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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ 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

Название: Лабораторная работа №3

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-22М (Группа) _____ (Подпись, дата) М.И. Замула (И.О. Фамилия)

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

Москва, 2023 г.

Вариант: 1

Задание 8

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.

Выполнение

Класс Complex

```
package lab3;

public class Complex {
    private double real;
    private double imag;

    public Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }

    public Complex() {
        this.real = 0;
        this.imag = 0;
    }

    public Complex addition(Complex complex) {
        double real = this.real + complex.real;
        double imag = this.imag + complex.imag;
        return new Complex(real, imag);
    }

    public Complex subtraction(Complex complex) {
        double real = this.real - complex.real;
        double imag = this.imag - complex.imag;
        return new Complex(real, imag);
    }

    public Complex multiplication(Complex complex) {
        double real = this.real * complex.real - this.imag * complex.imag;
        double imag = this.real * complex.imag + complex.real * this.imag;
        return new Complex(real, imag);
    }

    public Complex division(Complex complex) {

        double a = this.real;
        double b = this.imag;
        double c = complex.real;
        double d = complex.imag;

        double real = (a * c + b * d) / (c * c + d * d);
        double imag = (b * c - a * d) / (c * c + d * d);
```

```
        return new Complex(real, imag);
    }

    @Override
    public String toString() {
        return real + " + " + imag + "i";
    }
}
```

```
package lab3;

public class Complex {
    private double real;
    private double imag;

    public Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }

    public Complex() {
        this.real = 0;
        this.imag = 0;
    }

    public Complex addition(Complex complex) {
        double real = this.real + complex.real;
        double imag = this.imag + complex.imag;
        return new Complex(real, imag);
    }

    public Complex subtraction(Complex complex) {
        double real = this.real - complex.real;
        double imag = this.imag - complex.imag;
        return new Complex(real, imag);
    }

    public Complex multiplication(Complex complex) {
        double real = this.real * complex.real - this.imag * complex.imag;
        double imag = this.real * complex.imag + complex.real * this.imag;
        return new Complex(real, imag);
    }

    public Complex division(Complex complex) {

        double a = this.real;
        double b = this.imag;
        double c = complex.real;
        double d = complex.imag;

        double real = (a * c + b * d) / (c * c + d * d);
        double imag = (b * c - a * d) / (c * c + d * d);
        return new Complex(real, imag);
    }

    @Override
    public String toString() {
        return real + " + " + imag + "i";
    }
}
```

Main

```java
package lab3;

public class Lab3_1_8 {
    public static void main(String[] args) {
        Complex vector01 = new Complex(10.0, 15.0);
        Complex vector02 = new Complex(24.3, 23.11);
        Complex result = vector01.addition(vector02);
        System.out.print(result);
    }
}
```

Результаты

```
34.3 + 38.11i
Process finished with exit code 0
```

Задание 9

Определить класс Квадратное уравнение. Класс должен содержать несколько конструкторов. Реализовать методы для поиска корней, экстремумов, а также интервалов убывания/возрастания. Создать массив объектов и определить наибольшие и наименьшие по значению корни.

Выполнение

Класс Квадратное уравнение

```java
package lab3;

public class QuadraticEquation {

    private double a;
    private double b;
    private double c;

    public QuadraticEquation(double a, double b, double c) {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public QuadraticEquation(double a, double b) {
        this.a = a;
        this.b = b;
        this.c = 0;
    }
```

```
    public QuadraticEquation(double a) {
        this.a = a;
        this.b = 0;
        this.c = 0;
    }

    public double[] findRoot(){
        double discriminant = b * b - 4 * a * c;
        if (a == 0 || discriminant < 0) {
            return new double[0]; // возвращаем пустой массив для комплексных
корней
        }
        if (discriminant == 0) {
            double root = -b / (2 * a);
            return new double[]{root};
        }
        double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
        double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
        return new double[]{root1, root2};

    }

    public double[] findExtrema() {
        if (a == 0) {
            return new double[0];
        }
        double x = -b / (2 * a);
        double y = a * x * x + b * x + c;
        return new double[]{x, y};
    }

    public void findIntervals() {
        if (a == 0) {
            if (b > 0) {
                System.out.println("Функция возрастает при любом значении
x");
            } else if (b < 0) {
                System.out.println("Функция убывает при любом значении x");
            } else {
                System.out.println("Функция не имеет интервалов возрастания и
убывания");
            }
        } else {
            double x = -b / (2 * a);
            if (a > 0) {
                System.out.println("Функция убывает в промежутке x < " + x);
                System.out.println("Функция возрастает в промежутке x > " +
x);
            } else {
                System.out.println("Функция возрастает в промежутке x < " +
x);
                System.out.println("Функция убывает в промежутке x > " + x);
            }


        }
    }



    }
```

Main

```
package lab3;

public class Lab3_1_9 {

    public static void main(String[] args) {
        QuadraticEquation q1 = new QuadraticEquation(1, 2, -3.0);
        QuadraticEquation q2 = new QuadraticEquation(10, 12, -100);
        QuadraticEquation[] array = new QuadraticEquation[]{q1, q2};

        System.out.println("Максимальный корень = " + String.format("%.4f",
findMaxRoot(array)));
        System.out.println("Минимальный корень = " + String.format("%.4f",
findMinRoot(array)));
    }


    public static double findMaxRoot(QuadraticEquation[] array) {
        double currentMaxRoot = Double.NEGATIVE_INFINITY;

        for (QuadraticEquation quadraticEquation : array) {
            double[] currentRoots = quadraticEquation.findRoot();
            for (double currentRoot : currentRoots) {
                if (currentRoot > currentMaxRoot) {
                    currentMaxRoot = currentRoot;
                }
            }
        }
        return currentMaxRoot;
    }

    public static double findMinRoot(QuadraticEquation[] array) {
        double currentMinRoot = Double.POSITIVE_INFINITY;

        for (QuadraticEquation quadraticEquation : array) {
            double[] currentRoots = quadraticEquation.findRoot();
            for (double currentRoot : currentRoots) {
                if (currentRoot < currentMinRoot) {
                    currentMinRoot = currentRoot;
                }
            }
        }
        return currentMinRoot;
    }
}
```

Результаты

```
Максимальный корень = 2,6187
Минимальный корень = -3,8187
```

Вариант: 2

Задание 8

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.

Car: id, Марка, Модель, Год выпуска, Цвет, Цена, Регистрационный номер. Создать массив объектов. Вывести: a) список автомобилей заданной марки; b) список автомобилей заданной модели, которые эксплуатируются больше n лет; c) список автомобилей заданного года выпуска, цена которых больше указанной.

Выполнение

Класс Car

```
package lab3;

import java.time.Year;
import java.util.Objects;

public class Car {
    private int id;
    private String mark;
    private String model;
    private int year;
    private String color;
    private double price;
    private String number;

    public Car(int id, String mark, String model, int year, String color,
double price, String number) {
        this.id = id;
        this.mark = mark;
        this.model = model;
        this.year = year;
        this.color = color;
        this.price = price;
        this.number = number;
    }

    public int getId() {
        return id;
    }
```

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

    public String getMark() {
        return mark;
    }

    public void setMark(String mark) {
        this.mark = mark;
    }

    public String getModel() {
        return model;
    }

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

    public int getYear() {
        return year;
    }

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

    public String getColor() {
        return color;
    }

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

    public double getPrice() {
        return price;
    }

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

    public String getNumber() {
        return number;
    }

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

    @Override
    public String toString() {
        return  "Автомобиль: " + "id=" + id +
                ", mark='" + mark + '\'' +
                ", model='" + model + '\'' +
                ", year=" + year +
                ", color='" + color + '\'' +
                ", price=" + price +
                ", number=" + number;
    }
```

```
    public boolean isMark(String mark) {
        return mark.equals(this.mark);
    }

    public boolean isModelAndNYear(String model, int nYear) {
        return model.equals(this.model) && (Year.now().getValue() -
this.year) <= nYear;
    }

    public boolean isYearAndPrice(int currentYear, double currentPrice) {
        return this.year == currentYear && this.price > currentPrice;
    }
}
```

Main

```
package lab3;

import java.util.ArrayList;
import java.util.List;

public class Lab3_2_8 {
    public static void main(String[] args) {

        Car[] cars = new Car[]{
                new Car(1, "Ford", "Galaxy", 2010, "Blue", 1.6, "A123BC77"),
                new Car(2, "Renault", "Logan", 2010, "Red", 0.3, "E456HK77"),
                new Car(3, "Toyota", "Camry", 2008, "Black", 1.2,
"M007AB77"),
                new Car(4, "Ford", "Focus", 2015, "Grey", 1.3, "C223OT77"),
                new Car(5, "Ford", "Explorer", 2021, "Black", 3.5,
"A777AA77"),
                new Car(6, "Toyota", "Camry", 2010, "White", 1.7,
"P482CX77"),
                new Car(7, "Toyota", "Camry", 2010, "Green", 1.9,
"K911EC77"),
        };

        List<Car> fords = listMarks(cars, "Ford");
        System.out.println("Автомобили марки Ford: ");
        for (Car ford : fords) {
            System.out.println(ford);
        }

System.out.println("**********************************************************
*********");

        List<Car> camry2010 = listModelAndYear(cars, "Camry", 13);
        System.out.println("Автомобили модели Camry, младше 13 лет: ");
        for (Car car : camry2010) {
            System.out.println(car);
        }

System.out.println("**********************************************************
*********");

        List<Car> expansiveCars = listYearAndPrice(cars, 2010, 1.5);
        System.out.println("Автомобили 2010 года, дороже 1.5 млн: ");
```

```
        for (Car car : expansiveCars) {
            System.out.println(car);
        }

    }

    public static List<Car> listMarks(Car[] cars, String mark){
        List<Car> result = new ArrayList<>();
        for (Car car : cars) {
            if (car.isMark(mark)) {
                result.add(car);
            }
        }
        return result;
    }

    public static List<Car> listModelAndYear(Car[] cars, String model, int
nYear){
        List<Car> result = new ArrayList<>();
        for (Car car : cars) {
            if (car.isModelAndNYear(model, nYear)) {
                result.add(car);
            }
        }
        return result;
    }

    public static List<Car> listYearAndPrice(Car[] cars, int year, double
price){
        List<Car> result = new ArrayList<>();
        for (Car car : cars) {
            if (car.isYearAndPrice(year, price)) {
                result.add(car);
            }
        }
        return result;
    }
}
```

Результаты

```
Автомобили марки Ford:
Автомобиль: id=1, mark='Ford', model='Galaxy', year=2010, color='Blue', price=1.6, number=A123BC77
Автомобиль: id=4, mark='Ford', model='Focus', year=2015, color='Grey', price=1.3, number=C223OT77
Автомобиль: id=5, mark='Ford', model='Explorer', year=2021, color='Black', price=3.5, number=A777AA77
************************************************************
Автомобили модели Camry, младше 13 лет:
Автомобиль: id=6, mark='Toyota', model='Camry', year=2010, color='White', price=1.7, number=P482CX77
Автомобиль: id=7, mark='Toyota', model='Camry', year=2010, color='Green', price=1.9, number=K911EC77
************************************************************
Автомобили 2010 года, дороже 1.5 млн:
Автомобиль: id=1, mark='Ford', model='Galaxy', year=2010, color='Blue', price=1.6, number=A123BC77
Автомобиль: id=6, mark='Toyota', model='Camry', year=2010, color='White', price=1.7, number=P482CX77
Автомобиль: id=7, mark='Toyota', model='Camry', year=2010, color='Green', price=1.9, number=K911EC77
```

Задание 9

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.

Product: id, Наименование, UPC, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести: a) список товаров для заданного наименования; b) список товаров для заданного наименования, цена которых не превосходит заданную; c) список товаров, срок хранения которых больше заданного.

Выполнение

Класс Product

```
package lab3;

public class Product {
    private int id;
    private String name;
    private String upc;
    private String manufacturer;
    private double price;
    private double shelfLife;
    private int quantify;

    public Product(int id, String name, String upc, String manufacturer,
double price, double shelfLife, int quantify) {
        this.id = id;
        this.name = name;
        this.upc = upc;
        this.manufacturer = manufacturer;
        this.price = price;
        this.shelfLife = shelfLife;
        this.quantify = quantify;
    }

    public int getId() {
        return id;
    }

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

    public String getName() {
```

```
        return name;
    }

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

    public String getUpc() {
        return upc;
    }

    public void setUpc(String upc) {
        this.upc = upc;
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public double getPrice() {
        return price;
    }

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

    public double getShelfLife() {
        return shelfLife;
    }

    public void setShelfLife(double shelfLife) {
        this.shelfLife = shelfLife;
    }

    public int getQuantify() {
        return quantify;
    }

    public void setQuantify(int quantify) {
        this.quantify = quantify;
    }

    @Override
    public String toString() {
        return "Product: " +
                "id=" + id +
                ", name='" + name + '\'' +
                ", upc=" + upc +
                ", manufacturer='" + manufacturer + '\'' +
                ", price=" + price +
                ", shelfLife=" + shelfLife +
                ", quantify=" + quantify +
                '}';
    }

    public boolean isName(String name) {
        return name.equals(this.name);
    }
```

```
    public boolean isNameAndPrice(String currentName, double currentPrice) {
        return (currentName.equals(this.name) && (this.price <=
currentPrice));
    }

    public boolean isShelfLife(double shelfLife) {
        return this.shelfLife > shelfLife;
    }

}
```

Main

```
package lab3;

import java.util.ArrayList;
import java.util.List;

public class Lab3_2_9 {
    public static void main(String[] args) {
        Product[] products = new Product[]{
                new Product(1, "lipstick", "12345600005", "MAC", 1.7, 1.5,
148),
                new Product(2, "mascara", "21735600006", "INGLOT", 2.2, 1.0,
11),
                new Product(3, "blush", "086320600007", "MAC", 1.9, 3.3,
221),
                new Product(4, "cream", "002638600003", "PAYOT", 6.0, 1.4,
93),
                new Product(5, "shadow", "015338600009", "MAC", 4.5, 2.5,
160),
                new Product(6, "lipstick", "008437200003", "SCINIC", 0.7,
1.5, 20),
                new Product(7, "lipstick", "076649900004", "MAC", 1.4, 2.0,
45),
        };

        List<Product> lipsticks = listName(products, "lipstick");
        System.out.println("Lipsticks: ");
        for (Product lipstick : lipsticks) {
            System.out.println(lipstick);
        }

System.out.println("**********************************************************
*********");

        List<Product> cheapLipsticks = listNameAndPrice(products, "lipstick",
1.5);
        System.out.println("Lipsticks cheaper 1.5: ");
        for (Product product : cheapLipsticks) {
            System.out.println(product);
        }

System.out.println("**********************************************************
*********");

        List<Product> shelfLifeProducts = listShelfLife(products, 1.6);
        System.out.println("Products with a shelf life of more than 1.6
```

```
years: ");
        for (Product product : shelfLifeProducts) {
            System.out.println(product);
        }

    }

    public static List<Product> listName(Product[] products, String name){
        List<Product> result = new ArrayList<>();
        for (Product product : products) {
            if (product.isName(name)) {
                result.add(product);
            }
        }
        return result;
    }

    public static List<Product> listNameAndPrice(Product[] products, String
name, double price){
        List<Product> result = new ArrayList<>();
        for (Product product : products) {
            if (product.isNameAndPrice(name, price)) {
                result.add(product);
            }
        }
        return result;
    }

    public static List<Product> listShelfLife(Product[] products, double
shelfLife){
        List<Product> result = new ArrayList<>();
        for (Product product : products) {
            if (product.isShelfLife(shelfLife)) {
                result.add(product);
            }
        }
        return result;
    }


}
```

Результаты

```
Lipsticks:
Product: id=1, name='lipstick', upc=12345600005, manufacturer='MAC', price=1.7, shelfLife=1.5, quantify=148}
Product: id=6, name='lipstick', upc=008437200003, manufacturer='SCINIC', price=0.7, shelfLife=1.5, quantify=20}
Product: id=7, name='lipstick', upc=076649900004, manufacturer='MAC', price=1.4, shelfLife=2.0, quantify=45}
****************************************************************
Lipsticks cheaper 1.5:
Product: id=6, name='lipstick', upc=008437200003, manufacturer='SCINIC', price=0.7, shelfLife=1.5, quantify=20}
Product: id=7, name='lipstick', upc=076649900004, manufacturer='MAC', price=1.4, shelfLife=2.0, quantify=45}
****************************************************************
Products with a shelf life of more than 1.6 years:
Product: id=3, name='blush', upc=086320600007, manufacturer='MAC', price=1.9, shelfLife=3.3, quantify=221}
Product: id=5, name='shadow', upc=015338600009, manufacturer='MAC', price=4.5, shelfLife=2.5, quantify=160}
Product: id=7, name='lipstick', upc=076649900004, manufacturer='MAC', price=1.4, shelfLife=2.0, quantify=45}
```

Вариант: 3

Задание 8

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.

Выполнение

Класс Пианино

```
package lab3;

import java.util.Arrays;
import java.util.Objects;
import java.util.concurrent.TimeUnit;

public class Piano {

    private PianoKey[] keys;

    public Piano(PianoKey[] keys) {
        this.keys = keys;
    }

    public PianoKey[] getKeys() {
        return keys;
    }

    public void setKeys(PianoKey[] keys) {
        this.keys = keys;
    }

    @Override
    public String toString() {
        return "Piano {" +
                "keys=" + Arrays.toString(keys) +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Piano piano = (Piano) o;
        return Arrays.equals(keys, piano.keys);
```

```
    }

    @Override
    public int hashCode() {
        return Arrays.hashCode(keys);
    }

    public void tunePiano(double[] freqs) {
        for (int i = 0; i < freqs.length; i++) {
            this.keys[i].setFreq(freqs[i]);
        }
    }

    public void pressKey(double freq) throws InterruptedException {
        for (PianoKey key : keys) {
            if (key.hashCode() == Objects.hash(freq)) {
                key.setPressed(true);
                TimeUnit.SECONDS.sleep(1);
                key.setPressed(false);
            }
            break;
        }
    }

    public void playPiano(double[] freqs) throws InterruptedException {
        for (double freq : freqs) {
            pressKey(freq);
            System.out.println("Нажата клавиша следующей частоты: " + freq);
        }
    }
}
```

Класс Клавиша

```
package lab3;

import java.util.Objects;

public class PianoKey {
    private double freq;

    private boolean isPressed;


    public PianoKey(double freq) {
        this.freq = freq;
        this.isPressed = false;
    }

    public double getFreq() {
        return freq;
    }

    public void setFreq(double freq) {
        this.freq = freq;
    }

    public boolean isPressed() {
        return isPressed;
```

```
    }

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

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        PianoKey pianoKey = (PianoKey) o;
        return pianoKey.freq == getFreq();
    }

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

    @Override
    public String toString() {
        return "Piano Key :" +
                "freq=" + freq;
    }

}
```

Main

```
package lab3;

public class Lab3_3_8 {
    public static void main(String[] args) throws InterruptedException {
        double[] arrayTunedKeys =   {261.63, 277.18, 293.66, 311.13, 329.63,
349.23, 369.99, 392.00,
                415.30, 440.00, 466.16, 493.88};
        double[] arrayMusic = {261.63, 440.00, 440.00, 415.30, 440.00,
369.99, 293.66, 261.63};

        PianoKey[] pianoKeys = new PianoKey[]{
                new PianoKey(261),
                new PianoKey(277),
                new PianoKey(293),
                new PianoKey(311),
                new PianoKey(329),
                new PianoKey(349),
                new PianoKey(369),
                new PianoKey(392),
                new PianoKey(415),
                new PianoKey(440),
                new PianoKey(466),
                new PianoKey(493)
        };


        Piano casio = new Piano(pianoKeys);

        System.out.println(casio);
```

```
        casio.tunePiano(arrayTunedKeys);

        System.out.println(casio);

        casio.playPiano(arrayMusic);



    }


}



//        Си первой октавы 493.88
//         A#   Ля-диез первой октавы  466.16
//         A    Ля первой октавы   440.00
//         G#   Соль-диез первой октавы    415.30
//         G    Соль первой октавы 392.00
//         F#   Фа-диез первой октавы  369.99
//         F    Фа первой октавы   349.23
//         E    Ми первой октавы   329.63
//         D#   Ре-диез первой октавы  311.13
//         D    Ре первой октавы   293.66
//         C#   До-диез первой октавы  277.18
//         C    До первой октавы   261.63
```

Результаты

```
Нажата клавиша следующей частоты: 261.63
Нажата клавиша следующей частоты: 440.0
Нажата клавиша следующей частоты: 440.0
Нажата клавиша следующей частоты: 415.3
Нажата клавиша следующей частоты: 440.0
Нажата клавиша следующей частоты: 369.99
Нажата клавиша следующей частоты: 293.66
Нажата клавиша следующей частоты: 261.63
```

Задание 9

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 Фотоальбом, используя класс Фотография. Методы: задать название фотографии, дополнить фотоальбом фотографией, вывести на консоль количество фотографий.

Выполнение

Класс Фотоальбом

```
package lab3;

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

public class PhotoAlbum {
    private String name;
    private final List<Photo> photos;

    public PhotoAlbum(String name) {
        this.name = name;
        this.photos = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

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

    public void addPhoto(Photo photo) {
        photos.add(photo);
    }

    public int getNumberOfPhotos() {
        return photos.size();
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof PhotoAlbum that)) return false;
        return Objects.equals(getName(), that.getName()) &&
                Objects.equals(photos, that.photos);
    }

    @Override
    public int hashCode() {
        return Objects.hash(getName(), photos);
    }
```

```
    @Override
    public String toString() {
        return "PhotoAlbum: " +
                "name='" + name + '\'' +
                ", photos=" + photos +
                '}';
    }
}
```

Класс Фотография

```
package lab3;

import java.util.Objects;

public class Photo {
    private String name;

    public Photo(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

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

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Photo photo)) return false;
        return Objects.equals(getName(), photo.getName());
    }

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

    @Override
    public String toString() {
        return "Photo: " +
                "name='" + name + '\'' +
                '}';
    }
}
```

Main

```
package lab3;

public class Lab3_3_9 {

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

```
        Photo photo1 = new Photo("Turkey 2023");
        Photo photo2 = new Photo("BMSTU 2022");
        Photo photo3 = new Photo("2003");
        Photo photo4 = new Photo("Yaroslavl 2021");
        Photo photo5 = new Photo("Dancing 2007");

        PhotoAlbum album = new PhotoAlbum("Rita album");
            album.addPhoto(photo1);
            album.addPhoto(photo2);
            album.addPhoto(photo3);
            album.addPhoto(photo4);
            album.addPhoto(photo5);

            System.out.println("Number of photos in Rita album: " +
album.getNumberOfPhotos());
    }
}
```

Результаты

```
Number of photos in Rita album: 5
```

Вариант: 4

Задание 8

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.

Выполнение

Класс Диспетчер

```
package lab3;

import java.util.ArrayList;

public class Dispatcher {

    private final ArrayList<Driver> drivers;
```

```
    private final ArrayList<CarForAutobase> cars;
    private String result;

    public Dispatcher() {
        this.drivers = new ArrayList<>();
        this.cars = new ArrayList<>();
    }

    public void addDriver(Driver driver) {
        drivers.add(driver);
    }

    public void removeDriver(Driver driver) {
        drivers.remove(driver);
    }

    public void addCar(CarForAutobase car) {
        cars.add(car);
    }

    public void removeCar(CarForAutobase car) {
        cars.remove(car);
    }

    public void assignDriverToCar(Driver driver, CarForAutobase car) {
        if (driver.getCar() != null) {
            driver.getCar().setNeedsRepair(true);
        }
        driver.assignCar(car);
        driver.setAvailable(false);
    }

    public void markTripComplete(Driver driver) {
        driver.setAvailable(true);
        driver.getCar().setNeedsRepair(false);
    }

    public void markDriverForRepair(Driver driver) {
        driver.setNeedsRepair(true);
        driver.setAvailable(false);
    }

    public void suspendDriver(Driver driver) {
        driver.setOnDuty(false);
        driver.setAvailable(false);
        driver.getCar().setNeedsRepair(false);
    }

    public void reinstateDriver(Driver driver) {
        driver.setOnDuty(true);
    }

    public String getDrivers() {
        result = "";
        this.drivers.forEach(driver -> result += driver.toString() + "\n");
        return result;
    }

    public String getCars() {
        result = "";
        this.cars.forEach(car -> result += car.toString() + "\n");
        return result;
    }
```

```
}
```

Класс Водитель

```
package lab3;

class Driver {
    private final String name;
    private boolean available;
    private boolean onDuty;
    private boolean needsRepair;
    private CarForAutobase car;

    public Driver(String name) {
        this.name = name;
        this.available = true;
        this.onDuty = false;
        this.needsRepair = false;
        this.car = null;
    }

    public void setOnDuty(boolean onDuty) {
        this.onDuty = onDuty;
    }

    public void setAvailable(boolean available) {
        this.available = available;
    }

    public void setNeedsRepair(boolean needsRepair) {
        this.needsRepair = needsRepair;
    }

    public void assignCar(CarForAutobase car) {
        this.car = car;
    }

    public boolean isAvailable() {
        return available;
    }

    public boolean isOnDuty() {
        return onDuty;
    }

    public boolean needsRepair() {
        return needsRepair;
    }

    public CarForAutobase getCar() {
        return car;
    }

    public String toString() {
        return "{ name: " + this.name + ", available: " + this.available + ",
onDuty: " + this.onDuty + ", needsRepair: " + this.needsRepair + (this.car !=
null ? ", car: " + this.car : "") + "}";
```

```
    }
}
```

Класс Автомобиль

```
package lab3;

public class CarForAutobase {
    private final String mark;
    private final String model;
    private final int year;
    private boolean needsRepair;

    public CarForAutobase(String mark, String model, int year) {
        this.mark = mark;
        this.model = model;
        this.year = year;
        this.needsRepair = false;
    }

    public void setNeedsRepair(boolean needsRepair) {
        this.needsRepair = needsRepair;
    }

    public boolean needsRepair() {
        return needsRepair;
    }

    public String getMark() {
        return mark;
    }

    public String getModel() {
        return model;
    }

    public int getYear() {
        return year;
    }

    public String toString() {
        return "{ mark: " + this.mark + ", model: " + this.model + ", year: "
+ this.year + ", needsRepair: " + this.needsRepair;
    }

}
```

Main

```
package lab3;

public class Lab3_4_8 {
    public static void main(String[] args) {
        Dispatcher dispatcher = new Dispatcher();
```

```
        Driver d1 = new Driver("Rita");
        Driver d2 = new Driver("MyCar");
        Driver d3 = new Driver("Nargiz");

        dispatcher.addDriver(d1);
        dispatcher.addDriver(d2);
        dispatcher.addDriver(d3);

        CarForAutobase c1 = new CarForAutobase("Ford", "Galaxy", 2008);
        CarForAutobase c2 = new CarForAutobase("Renaut", "Logan", 2012);
        CarForAutobase c3 = new CarForAutobase("Audi", "TT", 2015);

        dispatcher.addCar(c1);
        dispatcher.addCar(c2);
        dispatcher.addCar(c3);

        //назначить водителя1 на автомобиль1
        dispatcher.assignDriverToCar(d1, c1);

        //пометить поездку водителя1 как завершенную
        dispatcher.markTripComplete(d1);

        //поставить машину водителя1 на ремонт
        dispatcher.markDriverForRepair(d1);

        //отстранить водителя1
        dispatcher.suspendDriver(d1);

        dispatcher.assignDriverToCar(d2, c2);

        dispatcher.markTripComplete(d2);

        dispatcher.suspendDriver(d2);

        dispatcher.reinstateDriver(d2);

        System.out.println(dispatcher.getDrivers());
        System.out.println(dispatcher.getCars());
    }
}
```

Результат

```
{ name: Rita, available: false, onDuty: false, needsRepair: true, car: { mark: Ford, model: Galaxy, year: 2008, needsRepair: false}
{ name: MyCar, available: false, onDuty: true, needsRepair: false, car: { mark: Renaut, model: Logan, year: 2012, needsRepair: false}
{ name: Nargiz, available: true, onDuty: false, needsRepair: false}

{ mark: Ford, model: Galaxy, year: 2008, needsRepair: false
{ mark: Renaut, model: Logan, year: 2012, needsRepair: false
{ mark: Audi, model: TT, year: 2015, needsRepair: false
```

Вариант: 4

Задание 9

Построить модель программной системы.

Система Интернет-магазин. Администратор добавляет информацию о Товаре. Клиент делает и оплачивает Заказ на Товары. Администратор регистрирует Продажу и может занести неплательщиков в «черный список».

Выполнение

Класс Администратор

```
package lab3;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class Admin {
    private final List<OnlineProduct> products;
    private final List<Sale> sales;
    private final Blacklist blacklist;

    public Admin() {
        products = new ArrayList<>();
        sales = new ArrayList<>();
        blacklist = new Blacklist();
    }

    public void addProduct(OnlineProduct product) {
        products.add(product);
    }

    public void registerSale(Order order) {
        Sale sale = new Sale(order, new Date());
        sales.add(sale);
    }

    public void addCustomerToBlacklist(String customer) {
        blacklist.addCustomer(customer);
    }

    public boolean isCustomerBlacklisted(String customer) {
        return blacklist.isBlacklisted(customer);
    }
}
```

Класс Онлайн продукт

```
package lab3;

public class OnlineProduct {

    private String name;
    private String description;
    private double price;

    public OnlineProduct(String name, String description, double price) {
```

```
        this.name = name;
        this.description = description;
        this.price = price;
    }

    public String getName() {
        return name;
    }

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

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getPrice() {
        return price;
    }

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

Класс Покупатель

```
package lab3;

import java.util.ArrayList;
import java.util.List;

public class Customer {
    private final String name;
    private final List<OnlineProduct> cart;
    private Order currentOrder;

    public Customer(String name, String email) {
        this.name = name;
        cart = new ArrayList<>();
    }

    public void addToCart(OnlineProduct product) {
        cart.add(product);
    }

    public void checkout() {
        currentOrder = new Order(cart);
        // реализация оплаты заказа
        currentOrder.pay();
    }

    public Order getCurrentOrder() {
        return currentOrder;
```

```
    }

    public String getName() {
        return name;
    }
}
```

Класс Заказ

```
package lab3;

import java.util.List;

public class Order {
    private final List<OnlineProduct> products;
    private double totalPrice;
    private boolean isPaid;

    public Order(List<OnlineProduct> products) {
        this.products = products;
        calculateTotalPrice();
    }

    public double getTotalPrice() {
        return totalPrice;
    }

    public boolean isPaid() {
        return isPaid;
    }

    public void pay() {
        isPaid = true;
    }

    private void calculateTotalPrice() {
        totalPrice = 0;
        for (OnlineProduct product : products) {
            totalPrice += product.getPrice();
        }
    }
}
```

Класс Черный список

```
package lab3;


import java.util.ArrayList;
import java.util.List;

public class Blacklist {
    private final List<String> customers;

    public Blacklist() {
```

```
        this.customers = new ArrayList<>();
    }

    public Blacklist(List<String> customers) {
        this.customers = customers;
    }

    public void addCustomer(String customer) {
        customers.add(customer);
    }

    public boolean isBlacklisted(String customer) {
        return customers.contains(customer);
    }
}
```

Main

```
package lab3;

public class Lab3_4_9 {
    public static void main(String[] args) {
        Admin admin = new Admin();

        // добаляю товары
        OnlineProduct p1 = new OnlineProduct("iPhone 13", "Smartphone Apple",
999.99);
        OnlineProduct p2 = new OnlineProduct("Samsung Galaxy S21",
"Smartphone Samsung", 799.99);
        admin.addProduct(p1);
        admin.addProduct(p2);

        // создаю клиентов
        Customer c1 = new Customer("Rita", "rita@gmail.com");
        Customer c2 = new Customer("Mycar", "mycar@gmail.com");

        // добавляю товары в корзину клиентов
        c1.addToCart(p1);
        c2.addToCart(p2);
        c2.addToCart(p1);

        // оформляю заказы и производим оплату
        c1.checkout();
        c2.checkout();

        // регистрирую продажи
        admin.registerSale(c1.getCurrentOrder());
        admin.registerSale(c2.getCurrentOrder());

        // добавляю неплательщика в черный список
        admin.addCustomerToBlacklist(c1.getName());

        // проверяю, есть ли клиент в черном списке
        System.out.println(admin.isCustomerBlacklisted(c1.getName())); //
true
        System.out.println(admin.isCustomerBlacklisted(c2.getName())); //
false
    }
```

```
}
```

Результат

```
true
false
```