



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 № 17

Название: Классы, наследование и полиморфизм

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

М.О. Усманов

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2022

Цель работы

Получение навыков работы с классами языка программирования Java. Понимание механизмов наследования и полиморфизма объектно-ориентированного программирования.

Ход работы

Задание 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.

– 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 1 – Код основной программы подзадания 1 первого задания

```
import java.util.Scanner;

public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("Введите k:");

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

        Fraction[] fractions = new Fraction[k];

        for (int i = 0; i < k; i++) {
            System.out.println("Введите m для дроби " + (i + 1) + ":");
            int m = console.nextInt();
            System.out.println("Введите n для дроби " + (i + 1) + ":");
            int n = console.nextInt();

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

```

    }

    PrintArray(fractions);

    for (int i = 0; i < k - 1; i += 2) {
        fractions[i] = fractions[i].Add(fractions[i + 1]);
    }

    PrintArray(fractions);

    console.close();
}

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

```

Листинг 2 – Код класса Fraction подзадания 1 первого задания

```

public class Fraction {
    int m, n;

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

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

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

    public Fraction(Fraction a) {
        m = a.m;
        n = a.n;
    }

    // Сложение
    public Fraction Add(Fraction addee) {
        // Создаем новый объект класса Fraction
        Fraction sum = new Fraction();
        // Находим общий знаменатель
        int new_n = this.n * addee.n;
        // Присваиваем его новому объекту-сумме
        sum.n = new_n;
    }
}

```

```

        // Получаем сумму числителей
        int new_m = m * addee.n + addee.m * n;
        // Присваиваем её новому объекту-сумме
        sum.m = new_m;

        return sum;
    }

    // Умножение
    public Fraction Multiply(Fraction multiplee) {
        // Создаем новый объект класса Fraction
        // Сразу с умножением
        Fraction product = new Fraction(
            m * multiplee.m,
            n * multiplee.n
        );

        return product;
    }

    // Инверсия знака
    public Fraction Invert() {
        // Создаем новый объект класса Fraction
        Fraction inverted = new Fraction(-m, n);

        return inverted;
    }

    // Вычитание
    public Fraction Subtract(Fraction subtractee) {
        // Инвертируем вычитаемое, чтобы провести это как сложение
        Fraction inverted_subtractee = new Fraction(subtractee.Invert());

        // Складываем две дроби
        Fraction sub = this.Add(inverted_subtractee);

        return sub;
    }

    // Переворачивание дроби
    public Fraction Switch() {
        // Создаем новый объект класса Fraction
        // Переворачиваем дробь
        Fraction switched = new Fraction(n, m);

        return switched;
    }

    // Деление
    public Fraction Divide(Fraction dividee) {
        // Переворачиваем множитель для выполнения деления
        Fraction switched_multiplicator = this.Switch();

        // Умножаем на перевернутую дробь
        Fraction product = this.Multiply(switched_multiplicator);

        return product;
    }

```

```

    }

    // Вывод дроби
    public String toString() {
        String fraction = m + "/" + n;
        return fraction;
    }
}

```

Приведем результаты выполнения данного кода.

```

PS E:\Projects\Java\Repo\lab_3_var_1_7> & 'C:\Program Files\Java\jdk-9.0.4\bin\java.exe' -cp 'E:\Projects\Java\Repo\lab_3_var_1_7\bin' 'App'
Введите k:
2
Введите m для дроби 1:
3
Введите n для дроби 1:
4
Введите m для дроби 2:
5
Введите n для дроби 2:
8
[3/4, 5/8]
[44/32, 5/8]
PS E:\Projects\Java\Repo\lab_3_var_1_7>

```

Рисунок 1 – Результат выполнения подзадачи 1 варианта задания 1

Вторая подзадача.

Листинг 3 – Код основной программы подзадачи 2 задания 1

```

import java.util.Scanner;
import java.util.Vector;

public class App {
    public static void main(String[] args) throws Exception {
        System.out.println("Введите n:");

        Scanner console = new Scanner(System.in);
        int n = console.nextInt();

        Vector<Complex> complex_1 = new Vector<>();

        for (int i = 0; i < n; i++) {
            System.out.println("Введите re для числа " + (i + 1) + " в векторе 1:");
            int re = console.nextInt();
            System.out.println("Введите im для числа " + (i + 1) + " в векторе 1:");
            int im = console.nextInt();

```

```

        complex_1.add(new Complex(re, im));
    }

    Vector<Complex> complex_2 = new Vector<>();

    for (int i = 0; i < n; i++) {
        System.out.println("Введите re для числа " + (i + 1) + " в
векторе 2:");
        int re = console.nextInt();
        System.out.println("Введите im для числа " + (i + 1) + " в
векторе 2:");
        int im = console.nextInt();

        complex_2.add(new Complex(re, im));
    }

    PrintVector(complex_1);
    PrintVector(complex_2);

    Vector<Complex> result = SumVectors(complex_1, complex_2);

    PrintVector(result);

    console.close();
}

private static void PrintVector(Vector<Complex> vector) {
    System.out.print("[");
    for (int i = 0; i < vector.size(); i++) {
        System.out.print(vector.get(i).toString());
        if (i != vector.size() - 1) System.out.print(", ");
    }
    System.out.print("]\n");
}

private static Vector<Complex> SumVectors(Vector<Complex> a,
Vector<Complex> b) {
    Vector<Complex> result = new Vector<>();
    for (int i = 0; i < a.size(); i++) {
        result.add(a.get(i).Add(b.get(i)));
    }
    return result;
}
}

```

Листинг 4 – Код класса Complex второй подзадачи задания 1

```

public class Complex {
    int re, im;

    public Complex() {
        this.re = 0;
        this.im = 0;
    }

    public Complex(int re) {

```

```

        this.re = re;
        this.im = 0;
    }

    public Complex(int re, int im) {
        this.re = re;
        this.im = im;
    }

    public Complex(Complex complex) {
        this.re = complex.re;
        this.im = complex.im;
    }

    // Сложение
    public Complex Add(Complex addee) {
        // Создаем новый объект класса Complex
        Complex sum = new Complex(
            this.re + addee.re,
            this.im + addee.im
        );

        return sum;
    }

    // Вычитание
    public Complex Subtract(Complex subtractee) {
        Complex sub = new Complex(
            this.re - subtractee.re,
            this.im - subtractee.im
        );

        return sub;
    }

    // Умножение
    public Complex Multiply(Complex multee) {

        int k1 = multee.re * (this.re + this.im);
        int k2 = this.re * (multee.im - multee.re);
        int k3 = this.im * (multee.re + multee.im);

        int real_part = k1 - k3;
        int imaginary_part = k1 + k2;

        Complex mul = new Complex(real_part, imaginary_part);

        return mul;
    }

    // Деление
    public Complex Divide(Complex dividee) {

        int re_num = (this.re * dividee.re) + (this.im * dividee.im);
        int re_den = (dividee.re * dividee.re + dividee.im * dividee.im);
    }

```

```

        int im_num = (this.im * divdee.re) - (this.re * divdee.im);
        int im_den = (divdee.re * divdee.re + divdee.im * divdee.im);

        Complex div = new Complex(
            re_num / re_den,
            im_num / im_den
        );

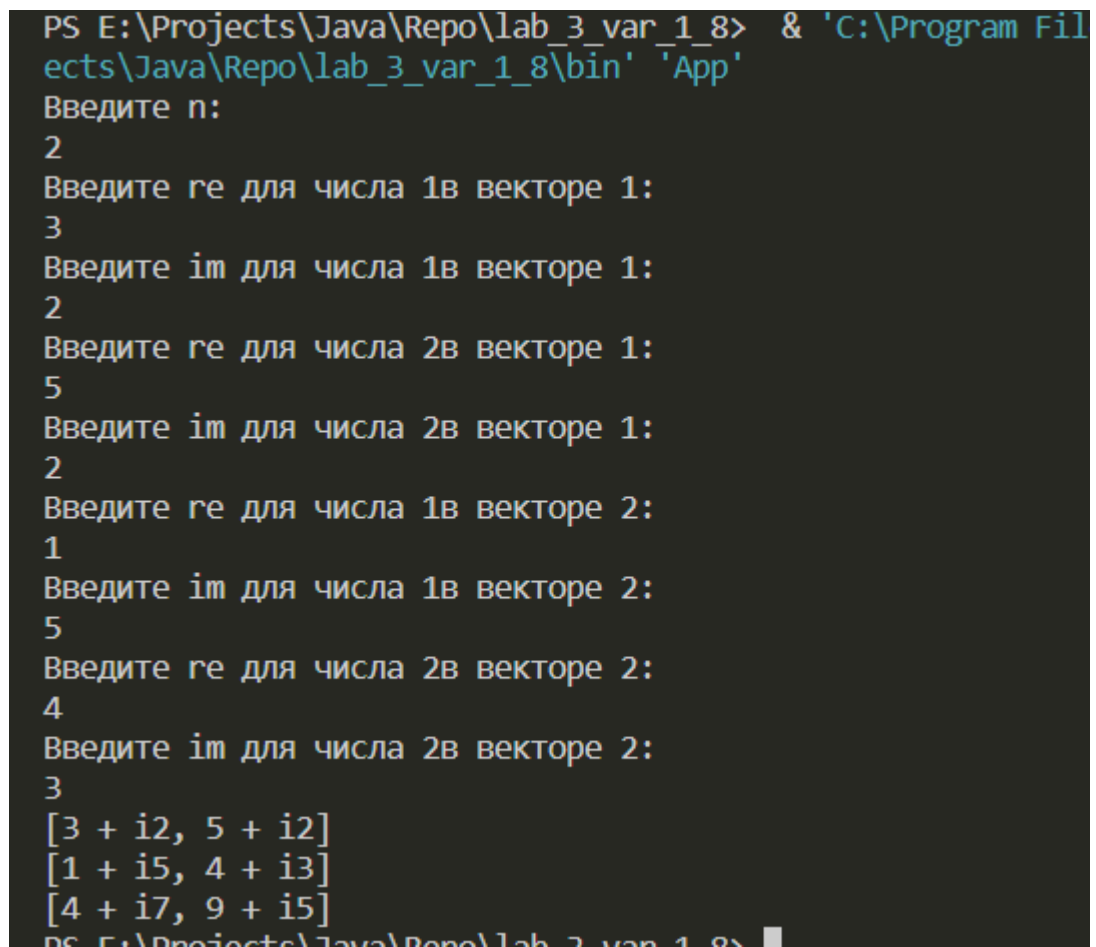
        return div;
    }

    // Присвоение
    public void SetComplex(int re, int im) {
        this.re = re;
        this.im = im;
    }

    // Вывод числа
    public String toString() {
        String complex = re + " + i" + im;
        return complex;
    }
}

```

Приведем результаты выполнения задачи.



```

PS E:\Projects\Java\Repo\lab_3_var_1_8> & 'C:\Program Fil
ects\Java\Repo\lab_3_var_1_8\bin' 'App'
Введите n:
2
Введите re для числа 1в векторе 1:
3
Введите im для числа 1в векторе 1:
2
Введите re для числа 2в векторе 1:
5
Введите im для числа 2в векторе 1:
2
Введите re для числа 1в векторе 2:
1
Введите im для числа 1в векторе 2:
5
Введите re для числа 2в векторе 2:
4
Введите im для числа 2в векторе 2:
3
[3 + i2, 5 + i2]
[1 + i5, 4 + i3]
[4 + i7, 9 + i5]
PS E:\Projects\Java\Repo\lab_3_var_1_8>

```

Рисунок 2 – Результат выполнения подзадачи 2 задания 1

Задание 2.

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; в) сведения об абонентах в алфавитном порядке.

– 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; в) список автомобилей заданного года выпуска, цена которых больше указанной.

Листинг 5 – Код выполнения основной программы задания 2

```
import java.util.Arrays;

public class App {
    public static void main(String[] args) throws Exception {
        Phone phones[] = new Phone[3];

        // Client 1
        phones[0] = new Phone();
        phones[0].setId(1);
        phones[0].setFirstName("Vadim");
        phones[0].setLastName("Santalov");
        phones[0].setPatronymic("Evgenyevich");
        phones[0].setAddress("Balashikha");
        phones[0].setCardNumber(4276000011110000L);
        phones[0].setDebit(300);
        phones[0].setCredit(100);
        phones[0].setIntercityCalls(10);
        phones[0].setIntracityCalls(135);

        // Client 2
        phones[1] = new Phone();
```

```

        phones[1].setId(2);
        phones[1].setFirstName("Octoslav");
        phones[1].setLastName("Eliseev");
        phones[1].setPatronymic("Alekseevich");
        phones[1].setAddress("Moscow, Ptichnoe");
        phones[1].setCardNumber(4276222211110000L);
        phones[1].setDebit(500);
        phones[1].setCredit(300);
        phones[1].setIntercityCalls(0);
        phones[1].setIntracityCalls(267);

        // Client 2
        phones[2] = new Phone();
        phones[2].setId(3);
        phones[2].setFirstName("Maxim");
        phones[2].setLastName("Usmanov");
        phones[2].setPatronymic("Oybekovich");
        phones[2].setAddress("Belorechensk");
        phones[2].setCardNumber(4276222211113333L);
        phones[2].setDebit(500);
        phones[2].setCredit(300);
        phones[2].setIntercityCalls(10);
        phones[2].setIntracityCalls(109);

        System.out.println("Превысившие лимиты по внутренней связи:");
        IntracityCallsExceeding(phones, 120);
        System.out.println("=====");
        System.out.println("Пользователи межгорода:");
        IntercityCallsUsers(phones);
        System.out.println("=====");
        System.out.println("Сведения в алфавитном порядке:");
        AllUsersAlphabetical(phones);
    }

    private static void IntracityCallsExceeding(Phone[] phones, int
limit) {
        Arrays.stream(phones)
            .filter(c -> c.getIntracityCalls() > limit)
            .forEach(c -> System.out.println(c.toString()));
    }

    private static void IntercityCallsUsers(Phone[] phones) {
        Arrays.stream(phones)
            .filter(c -> c.getIntercityCalls() > 0)
            .forEach(c -> System.out.println(c.toString()));
    }

    private static void AllUsersAlphabetical(Phone[] phones) {
        Arrays.sort(phones, (a,b) -> a.last_name.compareTo(b.last_name));
        Arrays.stream(phones)
            .forEach(c -> System.out.println(c.toString()));
    }
}

```

Листинг 6 – Код класса Phone

```
public class Phone {
    int id;
    String first_name, last_name, patronymic;
    String address;
    long card_number;
    double debit, credit;
    int intercity_calls, intracity_calls; // Междугородние звонки и
    внутригородские соотв.

    public int getId() {
        return id;
    }

    public String getLastName() {
        return last_name;
    }

    public String getFirstName() {
        return first_name;
    }

    public String getPatronymic() {
        return patronymic;
    }

    public String getAddress() {
        return address;
    }

    public long getCardNumber() {
        return card_number;
    }

    public double getDebit() {
        return debit;
    }

    public double getCredit() {
        return credit;
    }

    public int getIntercityCalls() {
        return intercity_calls;
    }

    public int getIntracityCalls() {
        return intracity_calls;
    }

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

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

```

    }

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

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

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

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

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

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

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

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

    @Override
    public String toString() {
        String result = "{ " +
            "ID: " + this.id + "\n" +
            "Last Name: " + this.last_name + "\n" +
            "First Name: " + this.first_name + "\n" +
            "Patronymic: " + this.patronymic + "\n" +
            "Address: " + this.address + "\n" +
            "Card Number: " + this.card_number + "\n" +
            "Debit: " + this.debit + "\n" +
            "Credit: " + this.credit + "\n" +
            "Intercity Calls: " + this.intercity_calls + " min. \n" +
            "Intracity Calls: " + this.intracity_calls + " min. " +
            "}";
        return result;
    }
}

```

Результаты выполнения задания приведем на рисунке далее.

```

PS E:\Projects\Java\Repo\lab_3_var_2_7> & 'C:\Program Files\Amazon Corretto\jdk17.0.2_8\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'E:\Projects\Java\Repo\lab_3_var_2_7\bin' 'App'
Превысившие лимиты по внутренней связи:
{ ID: 1
  Last Name: Santalov
  First Name: Vadim
  Patronymic: Evgenyevich
  Address: Balashikha
  Card Number: 4276000011110000
  Debit: 300.0
  Credit: 100.0
  Intercity Calls: 10 min.
  Intracity Calls: 135 min. }
{ ID: 2
  Last Name: Eliseev
  First Name: Octoslav
  Patronymic: Alekseevich
  Address: Moscow, Ptichnoe
  Card Number: 4276222211110000
  Debit: 500.0
  Credit: 300.0
  Intercity Calls: 0 min.
  Intracity Calls: 267 min. }
=====
Пользователи межгорода:
{ ID: 1
  Last Name: Santalov
  First Name: Vadim
  Patronymic: Evgenyevich
  Address: Balashikha
  Card Number: 4276000011110000
  Debit: 300.0
  Credit: 100.0
  Intercity Calls: 10 min.
  Intracity Calls: 135 min. }
{ ID: 3
  Last Name: Usmanov
  First Name: Maxim
  Patronymic: Oybekovich
  Address: Belorechensk
  Card Number: 4276222211113333

```

Рисунок 3 – Выполненный код п. 1 задания 2

Второй подпункт.

Листинг 7 – Код выполнения второго пункта задания 2

```

import java.util.Arrays;

public class App {
    public static void main(String[] args) throws Exception {
        Car cars[] = new Car[4];

        // Car 1
        cars[0] = new Car();
        cars[0].setId(1);
        cars[0].setBrand("Volkswagen");
        cars[0].setModel("Passat");
        cars[0].setYear(2008);
        cars[0].setColor("Metallic");
        cars[0].setPrice(450000);
        cars[0].setNumber("H312ae");

        // Car 2
        cars[1] = new Car();
        cars[1].setId(2);
        cars[1].setBrand("Volkswagen");
        cars[1].setModel("Polo");
        cars[1].setYear(2008);
        cars[1].setColor("Black");
        cars[1].setPrice(350000);
        cars[1].setNumber("H1546r");

        // Car 3
        cars[2] = new Car();
        cars[2].setId(3);
        cars[2].setBrand("Mitsubishi");
    }
}

```

```

        cars[2].setModel("Lancer");
        cars[2].setYear(2010);
        cars[2].setColor("Red");
        cars[2].setPrice(1350000);
        cars[2].setNumber("a102ne");

        // Car 4
        cars[3] = new Car();
        cars[3].setId(4);
        cars[3].setBrand("Hyundai");
        cars[3].setModel("Accent");
        cars[3].setYear(2010);
        cars[3].setColor("Red");
        cars[3].setPrice(400000);
        cars[3].setNumber("н252см");

        System.out.println("Автомобили по марке:");
        PickBrand(cars, "Volkswagen");
        System.out.println("=====");
        System.out.println("Автомобили по модели и эксплуатации:");
        PickModel(cars, "Accent", 2022, 10);
        System.out.println("=====");
        System.out.println("По году выпуска с превышением заданной
цены:");
        PickYearPrice(cars, 2010, 1000000);
    }

    private static void PickBrand(Car[] cars, String brand) {
        Arrays.stream(cars)
            .filter(c -> c.getBrand() == brand)
            .forEach(c -> System.out.println(c.toString()));
    }

    private static void PickModel(Car[] cars, String model, int
currentYear, int n) {
        Arrays.stream(cars)
            .filter(c -> c.getModel() == model)
            .filter(c -> currentYear - c.getYear() > n)
            .forEach(c -> System.out.println(c.toString()));
    }

    private static void PickYearPrice(Car[] cars, int year, int
priceLimit) {
        Arrays.stream(cars)
            .filter(c -> c.getYear() == year)
            .filter(c -> c.getPrice() > priceLimit)
            .forEach(c -> System.out.println(c.toString()));
    }
}

```

Листинг 8 – Код класса Машина

```

public class Car {
    int id;
    String brand;
    String model;
    int year;
}

```

```
String color;
int price;
String number;

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

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

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

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

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

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

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

public int getId() {
    return id;
}

public String getBrand() {
    return brand;
}

public String getModel() {
    return model;
}

public String getColor() {
    return color;
}

public String getNumber() {
    return number;
}

public int getPrice() {
    return price;
}

public int getYear() {
```

```

        return year;
    }

    @Override
    public String toString() {
        String result = "{ " +
            "ID: " + this.id + "\n" +
            "Brand: " + this.brand + "\n" +
            "Model: " + this.model + "\n" +
            "Year: " + this.year + "\n" +
            "Color: " + this.color + "\n" +
            "Price: " + this.price + "\n" +
            "Number: " + this.number + "\n" +
            "}";
        return result;
    }
}

```

```

s' '-cp' 'E:\Projects\Java\Repo\lab_3_var_2_8\bin' 'App'
Автомобили по марке:
{ ID: 1
Brand: Volkswagen
Model: Passat
Year: 2008
Color: Metallic
Price: 450000
Number: н312ae
}
{ ID: 2
Brand: Volkswagen
Model: Polo
Year: 2008
Color: Black
Price: 350000
Number: н1546г
}
=====
Автомобили по модели и эксплуатации:
{ ID: 4
Brand: Hyundai
Model: Accent
Year: 2010
}

```

Рисунок 4 – Результаты выполнения задания для класса Car

Задание 3.

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

Листинг 9 – Код класса «Бутон»

```
import java.util.Objects;

public class Bud {
    public String color;

    public Petal[] petals;

    public Bud(String color) {
        petals = new Petal[10];

        for (int i = 0; i < 10; i++) {
            petals[i] = new Petal();
        }

        this.color = color;
    }

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

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

    @Override
    public String toString() {
        return "Bud{" +
            "color='" + color + '\'' +
            '}';
    }
}
```

Листинг 10 – Код класса «Лепесток»

```
import java.util.Objects;

public class Petal {
    public String shape;

    public Petal() {
        this.shape = "Basic";
    }
}
```

```

    }

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

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

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

    @Override
    public String toString() {
        return "Petal{" +
            "shape='" + shape + '\'' +
            '}';
    }
}

```

Листинг 11 – Код класса «Роза»

```

import java.util.Objects;

public class Rose {
    Bud bud;
    FlowerState state = FlowerState.GROWING;

    public Rose(String color) {
        bud = new Bud(color);
    }

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

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

    public void PrintBudColor() {
        System.out.println(bud.color);
    }

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

```

```

    }

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

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

```

Листинг 12 – Код enum FlowerState

```

public enum FlowerState {
    GROWING, BLOOM, WITHER
}

```

Вторая подзадача.

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

Листинг 13 – Код класса «Лист»

```

import java.util.Objects;

public class Leaf {
    String color = "Green";
    LeafState state = LeafState.CLEAN;

    public void YellowLeaf() {
        color = "Yellow";
    }

    public void SnowLeaf() {
        state = LeafState.SNOWY;
    }

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

    @Override
    public int hashCode() {
        return Objects.hash(color + state.toString());
    }

    @Override

```

```

    public String toString() {
        return "Leaf{" +
            "color='" + color + '\'' +
            "state='" + state + '\'' +
            '}';
    }
}

```

Листинг 14 – Код класса «Дерево»

```

import java.util.Arrays;
import java.util.Objects;

public class Tree {
    Leaf leafs[];
    TreeState state = TreeState.GROWING;

    public Tree() {
        leafs = new Leaf[25];

        for (int i = 0; i < 25; i++) {
            leafs[i] = new Leaf();
        }
    }

    public void YellowLeafs() {
        Arrays.stream(leafs)
            .forEach(c -> c.YellowLeaf());
    }

    public void Blossom() {
        state = TreeState.BLOSSOMING;
    }

    public void DropLeafs() {
        Arrays.fill(leafs, null);
    }

    public void GetSnowy() {
        Arrays.stream(leafs)
            .forEach(c -> c.SnowLeaf());
    }

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

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

    @Override

```

```

    public String toString() {
        return "Tree{" +
            "state='" + state + '\'' +
            '}';
    }
}

```

Листинг 15 – Классы enum состояния дерева и листа

```

public enum TreeState {
    GROWING, BLOSSOMING
}

public enum LeafState {
    CLEAN, SNOWY
}

```

Задание 4.

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 16 – Код класса «Bill»

```

import java.util.HashMap;

public class Bill {
    HashMap<Integer, Service> activeServices = new HashMap<>();
    int sum;

    public void AddService(int id, Service service) {
        activeServices.put(id, service);
        sum += service.price;
    }

    public void RemoveService(int id) {
        sum -= activeServices.get(id).price;
        activeServices.remove(id);
    }

    public int GetSum() {
        return sum;
    }
}

```

Листинг 17 – Код класса «Client»

```

public class Client {
    boolean status = true;
    int balance = 0;
    int number;
}

```

```

public Bill bill = new Bill();

public Client(int number) {
    this.number = number;
}

public void SwitchStatus() {
    status = !status;
}

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

public void Pay(int amount) {
    balance += amount;
}

public int GetBalance() {
    return balance;
}

public boolean CheckPayment() {
    boolean result = true;
    if (balance >= bill.sum) {
        balance -= bill.sum;
    } else {
        result = false;
    }

    status = result;
    return result;
}
}

```

Листинг 18 – Код класса «Service»

```

public class Service {
    public String title;
    public int price;

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

```

Листинг 19 – Код выполнения основной программы

```

import java.util.HashMap;

public class App {
    public static void main(String[] args) throws Exception {
        // Своеобразная "БД" услуг
        HashMap<Integer, Service> services = new HashMap<>();
    }
}

```

```

        services.put(
            0, new Service("Какая-то платная рекламная подписка нио чем",
300)
        );

        services.put(
            1, new Service("Безлимитный интернет", 1500)
        );

        services.put(
            2, new Service("SMS-бомбинг", 100)
        );

        // Несколько тестовых пользователей
        Client client_1 = new Client(111);
        Client client_2 = new Client(222);

        // Клиенты при подключении выбирают услуги
        client_1.bill.AddService(0, services.get(0));
        client_1.bill.AddService(2, services.get(2));

        client_2.bill.AddService(1, services.get(1));
        client_2.bill.AddService(2, services.get(2));

        // Клиенты пополняют счет
        client_1.Pay(400);
        client_2.Pay(1000);

        // Проходит месяц
        System.out.println(client_1.CheckPayment());
        System.out.println(client_2.CheckPayment());

        // Команды администратора
        RemoveClientService(client_1, 2);
        ChangeNumber(client_2, 314);
    }

    public static void ChangeNumber(Client client, int number) {
        client.SetNumber(number);
    }

    public static void RemoveClientService(Client client, int id) {
        client.bill.RemoveService(id);
    }
}

```

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 20 – Класс «Car»

```
public class Car {  
    String brand;  
  
    public Car(String brand) {  
        this.brand = brand;  
    }  
}
```

Листинг 21 – enum «CarState»

```
public enum CarState {  
    GOOD, AGED, DAMAGED  
}
```

Листинг 22 – Класс «Driver»

```
import java.util.Objects;  
  
public class Driver {  
    String name;  
    Trip trip;  
    DriverStatus status = DriverStatus.ON_HOLD;  
  
    public Driver(String name) {  
        this.name = name;  
    }  
  
    public void SetTrip(Trip trip) {  
        this.trip = trip;  
        status = DriverStatus.ON_TRIP;  
    }  
  
    public void TakeOffTrip() {  
        this.trip = null;  
        status = DriverStatus.ON_HOLD;  
    }  
  
    public void RequestRepair() {  
        status = DriverStatus.ON_REPAIR;  
    }  
  
    public void DoRepair() {  
        if (Objects.isNull(trip))  
            status = DriverStatus.ON_HOLD;  
        else  
            status = DriverStatus.ON_TRIP;  
    }  
}
```

Листинг 23 – enum «DriverStatus»

```
public enum DriverStatus {  
    ON_HOLD, ON_TRIP, ON_REPAIR  
}
```


Листинг 24 – Класс «Trip»

```
public class Trip {
    String destination;
    Car car;
    boolean finished = false;
    CarState carState;

    public Trip(String destination) {
        this.destination = destination;
    }

    public void SetCar(Car car) {
        this.car = car;
    }

    public void MarkTripFinish(CarState state) {
        carState = state;
        finished = true;
    }
}
```

В результате выполнения задания были построены структуры двух программных систем. Классы в данных системах взаимодействуют между собой в том числе в рамках взаимоотношений агрегации.

Местоположение репозитория с файлами проекта

Файлы проекта расположены в репозитории веб-платформы для совместной разработки Github. Местоположение в репозитории:

https://github.com/s314/big-data-studies/tree/main/lab_3_var_1_7

https://github.com/s314/big-data-studies/tree/main/lab_3_var_1_8

https://github.com/s314/big-data-studies/tree/main/lab_3_var_2_7

https://github.com/s314/big-data-studies/tree/main/lab_3_var_2_8

https://github.com/s314/big-data-studies/tree/main/lab_3_var_3_6

https://github.com/s314/big-data-studies/tree/main/lab_3_var_3_7

https://github.com/s314/big-data-studies/tree/main/lab_3_var_4_7

https://github.com/s314/big-data-studies/tree/main/lab_3_var_4_8

Вывод

По итогам выполнения лабораторной работы были получены навыки работы с классами ЯП Java. Были также изучены механизмы полиморфизма и наследования.