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 **09.04.01/12 Интеллектуальный анализ больших данных в системах поддержки принятия решений.**

О Т Ч Е Т

по лабораторной работе № 3

Вариант 6

Название: Классы, наследование, полиморфизм

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-23М		Г.Л. Кушнир
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023

Цель: Освоить принципы ООП на языке программирования Java.

Вариант 1 задание 6: Определить класс Цепная дробь.

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

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

Код класса ContinuedFraction:

```
public class ContinuedFraction {

    private double result;

    public ContinuedFraction(){
        setResult(0);
    }

    public ContinuedFraction(double result){
        setResult(result);
    }

    public ContinuedFraction(int x, int n, int[] a){
        double result = 1;
        for (int i = n - 1; i >= 0; i--){
            result = a[i] + x / result;
        }
        setResult(result);
    }

    public void setResult(double result) {
        this.result = result;
    }

    public double getResult() {
        return result;
    }

    public static ContinuedFraction add(ContinuedFraction fractionFirst,
ContinuedFraction fractionSecond){
        return new ContinuedFraction(fractionFirst.getResult() +
fractionSecond.getResult());
    }

    public static ContinuedFraction subtract(ContinuedFraction fractionFirst,
ContinuedFraction fractionSecond){
        return new ContinuedFraction(fractionFirst.getResult() -
fractionSecond.getResult());
    }

    public static ContinuedFraction multiply(ContinuedFraction fractionFirst,
ContinuedFraction fractionSecond){
        return new ContinuedFraction(fractionFirst.getResult() *
fractionSecond.getResult());
    }
```

```
    public static ContinuedFraction divide(ContinuedFraction fractionFirst,
ContinuedFraction fractionSecond){
        return new ContinuedFraction(fractionFirst.getResult() /
fractionSecond.getResult());
    }

}
```

Вариант 1 задание 7: Определить класс Дробь в виде пары (m,n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения и деления дробей. Объявить массив из k дробей, ввести/вывести значения для массива дробей. Создать массив объектов и передать его в метод, который изменяет каждый элемент массива с четным индексом путем добавления следующего за ним элемента массива.

Код класса Fraction:

```
public class Fraction {

    private int numerator;
    private int divider;

    public Fraction(){
        setNumerator(1);
        setDivider(1);
    }

    public Fraction(int numerator, int divider){
        setNumerator(numerator);
        setDivider(divider);
    }

    public void setNumerator(int numerator) {
        this.numerator = numerator;
    }

    public int getNumerator() {
        return numerator;
    }

    public void setDivider(int divider) {
        this.divider = divider;
    }

    public int getDivider() {
        return divider;
    }

    public static Fraction add(Fraction fractionFirst, Fraction
fractionSecond){
        int numerator = fractionFirst.getNumerator() *
fractionSecond.getDivider() + fractionSecond.getNumerator() *
fractionFirst.getDivider();
        int divider = fractionFirst.getDivider() *
fractionSecond.getDivider();
        return correction(numerator, divider);
    }
```

```
    public static Fraction subtract(Fraction fractionFirst, Fraction
fractionSecond){
        int numerator = fractionFirst.getNumerator() *
fractionSecond.getDivider() - fractionSecond.getNumerator() *
fractionFirst.getDivider();
        int divider = fractionFirst.getDivider() *
fractionSecond.getDivider();
        return correction(numerator, divider);
    }

    public static Fraction multiply(Fraction fractionFirst, Fraction
fractionSecond){
        int numerator = fractionFirst.getNumerator() *
fractionSecond.getNumerator();
        int divider = fractionFirst.getDivider() *
fractionSecond.getDivider();
        return correction(numerator, divider);
    }

    public static Fraction divide(Fraction fractionFirst, Fraction
fractionSecond){
        int numerator = fractionFirst.getNumerator() *
fractionSecond.getDivider();
        int divider = fractionFirst.getDivider() *
fractionSecond.getNumerator();
        return correction(numerator, divider);
    }

    public static Fraction correction(int numerator, int divider){
        int greatCommonDivision = gcd(numerator, divider);
        numerator /= greatCommonDivision;
        divider /= greatCommonDivision;
        return divider < 0 ? new Fraction(-numerator, -divider) : new
Fraction(numerator, divider);
    }

    private static int gcd(int a,int b){
        return b == 0 ? a : gcd(b,a % b);
    }

    public static void method(Fraction[] fractionArray){
        for (int i = 1; i < fractionArray.length; i++){
            fractionArray[i-1] = Fraction.add(fractionArray[i-1],
fractionArray[i]);
        }
    }

}
```

Вариант 2 задание 6: House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: a) список квартир, имеющих заданное число комнат; b) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; c) список квартир, имеющих площадь, превосходящую заданную.

Код класса House без учета всех геттеров и сеттеров:

```
import java.util.Calendar;
import java.util.GregorianCalendar;

public class House {

    private int id;
    private int houseNumber;
    private double square;
    private int floor;
    private int roomsCount;
    private String address;
    private String type;
    private GregorianCalendar data;

    public House(){}

    public House(int id, int houseNumber, double square, int floor, int
roomsCount){
        setId(id);
        setHouseNumber(houseNumber);
        setSquare(square);
        setFloor(floor);
        setRoomsCount(roomsCount);
    }

    public House(int id, int houseNumber, double square, int floor, int
roomsCount, String address,
                 String type, GregorianCalendar data){
        setId(id);
        setHouseNumber(houseNumber);
        setSquare(square);
        setFloor(floor);
        setRoomsCount(roomsCount);
        setAddress(address);
        setType(type);
        setData(data);
    }

    @Override
    public String toString(){
        return String.format("""
                House = { \n\tid: %d \n\thouseNumber: %d \n\tsquare: %f
\n\tfloor: %d \n\troomsCount: %d \n\taddress: %s
                type: %s \n\t lifetime: %d:%d:%d""", id, houseNumber, square,
floor, roomsCount, address, type,
                data.get(Calendar.DAY_OF_MONTH), data.get(Calendar.MONTH),
data.get(Calendar.YEAR));
    }
}
```

Вариант 2 задание 7: Phone: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Дебет, Кредит, Время городских и междугородных разговоров. Создать массив объектов. Вывести: a) сведения об абонентах, у которых время внутригородских разговоров превышает заданное; b) сведения об абонентах, которые пользовались междугородной связью; c) сведения об абонентах в алфавитном порядке.

Код класса Phone без учета всех сеттеров и гетеров:

```
public class Phone {

    private int id;
    private String surname;
    private String name;
    private String patronymic;
    private String address;
    private String creditCardNumber;
    private double debit;
    private double credit;
    private long cityCallTimeInSeconds;
    private long intercityCallTimeInSeconds;

    public Phone(){}

    public Phone(int id, String surname, String name, String patronymic,
                 long cityCallTimeInSeconds, long
intercityCallTimeInSeconds){
        setId(id);
        setSurname(surname);
        setName(name);
        setPatronymic(patronymic);
        setCityCallTimeInSeconds(cityCallTimeInSeconds);
        setIntercityCallTimeInSeconds(intercityCallTimeInSeconds);
    }

    public Phone(int id, String surname, String name, String patronymic,
String address, String creditCardNumber,
                 double debit, double credit, long cityCallTimeInSeconds,
long intercityCallTimeInSeconds){
        setId(id);
        setSurname(surname);
        setName(name);
        setPatronymic(patronymic);
        setAddress(address);
        setCreditCardNumber(creditCardNumber);
        setDebit(debit);
        setCredit(credit);
        setCityCallTimeInSeconds(cityCallTimeInSeconds);
        setIntercityCallTimeInSeconds(intercityCallTimeInSeconds);
    }

    public static void sort(Phone[] array, int leftBorder, int rightBorder){
        int leftMarker = leftBorder;
        int rightMarker = rightBorder;
        Phone pivot = array[(leftMarker + rightMarker) / 2];
        do {
            while (compare(array[leftMarker], pivot)) {
                leftMarker++;
            }
            while (compare(pivot, array[rightMarker])) {
                rightMarker--;
            }
            if (leftMarker <= rightMarker) {
                if (leftMarker < rightMarker) {
                    Phone temp = array[leftMarker];
                    array[leftMarker] = array[rightMarker];
                    array[rightMarker] = temp;
                }
                leftMarker++;
                rightMarker--;
            }
        } while (leftMarker <= rightMarker);
        if (leftMarker < rightBorder) {
```

```
            sort(array, leftMarker, rightBorder);
        }
        if (leftBorder < rightMarker) {
            sort(array, leftBorder, rightMarker);
        }
    }

    public static boolean compare(Phone firstPhone, Phone secondPhone){
        int surnameCompare =
secondPhone.getSurname().compareTo(firstPhone.getSurname());
        int nameCompare =
secondPhone.getName().compareTo(firstPhone.getName());
        int patronymicCompare =
secondPhone.getPatronymic().compareTo(firstPhone.getPatronymic());
        if (surnameCompare > 0){
            return true;
        } else if ((surnameCompare == 0) && (nameCompare > 0)) {
            return true;
        } else if ((surnameCompare == 0) && (nameCompare == 0) &&
(patronymicCompare > 0)) {
            return true;
        } else {
            return false;
        }
    }

    @Override
    public String toString(){
        return String.format("""
                Phone = { \n\tid: %d \n\tsurname: %s \n\tname: %s
\n\tpatronymic: %s \n\taddress: %s
                \tcreditCardNumber: %s \n\tdebit: %f \n\tcredit: %f
\n\tcityCallTimeInSeconds: %d
                \tintercityCallTimeInSeconds: %d \n}""", id, surname, name,
patronymic, address, creditCardNumber,
                debit, credit, cityCallTimeInSeconds,
intercityCallTimeInSeconds);
    }

}
```

Вариант 3 задание 6: Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завять, вывести на консоль цвет бутона.

Код класса Bud:

```
public abstract class Bud {

    protected int countPetals;
    protected Petal[] petals;

    public Bud(){
    }

    public int getCountPetals() {
        return countPetals;
    }

    public void setPetals(Petal[] petals) {
        this.petals = petals;
        this.countPetals = petals.length;
    }
```

```
    public Petal[] getPetals() {
        return petals;
    }

}
```

Код класса Petal:

```
import java.util.Objects;

public class Petal {

    private String color;

    public Petal(String color){
        setColor(color);
    }

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

    public String getColor() {
        return color;
    }

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

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

    @Override
    public String toString(){
        return String.format("Petal = { color: %s }", color);
    }

}
```

Код класса Rose:

```
import java.util.Objects;
import java.util.Random;

public class Rose extends Bud{

    public Rose(){
        bloom();
    }

    public void bloom(){
        Random random = new Random();
        this.countPetals = random.nextInt(10) + 2;
        this.petals = new Petal[this.countPetals];
        String color = switch (random.nextInt(9)) {
            case 0 -> "Red";
            case 1 -> "Black";
            case 2 -> "White";
            case 3 -> "Violet";
            case 4 -> "Pink";
```

```
            case 5 -> "Orange";
            case 6 -> "Cream";
            case 7 -> "Lavender";
            default -> "Blue";
        };
        for (int i = 0; i < this.countPetals; i++) {
            this.petals[i] = new Petal(color);
        }
    }

    public void wither(){
        setPetals(new Petal[0]);
    }

    public String getColor(){
        return petals[0].getColor();
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) { return true; }
        if (!(o instanceof Rose rose)) return false;
        Petal[] firstPetals = getPetals();
        Petal[] secondPetals = rose.getPetals();
        if (firstPetals.length != secondPetals.length) { return false; }
        boolean check = true;
        for (int i = 0; check && i < countPetals; i++) {
            if (!firstPetals[i].equals(secondPetals[i])) { check = false; }
        }
        return check;
    }

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

    @Override
    public String toString() {
        StringBuilder result = new StringBuilder("Rose = {\n");
        result.append(String.format("\tcountPetal: %d\n\tpetals = {\n",
countPetals));
        for (Petal petals : getPetals()) {
            result.append("\t\t").append(petals.toString()).append("\n");
        }
        result.append("\t}\n}");
        return result.toString();
    }
}
```

Вариант 3 задание 7: Создать объект класса Дерево, используя классы Лист. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.

Код класса Leaf:

```
import java.util.Objects;

public class Leaf {

    private String state;
```

```
    public Leaf(){
        setState("Green");
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getState() {
        return state;
    }

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

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

    @Override
    public String toString(){
        return String.format("Leaf = { state: %s }", state);
    }

}
```

Код класса Tree:

```
import java.util.Objects;
import java.util.Random;

public class Tree {

    protected int countLeaves;
    protected Leaf[] leaves;

    public Tree(){
        bloom();
    }

    public void bloom(){
        Random random = new Random();
        this.countLeaves = random.nextInt(20) + 10;
        this.leaves = new Leaf[this.countLeaves];
        for (int i = 0; i < this.countLeaves; i++) {
            this.leaves[i] = new Leaf();
        }
    }

    public void wither(){
        setLeaves(new Leaf[0]);
    }

    public void turnYellow(){
        Random random = new Random();
        for (int i = 0; i < this.countLeaves; i++){
            boolean choice = random.nextBoolean();
            if (choice) { leaves[i].setState("Yellow"); }
        }
    }
```

```
    public void coverWithFrost(){
        Random random = new Random();
        for (int i = 0; i < this.countLeaves; i++){
            boolean choice = random.nextBoolean();
            if (choice) { leaves[i].setState("Covered with frost"); }
        }
    }

    public int getCountLeaves() {
        return countLeaves;
    }

    public void setLeaves(Leaf[] leaves) {
        this.leaves = leaves;
        this.countLeaves = leaves.length;
    }

    public Leaf[] getLeaves() {
        return leaves;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) { return true; }
        if (!(o instanceof Tree tree)) return false;
        Leaf[] firstLeaves = getLeaves();
        Leaf[] secondLeaves = tree.getLeaves();
        if (firstLeaves.length != secondLeaves.length) { return false; }
        boolean check = true;
        for (int i = 0; check && i < countLeaves; i++) {
            if (!firstLeaves[i].equals(secondLeaves[i])) { check = false; }
        }
        return check;
    }

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

    @Override
    public String toString() {
        StringBuilder result = new StringBuilder("Tree = {\n");
        result.append(String.format("\tcountLeaves: %d\n\tleaves = {\n",
countLeaves));
        for (Leaf leaves : getLeaves()) {
            result.append("\t\t").append(leaves.toString()).append("\n");
        }
        result.append("\t}\n}");
        return result.toString();
    }

}
```

Вариант 4 задание 6: Система Конструкторское бюро. Заказчик представляет Техническое Задание (ТЗ) на проектирование многоэтажного Дома. Конструктор регистрирует ТЗ, определяет стоимость проектирования и строительства, выставляет Заказчику Счет за проектирование и создает Бригаду Конструкторов для выполнения Проекта.

Код класса DesignDepartment:

```
import java.util.Objects;
import java.util.Scanner;

public class DesignDepartment {

    private int countProjects;
    private Project[] projectList;

    public DesignDepartment(){
        setProjectList(new Project[0]);
    }

    public DesignDepartment(Project[] projectList){
        setProjectList(projectList);
    }

    public void addProject(){
        Project[] projects = new Project[countProjects + 1];
        if (countProjects >= 0) System.arraycopy(projectList, 0, projects, 0,
countProjects);
        Scanner in = new Scanner(System.in);
        System.out.print("Enter project name: ");
        String name = in.nextLine();
        System.out.print("Enter project description: ");
        String description = in.nextLine();
        System.out.print("Enter amount of work in project: ");
        int workCount = in.nextInt();
        Work[] workList = new Work[workCount];
        for (int i = 0; i < workCount; i++){
            System.out.print(String.format("Enter name of %d work: ", i+1));
            String workName = in.nextLine();
            System.out.print(String.format("Enter name of %d work: ", i+1));
            double cost = in.nextDouble();
            System.out.print(String.format("Is %d work building type work?
(yes/no) ", i+1));
            String choice = in.nextLine();
            if (Objects.equals(choice, "yes")){
                workList[i] = new Work(workName, cost, true);
            } else{
                workList[i] = new Work(workName, cost, false);
            }
        }
        projects[countProjects] = new Project(countProjects+1, name, new
TechnicalTask(description, workList));
        countProjects += 1;
        projectList = projects;
    }

    public void showInvoice(){
        int pos = searchCustomer();
        if (pos != -1){
            System.out.println(String.format("Totally cost of this project is
%f",
                    projectList[pos].getProjectCost() +
projectList[pos].getBuildingCost()));
        }
    }

    public void setProjectTeam(){
        int pos = searchCustomer();
        if (pos != -1){
            projectList[pos].setProjectTeam();
```

```
        }
    }

    public int searchCustomer(){
        Scanner in = new Scanner(System.in);
        System.out.print("Enter project's name for searching: ");
        String name = in.nextLine();
        for (int i = 0; i < countProjects; i++){
            if (name.equals(projectList[i].getName())){
                return i;
            }
        }
        System.out.println("No such project");
        return -1;
    }

    public void setProjectList(Project[] projectList) {
        this.projectList = projectList;
        this.countProjects = projectList.length;
    }

    public Project[] getProjectList() {
        return projectList;
    }

    public int getCountProjects() {
        return countProjects;
    }
}
```

Вариант 4 задание 7: Система Телефонная станция. Абонент оплачивает Счет за разговоры и Услуги, может попросить Администратора сменить номер и отказаться от услуг. Администратор изменяет номер, Услуги и временно отключает Абонента за неуплату.

Код класса TelephoneStation:

```
import java.util.Scanner;

public class TelephoneStation {

    int customerCount;
    Customer[] customerList;

    public TelephoneStation(){
        setCustomerList(new Customer[0]);
    }

    public void addCustomer(){
        Customer[] temp = new Customer[customerCount + 1];
        if (customerCount >= 0) System.arraycopy(customerList, 0, temp, 0,
customerCount);
        Scanner in = new Scanner(System.in);
        System.out.print("Enter customer's surname: ");
        String surname = in.nextLine();
        System.out.print("Enter customer's name: ");
        String name = in.nextLine();
        System.out.print("Enter customer's phone number: ");
        String phoneNumber = in.nextLine();
        System.out.print("Enter customer's balance: ");
        double balance = in.nextDouble();
```

```
        temp[customerCount] = new Customer(customerCount + 1, surname, name,
phoneNumber, balance);
        customerCount += 1;
        customerList = temp;
    }

    public void changeNumber(){
        int pos = searchCustomer();
        if (pos != -1){
            Scanner in = new Scanner(System.in);
            System.out.print("Enter new customer's phone: ");
            String phoneNumber = in.nextLine();
            customerList[pos].setPhoneNumber(phoneNumber);
        }
    }

    public void changeServices(){
        int pos = searchCustomer();
        if (pos != -1){
            customerList[pos].changeServices();
        }
    }

    public void repayPayment(){
        int pos = searchCustomer();
        if (pos != -1){
            System.out.print("Enter customer's payment: ");
            Scanner in = new Scanner(System.in);
            double payment = in.nextDouble();
            customerList[pos].payToBalance(payment);
        }
    }

    public void payForServices(){
        System.out.println("Payment for service starts");
        for (int i = 0; i < customerCount; i++){
            customerList[i].payForPayment();
        }
        System.out.println("Payment for service ended");
    }

    public void checkBalance(){
        int pos = searchCustomer();
        if (pos != -1){
            System.out.println(String.format("Customer's balance: %f",
customerList[pos].getBalance()));
        }
    }

    public int searchCustomer(){
        Scanner in = new Scanner(System.in);
        System.out.print("Enter customer's surname for searching: ");
        String surname = in.nextLine();
        System.out.print("Enter customer's name for searching: ");
        String name = in.nextLine();
        for (int i = 0; i < customerCount; i++){
            if (surname.equals(customerList[i].getSurname()) &&
name.equals(customerList[i].getName())){
                return i;
            }
        }
        System.out.println("No such customer");
        return -1;
    }
```

```
    @Override
    public String toString() {
        StringBuilder result = new StringBuilder("TelephoneStation = {\n");
        result.append(String.format("customerList: %d\ncustomerList = {\n",
customerCount));
        for (Customer customerList : getCustomerList()) {
            result.append(customerList.toString()).append("\n");
        }
        result.append("}\n}");
        return result.toString();
    }

    public int getCustomerCount() {
        return customerCount;
    }

    public void setCustomerList(Customer[] customerList) {
        this.customerList = customerList;
        this.customerCount = customerList.length;
    }

    public Customer[] getCustomerList() {
        return customerList;
    }
}
```

Код общего класса Main:

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

        Scanner in = new Scanner(System.in);

        // Вариант 1 задание 6

        System.out.print("Enter x for first continued fraction: ");
        int x = Integer.parseInt(in.nextLine());
        System.out.print("Enter n for first continued fraction: ");
        int n = Integer.parseInt(in.nextLine());
        int[] a = new int[n];
        for (int i = 0; i < n; i++) {
            System.out.print(String.format("Enter %d element of array for
first continued fraction: ", i+1));
            a[i] = Integer.parseInt(in.nextLine());
        }
        ContinuedFraction firstFraction = new ContinuedFraction(x, n, a);
        System.out.println("Value of first continued fraction: " +
firstFraction.getResult());
        System.out.print("Enter x for second continued fraction: ");
        x = Integer.parseInt(in.nextLine());
        System.out.print("Enter n for second continued fraction: ");
        n = Integer.parseInt(in.nextLine());
        a = new int[n];
        for (int i = 0; i < n; i++) {
            System.out.print(String.format("Enter %d element of array for
second continued fraction: ", i+1));
            a[i] = Integer.parseInt(in.nextLine());
        }
        ContinuedFraction secondFraction = new ContinuedFraction(x, n, a);
        System.out.println("Value of second continued fraction: " +
secondFraction.getResult());
        System.out.println("Addition of two continued fractions: " +
```

```
ContinuedFraction.add(firstFraction, secondFraction).getResult());
        System.out.println("Subtraction of two continued fractions: " +
ContinuedFraction.subtract(firstFraction, secondFraction).getResult());
        System.out.println("Multiplication of two continued fractions: " +
ContinuedFraction.multiply(firstFraction, secondFraction).getResult());
        System.out.println("Division of two continued fractions: " +
ContinuedFraction.divide(firstFraction, secondFraction).getResult());

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

        // Вариант 1 задание 7

        System.out.print("Enter length of array: ");
        n = Integer.parseInt(in.nextLine());
        Fraction[] fractionArray = new Fraction[n];
        for (int i = 0; i < n; i++){
            System.out.print("Enter " + i+1 + " numerator: ");
            int numerator = Integer.parseInt(in.nextLine());
            System.out.print("Enter " + i+1 + " divider: ");
            int divider = Integer.parseInt(in.nextLine());
            fractionArray[i] = Fraction.correction(numerator, divider);
        }

        System.out.print("Original array of fraction: ");
        for (int i = 0; i < n; i++){
            System.out.print(fractionArray[i].getNumerator() + "/" +
fractionArray[i].getDivider() + " ");
        }
        System.out.println();

        System.out.print("Changed array of fraction: ");
        Fraction.method(fractionArray);
        for (int i = 0; i < n; i++){
            System.out.print(fractionArray[i].getNumerator() + "/" +
fractionArray[i].getDivider() + " ");
        }
        System.out.println();

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

        // Вариант 2 задание 6

        System.out.print("Enter count of houses: ");
        n = Integer.parseInt(in.nextLine());
        House[] houseList = new House[n];
        for (int i = 0; i < n; i++){
            System.out.print(String.format("Enter number of %d house: ",
i+1));
            int houseNumber = Integer.parseInt(in.nextLine());
            System.out.print(String.format("Enter square of %d house: ",
i+1));
            double square = Double.parseDouble(in.nextLine());
            System.out.print(String.format("Enter floor of %d house: ",
i+1));
            int floor = Integer.parseInt(in.nextLine());
            System.out.print(String.format("Enter count of rooms of %d house:
", i+1));
            int roomsCount = Integer.parseInt(in.nextLine());
            houseList[i] = new House(i+1, houseNumber, square, floor,
roomsCount);
        }
        System.out.println("Original list of houses:");
```

```
        for (int i = 0; i < n; i++){
            System.out.println(houseList[i].toString());
        }

        System.out.print("Enter count of rooms, that should be searched by:
");
        int count = Integer.parseInt(in.nextLine());
        System.out.println(String.format("Houses, that have %d rooms:",
count));
        for (int i = 0; i < n; i++){
            if (houseList[i].getRoomsCount() == count){
                System.out.println(houseList[i].toString());
            }
        }

        System.out.print("Enter bottom floor limit, that should be searched
by: ");
        int bottomFloorLimit = Integer.parseInt(in.nextLine());
        System.out.print("Enter upper floor limit, that should be searched
by: ");
        int upperFloorLimit = Integer.parseInt(in.nextLine());
        System.out.println(String.format("Houses, that have %d rooms and are
below %d and %d floor:",
                count, bottomFloorLimit, upperFloorLimit));
        for (int i = 0; i < n; i++){
            if ((houseList[i].getRoomsCount() == count) &&
(houseList[i].getFloor() >= bottomFloorLimit) &&
                    (houseList[i].getFloor() <= upperFloorLimit)){
                System.out.println(houseList[i].toString());
            }
        }

        System.out.print("Enter square, that should be searched by: ");
        double squareCompare = Double.parseDouble(in.nextLine());
        System.out.println(String.format("Houses, which square is more than
%f:", squareCompare));
        for (int i = 0; i < n; i++){
            if (houseList[i].getSquare() > squareCompare){
                System.out.println(houseList[i].toString());
            }
        }

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

        // Вариант 2 задание 7

        System.out.print("Enter count of phones: ");
        n = Integer.parseInt(in.nextLine());
        Phone[] phoneBook = new Phone[n];
        for (int i = 0; i < n; i++){
            System.out.print(String.format("Enter surname of %d person: ",
i+1));
            String surname = in.nextLine();
            System.out.print(String.format("Enter name of %d person: ",
i+1));
            String name = in.nextLine();
            System.out.print(String.format("Enter patronymic of %d person: ",
i+1));
            String patronymic = in.nextLine();
            System.out.print(String.format("Enter city call time of %d person
in seconds: ", i+1));
            int cityCallTimeInSeconds = Integer.parseInt(in.nextLine());
            System.out.print(String.format("Enter intercity call time of %d
```

```
person in seconds: ", i+1));
            int intercityCallTimeInSeconds = Integer.parseInt(in.nextLine());
            phoneBook[i] = new Phone(i+1, surname, name, patronymic,
cityCallTimeInSeconds, intercityCallTimeInSeconds);
        }
        System.out.println("Original phonebook:");
        for (int i = 0; i < n; i++){
            System.out.println(phoneBook[i].toString());
        }

        System.out.print("Enter time in seconds to compare with city call
times: ");
        int time = Integer.parseInt(in.nextLine());
        System.out.println("People, who call in city over given time:");
        for (int i = 0; i < n; i++){
            if (phoneBook[i].getCityCallTimeInSeconds() > time){
                System.out.println(phoneBook[i].toString());
            }
        }

        System.out.println("People, who use intercity calls:");
        for (int i = 0; i < n; i++){
            if (phoneBook[i].getCityCallTimeInSeconds() != 0){
                System.out.println(phoneBook[i].toString());
            }
        }

        System.out.println("Sorted phonebook:");
        Phone.sort(phoneBook, 0, n-1);
        for (int i = 0; i < n; i++){
            System.out.println(phoneBook[i].toString());
        }

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

        // Вариант 3 задание 6

        Rose rose = new Rose();
        System.out.println("Rose bloomed");
        System.out.println(rose);
        System.out.print("Color of rose: ");
        System.out.println(rose.getColor());
        System.out.println("Rose withered");
        rose.wither();
        System.out.println(rose);

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

        // Вариант 3 задание 7

        Tree tree = new Tree();
        System.out.println("Tree bloomed");
        System.out.println(tree);
        System.out.println("Leaves of tree are getting yellow");
        tree.turnYellow();
        System.out.println(tree);
        System.out.println("Leaves of tree are covered with frost");
        tree.coverWithFrost();
        System.out.println(tree);
        System.out.println("Tree withered");
        tree.wither();
        System.out.println(tree);
```

```
        System.out.println("-------------------------------------------------
-----------------------------------------------------------------");

        // Вариант 4 задание 6

        System.out.print("Welcome to design department. ");
        DesignDepartment designDepartment = new DesignDepartment();
        int choice;
        do {
            System.out.println("""
                What do you want to do?
                1 - See all projects
                2 - Add project
                3 - Invoice for project
                4 - Make constructor team
                5 - Exit""");
            System.out.print("Enter your choice: ");
            choice = in.nextInt();
            switch (choice){
                case 1: System.out.println(designDepartment); break;
                case 2: designDepartment.addProject(); break;
                case 3: designDepartment.showInvoice(); break;
                case 4: designDepartment.setProjectTeam(); break;
                case 5: System.out.println("Thanks for using our program. See
you later."); break;
                default: System.out.println("Wrong choice input. Try
again.");
            }
        } while (choice != 5);

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

        // Вариант 4 задание 7

        System.out.print("Welcome to telephone station. ");
        TelephoneStation telephoneStation = new TelephoneStation();
        do {
            System.out.println("""
                What do you want to do?
                1 - See all customers
                2 - Add customer
                3 - Change number of customer
                4 - Change services of customer
                5 - Repay the payment made by the client
                6 - Make payments for services to all clients
                7 - Check balance of client
                8 - Exit""");
            System.out.print("Enter your choice: ");
            choice = in.nextInt();
            switch (choice){
                case 1: System.out.println(telephoneStation); break;
                case 2: telephoneStation.addCustomer(); break;
                case 3: telephoneStation.changeNumber(); break;
                case 4: telephoneStation.changeServices(); break;
                case 5: telephoneStation.repayPayment(); break;
                case 6: telephoneStation.payForServices(); break;
                case 7: telephoneStation.checkBalance(); break;
                case 8: System.out.println("Thanks for using our program. See
you later."); break;
                default: System.out.println("Wrong choice input. Try
again.");
            }
```

```
        } while (choice != 8);

    }
}
```

Работа программы представлена на рисунках 1-4.

```
Enter x for first continued fraction: 2
Enter n for first continued fraction: 3
Enter 1 element of array for first continued fraction: 4
Enter 2 element of array for first continued fraction: 5
Enter 3 element of array for first continued fraction: 6
Value of first continued fraction: 4.380952380952381
Enter x for second continued fraction: 4
Enter n for second continued fraction: 5
Enter 1 element of array for second continued fraction: 7
Enter 2 element of array for second continued fraction: 2
Enter 3 element of array for second continued fraction: 4
Enter 4 element of array for second continued fraction: 1
Enter 5 element of array for second continued fraction: 3
Value of second continued fraction: 8.53191489361702
Addition of two continued fractions: 12.912867274569402
Subtraction of two continued fractions: -4.1509625126646394
Multiplication of two continued fractions: 37.377912867274574
Division of two continued fractions: 0.5134782092388078
-----------------------------------------------------------------------------------------------------------
Enter length of array: 3
Enter 01 numerator: 1
Enter 01 divider: 2
Enter 11 numerator: 1
Enter 11 divider: 3
Enter 21 numerator: 4
Enter 21 divider: 5
Original array of fraction: 1/2 1/3 4/5
Changed array of fraction: 5/6 17/15 4/5
```

Рисунок 1 – Работа программы (1 вариант 6 и 7 задания)

```
Enter count of houses: 1
Enter number of 1 house: 88
Enter square of 1 house: 65
Enter floor of 1 house: 4
Enter count of rooms of 1 house: 2
Original list of houses:
House = {
    id: 1
    houseNumber: 88
    square: 65,000000
    floor: 4
    roomsCount: 2
    }
Enter count of rooms, that should be searched by: 2
Houses, that have 2 rooms:
House = {
    id: 1
    houseNumber: 88
    square: 65,000000
    floor: 4
    roomsCount: 2
    }
Enter bottom floor limit, that should be searched by: 5
Enter upper floor limit, that should be searched by: 6
Houses, that have 2 rooms and are below 5 and 6 floor:
Enter square, that should be searched by: 78
Houses, which square is more than 78,000000:
```

Рисунок 2 – Работа программы (2 вариант 6 задание)

```
Rose bloomed
Rose = {
    countPetal: 5
    petals = {
        Petal = { color: Pink }
        Petal = { color: Pink }
        Petal = { color: Pink }
        Petal = { color: Pink }
        Petal = { color: Pink }
    }
}
Color of rose: Pink
Rose withered
Rose = {
    countPetal: 0
    petals = {
    }
}
```

Рисунок 3 – Работа программы (3 вариант 6 задание)

```
Welcome to telephone station. What do you want to do?
1 - See all customers
2 - Add customer
3 - Change number of customer
4 - Change services of customer
5 - Repay the payment made by the client
6 - Make payments for services to all clients
7 - Check balance of client
8 - Exit
Enter your choice: 2
Enter customer's surname: Kushnir
Enter customer's name: Gregory
Enter customer's phone number: 88005553535
Enter customer's balance: 1400
What do you want to do?
1 - See all customers
2 - Add customer
3 - Change number of customer
4 - Change services of customer
5 - Repay the payment made by the client
6 - Make payments for services to all clients
7 - Check balance of client
8 - Exit
Enter your choice: 1
TelephoneStation = {
customerList: 1
customerList = {
Customer = {
    id: 1
    surname: Kushnir
    name: Gregory
    balance: 1400,000000
    state: true
    servicesCount: 0
    services = {
    }
}
}
}
```

Рисунок 4 – Работа программы (4 вариант 7 задания)

Вывод: Были освоены принципы ООП на языке программирования Java.