    }

    System.out.println();
    System.out.print("use outsideCityTime");
    for (var i = 0; i < n; i++) {
        if (arr[i].getOutsideCityTime() > 0) {
            System.out.println();
            arr[i].printPhone();
            System.out.println();
        }
    }

    System.out.print("sorted phones: ");
    Arrays.stream(arr).sorted(new
PhoneComparator()).forEach(Phone::printPhone);

    }
}

class PhoneComparator implements Comparator<Phone> {

    public int compare(Phone a, Phone b){

        return
a.getSurname().toUpperCase().compareTo(b.getSurname().toUpperCase());
    }
}

class Phone {
    private int id;
    private String surname;
    private String name;
    private String lastname;
    private String address;
    private long cardNumber;
    private int debit;
    private int credit;
    private int cityTime;
    private int outsideCityTime;

```

```

    public Phone(int id, String surname, String name, String lastname,
String address, long cardNumber, int debit, int credit, int cityTime,
int outsideCityTime) {
        this.id = id;
        this.surname = surname;
        this.name = name;
        this.lastname = lastname;
        this.address = address;
        this.cardNumber = cardNumber;
        this.debit = debit;
        this.credit = credit;
        this.cityTime = cityTime;
        this.outsideCityTime = outsideCityTime;
    }

    public Phone() {
        this.id = 0;
        this.surname = "surname";
        this.name = "name";
        this.lastname = "lastname";
        this.address = "address";
        this.cardNumber = 0;
        this.debit = 0;
        this.credit = 0;
        this.cityTime = 0;
        this.outsideCityTime = 0;
    }

    public int getId() {
        return id;
    }

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

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getName() {
        return name;
    }

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

    public String getLastName() {
        return lastname;
    }

```

```

public void setLastname(String lastname) {
    this.lastname = lastname;
}

public String getAddress() {
    return address;
}

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

public long getCardNumber() {
    return cardNumber;
}

public void setCardNumber(long cardNumber) {
    this.cardNumber = cardNumber;
}

public int getDebit() {
    return debit;
}

public void setDebit(int debit) {
    this.debit = debit;
}

public int getCredit() {
    return credit;
}

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

public int getCityTime() {
    return cityTime;
}

public void setCityTime(int cityTime) {
    this.cityTime = cityTime;
}

public int getOutsideCityTime() {
    return outsideCityTime;
}

public void setOutsideCityTime(int outsideCityTime) {
    this.outsideCityTime = outsideCityTime;
}

public void printPhone() {
    System.out.println("id: " + id);
    System.out.println("surname: " + surname);
    System.out.println("name: " + name);
    System.out.println("lastname: " + lastname);
    System.out.println("address: " + address);
    System.out.println("cardNumber: " + cardNumber);
}

```

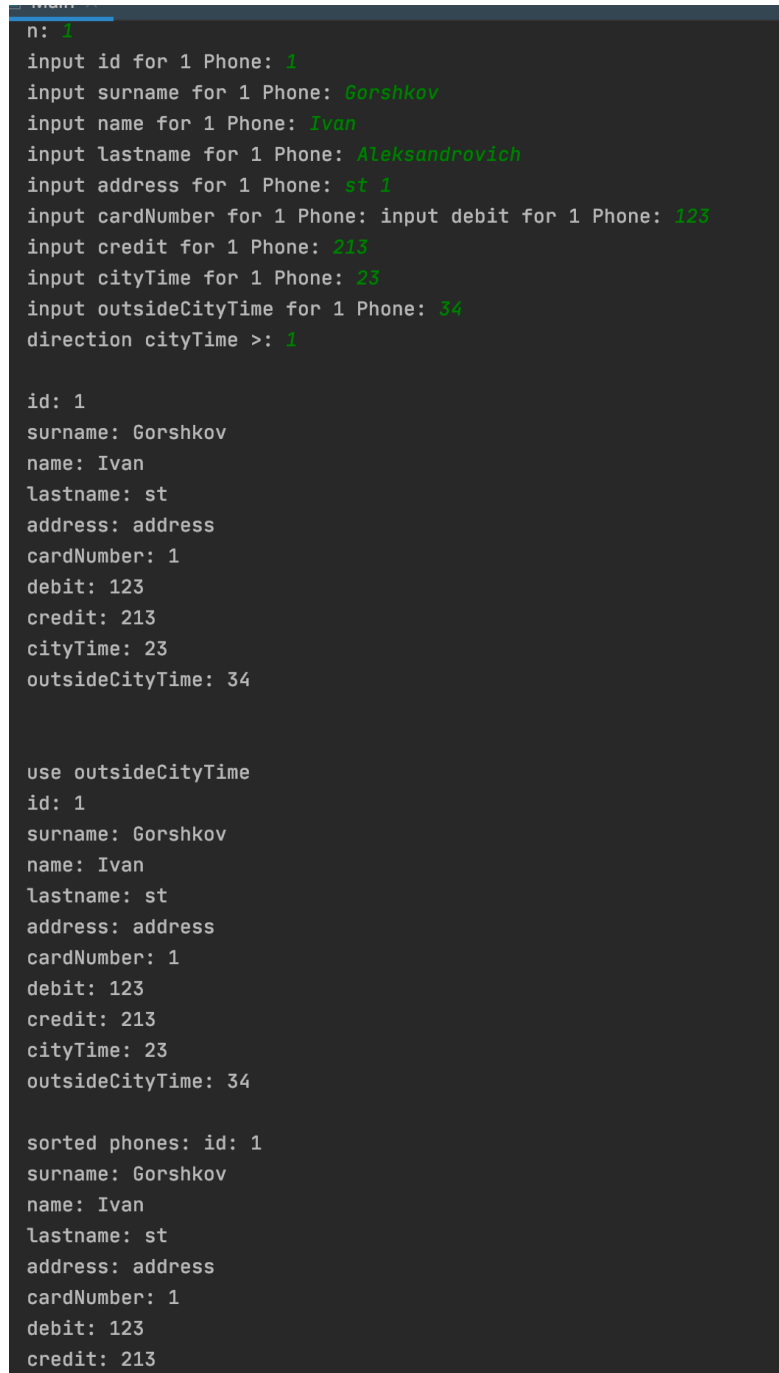


```

        System.out.println("debit: " + debit);
        System.out.println("credit: " + credit);
        System.out.println("cityTime: " + cityTime);
        System.out.println("outsideCityTime: " + outsideCityTime);
    }
}

```

Результат выполнения программы показан на рисунке 4.



```

Main
n: 1
input id for 1 Phone: 1
input surname for 1 Phone: Gorshkov
input name for 1 Phone: Ivan
input lastname for 1 Phone: Aleksandrovich
input address for 1 Phone: st 1
input cardNumber for 1 Phone: input debit for 1 Phone: 123
input credit for 1 Phone: 213
input cityTime for 1 Phone: 23
input outsideCityTime for 1 Phone: 34
direction cityTime >: 1

id: 1
surname: Gorshkov
name: Ivan
lastname: st
address: address
cardNumber: 1
debit: 123
credit: 213
cityTime: 23
outsideCityTime: 34

use outsideCityTime
id: 1
surname: Gorshkov
name: Ivan
lastname: st
address: address
cardNumber: 1
debit: 123
credit: 213
cityTime: 23
outsideCityTime: 34

sorted phones: id: 1
surname: Gorshkov
name: Ivan
lastname: st
address: address
cardNumber: 1
debit: 123
credit: 213

```

Рисунок 4 – Результат выполнения программы

## Задание 5

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

Код написанной программы представлен в листинге 5.

### Листинг 5 – Программа для пятого задания

```
package com.java.lab;
import java.util.Arrays;
import java.util.Objects;

// Создать объект класса Роза, используя классы Лепесток, Бутон.
Методы: расцвести, завясть, вывести на консоль цвет бутона.
public class Main {

    public static void main(String[] args) {
        Petal[] petals = new Petal[4];
        petals[0] = new Petal();
        petals[1] = new Petal();
        petals[2] = new Petal();
        petals[3] = new Petal();
        Rose rose = new Rose(new Bud(petals, "White"));
        rose.getColor();
        rose.withering();
        rose.blooming();
        System.out.println(rose.toString());
    }
}

class Rose {
    private final Bud bud;
    private Boolean isBloomed = true;

    public Rose(Bud bud) {
        this.bud = bud;
    }

    public void blooming() {
        this.bud.blooming();
        this.isBloomed = true;
        System.out.println("The rose has blossomed");
    }

    public void withering() {
        this.bud.withering();
        this.isBloomed = false;
        System.out.println("Rose withered");
    }

    public void getColor() {
        System.out.println("Bud color: " + this.bud.getColor());
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Rose rose)) return false;
        return bud.equals(rose.bud) &&
isBloomed.equals(rose.isBloomed);
    }
}
```

```

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

@Override
public String toString() {
    return "Rose {\n" +
        "bud: " + bud + ",\n" +
        "isBloomed: " + isBloomed + "\n" +
        "}";
}
}

class Bud {
    private Petal[] petal;
    private String color;
    private Boolean isBloomed = true;

    public Bud(Petal[] petal, String color) {
        this.petal = petal;
        this.color = color;
    }

    public String getColor() {
        return color;
    }

    public void blooming() {
        for (Petal value : petal) {
            value.blooming();
        }
        this.isBloomed = true;
        System.out.println("The bud has blossomed");
    }

    public void withering() {
        for (Petal value : petal) {
            value.withering();
        }
        this.isBloomed = false;
        System.out.println("Bud wilted");
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Bud bud)) return false;
        return Arrays.equals(petal, bud.petal) &&
color.equals(bud.color) && isBloomed.equals(bud.isBloomed);
    }

    @Override
    public int hashCode() {
        int result = Objects.hash(color, isBloomed);
        return result + Arrays.hashCode(petal);
    }
}

```

```

@Override
public String toString() {
    return "Bud {\n" +
        " petal: " + Arrays.toString(petal) + ",\n" +
        " color: '" + color + "'" + ",\n" +
        " isBloomed: " + isBloomed + "\n" +
        "}";
}
}

class Petal {
    private Boolean isBloomed = true;

    public void blooming() {
        this.isBloomed = true;
        System.out.println("petal blossomed");
    }

    public void withering() {
        this.isBloomed = false;
        System.out.println("petal withered");
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Petal petal)) return false;
        return Objects.equals(isBloomed, petal.isBloomed);
    }

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

    @Override
    public String toString() {
        return "Petal {" + " isBloomed:" + isBloomed + " }";
    }
}

```

Результат выполнения программы показан на рисунке 5.

```

Bud color: White
petal withered
petal withered
petal withered
petal withered
Bud wilted
Rose withered
petal blossomed
petal blossomed
petal blossomed
petal blossomed
The bud has blossomed
The rose has blossomed
Rose {
  bud: Bud {
    petal: [Petal { isBloomed:true }, Petal { isBloomed:true }, Petal { isBloomed:true }, Petal { isBloomed:true }],
    color: 'White',
    isBloomed: true
  },
  isBloomed: true
}

```

Рисунок 5 – Результат выполнения программы

### Задание 6

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

Код написанной программы представлен в листинге 6.

#### Листинг 6 – Программа для шестого задания

```

package com.java.lab;
import java.util.Arrays;
import java.util.Objects;

// Создать объект класса Дерево, используя классы Лист. Методы:
// зацвести, опать листьям, покрыться инеем, пожелтеть листьям.
public class Main {

    public static void main(String[] args) {
        Leaf[] leaves = new Leaf[4];
        leaves[0] = new Leaf(1);
        leaves[1] = new Leaf(2);
        leaves[2] = new Leaf(3);
        leaves[3] = new Leaf(4);
        Tree tree = new Tree(leaves);
        tree.blooming();
        tree.turningYellowLeaves();
        tree.fallingDown();
        tree.frostingOver();
        System.out.println(tree.toString());
    }
}

class Tree {
    private Leaf[] leaves;

    public Tree(Leaf[] leaves) {
        this.leaves = leaves;
    }
}

```

```

    public void blooming() {
        for (Leaf value : leaves) {
            value.blooming();
        }
        System.out.println("The tree's leaves has blossomed");
    }

    public void fallingDown() {
        for (Leaf value : leaves) {
            value.fallingDown();
        }
        System.out.println("The tree's leaves has falled down");
    }

    public void frostingOver() {
        for (Leaf value : leaves) {
            value.frostingOver();
        }
        System.out.println("The tree's leaves has frosted over");
    }

    public void turningYellowLeaves() {
        for (Leaf value : leaves) {
            value.turningYellowLeaves();
        }
        System.out.println("The tree's leaves has turned yellow");
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Tree tree)) return false;
        return Arrays.equals(leaves, tree.leaves);
    }

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

    @Override
    public String toString() {
        return "Rose {\n" +
            "leaves: " + Arrays.toString(leaves) + "\n" +
            "}";
    }
}

class Leaf {
    private int id;

    public Leaf(int id) {
        this.id = id;
    }

    public void blooming() {
        System.out.println("The Leaf has blossomed");
    }
}

```

```

    public void fallingDown() {
        System.out.println("The Leaf has falled down");
    }

    public void frostingOver() {
        System.out.println("The Leaf has frosted over");
    }

    public void turningYellowLeaves() {
        System.out.println("The Leaf has turned yellow");
    }

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

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

    @Override
    public String toString() {
        return "Leaf {" + " id:" + id + " }";
    }
}

```

Результат выполнения программы показан на рисунке 6.

```

The Leaf has blossomed
The Leaf has blossomed
The Leaf has blossomed
The Leaf has blossomed
The tree's leaves has blossomed
The Leaf has turned yellow
The Leaf has turned yellow
The Leaf has turned yellow
The Leaf has turned yellow
The tree's leaves has turned yellow
The Leaf has falled down
The Leaf has falled down
The Leaf has falled down
The Leaf has falled down
The tree's leaves has falled down
The Leaf has frosted over
The Leaf has frosted over
The Leaf has frosted over
The Leaf has frosted over
The tree's leaves has frosted over
Rose {
  leaves: [Leaf { id:1 }, Leaf { id:2 }, Leaf { id:3 }, Leaf { id:4 }]
}

```

Рисунок 6 – Результат выполнения программы

### Задание 7

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.

Код написанной программы представлен в листинге 7.

### Листинг 7 – Программа для седьмого задания

```

package com.java.lab;

// 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.
public class Main {

    public static void main(String[] args) {
        House house = new House(7);
        TZ tz = new TZ(20000, house);
        ConstrutorBuro constr = new ConstrutorBuro(tz);
    }
}

```



```

        System.out.println(constr.countCostString());
    }
}

class Brigada {
    private double price;
    private int n;

    public Brigada(double price, int n) {
        this.price = price;
        this.n = n;
    }

    public double getCost() {
        return n*price;
    }
}

class House {
    private final int floors;

    public House(int floors) {
        this.floors = floors;
    }

    public int getFloors() {
        return floors;
    }
}

class TZ {
    double moneyForFlour;
    House house;

    public TZ(double moneyForFlour, House house) {
        this.moneyForFlour = moneyForFlour;
        this.house = house;
    }

    public double getCost() {
        return house.getFloors() * moneyForFlour;
    }
}

class Designer {
    private final double price;

    public Designer(double price) {
        this.price = price;
    }

    public double getCost() {
        return price;
    }
}

```

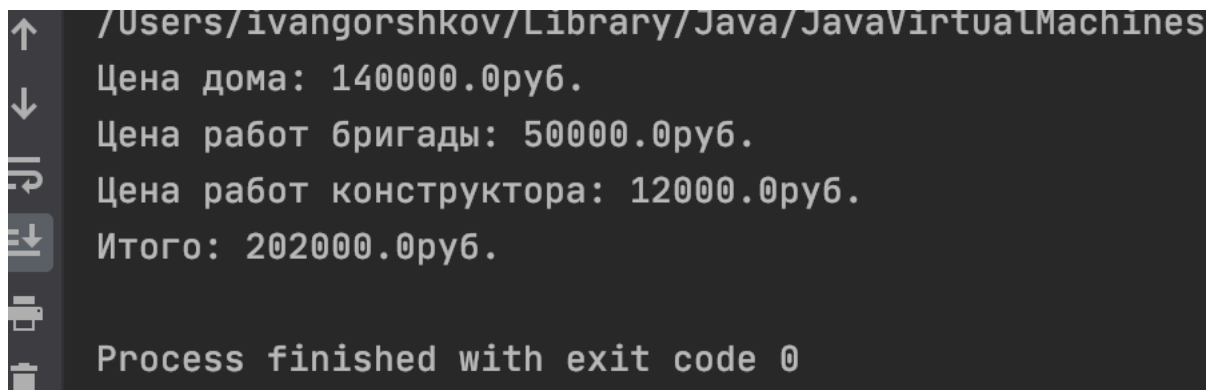
```
//класс Конструктор
class ConstructorBuro {
    TZ tz;
    Brigada brigada;
    Designer designer;
    double brigadaK = 0.5;
    double designerK = 0.6;

    public ConstructorBuro(TZ tz) {
        this.tz = tz;
        this.designer = new Designer(tz.moneyForFlour * designerK);
        this.brigada = new Brigada(tz.moneyForFlour * brigadaK, 5);
    }

    public double countCost() {
        return tz.getCost() + designer.getCost() + brigada.getCost();
    }

    public String countCostString() {
        double houseCost = tz.getCost();
        double brigadaCost = brigada.getCost();
        double designerCost = designer.getCost();
        return "Цена дома: " + houseCost + "руб.\n" +
            "Цена работ бригады: " + brigadaCost + "руб.\n" +
            "Цена работ конструктора: " + designerCost + "руб.\n"
+
            "Итого: " + countCost() + "руб.";
    }
}
```

Результат выполнения программы показан на рисунке 7.



```
/Users/ivangorshkov/Library/Java/JavaVirtualMachines
Цена дома: 140000.0руб.
Цена работ бригады: 50000.0руб.
Цена работ конструктора: 12000.0руб.
Итого: 202000.0руб.
Process finished with exit code 0
```

Рисунок 7 – Результат выполнения программы

## Задание 8

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.

Код написанной программы представлен в листинге 8.

Листинг 8 – Программа для четвертого задания

```

package com.java.lab;

// 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) {
        var admin = new Admin();
        var subscriber = new Subscriber("+799999999387", 123, true);
        admin.setSubscriber(subscriber);
        admin.updateNumber("+79999870140");
        admin.updateBalance(100);
        admin.updateService(false);
        admin.updateService(true);
        admin.updateBalance(-100);
        admin.disableServiceZeroBalance();
        System.out.println(subscriber.toString());
    }
}

class Subscriber {
    private String phoneNumber;
    private int balance;
    private boolean isServicesEnabled;

    public Subscriber(String phoneNumber, int balance, boolean
isServicesEnabled) {
        this.phoneNumber = phoneNumber;
        this.balance = balance;
        this.isServicesEnabled = isServicesEnabled;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public int getBalance() {
        return balance;
    }

    public void setBalance(int balance) {
        this.balance = balance;
    }

    public boolean isServicesEnabled() {
        return isServicesEnabled;
    }

    public void setServicesEnabled(boolean servicesEnabled) {
        isServicesEnabled = servicesEnabled;
    }
}

```

```

@Override
public String toString() {
    return "Subscriber{" +
        "phoneNumber='" + phoneNumber + '\'' +
        ", balance=" + balance +
        ", isServicesEnabled=" + isServicesEnabled +
        '}';
}
}

class Admin {
    Subscriber subscriber;

    public void setSubscriber(Subscriber subscriber) {
        this.subscriber = subscriber;
    }

    void updateNumber(String newPhoneNumber) {
        subscriber.setPhoneNumber(newPhoneNumber);
    }

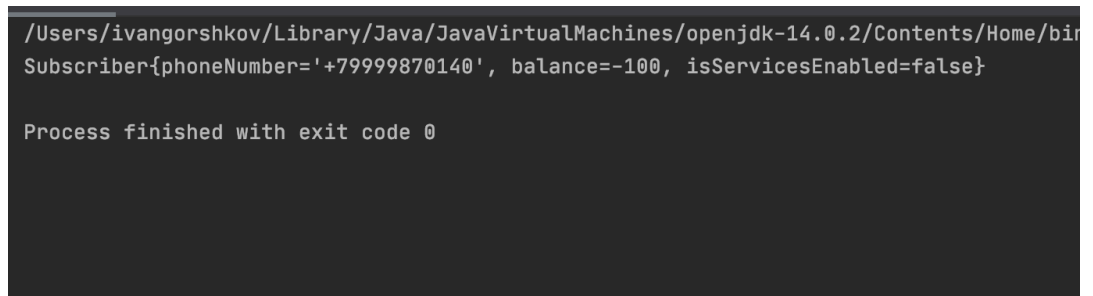
    void updateBalance(int newBalance) {
        subscriber.setBalance(newBalance);
    }

    void updateService(boolean servicesEnabled) {
        subscriber.setServicesEnabled(servicesEnabled);
    }

    void disableServiceZeroBalance() {
        if (subscriber.getBalance() <= 0){
            subscriber.setServicesEnabled(false);
        }
    }
}
}

```

Результат выполнения программы показан на рисунке 8.



```

/Users/ivangorshkov/Library/Java/JavaVirtualMachines/openjdk-14.0.2/Contents/Home/bin
Subscriber{phoneNumber='+79999870140', balance=-100, isServicesEnabled=false}

Process finished with exit code 0

```

Рисунок 8 – Результат выполнения программы

**Вывод:** В результате выполнения лабораторной работы были приобретены навыки работы с классами, наследованием и полиморфизмом на языке программирования Java.

Репозитории по ссылке <https://github.com/IvanGorshkov/Java-IU6-12M>.