



**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 № 5

Название: Классы, наследование и полиморфизм

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

А.О.Крейденко

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2024

Цель: освоить принципы ООП на языке программирования Java.

Задание 1: определить класс Матрица размерности (m x n). Класс должен содержать несколько конструкторов. Объявить массив объектов. Передать объекты в метод, меняющий местами строки с максимальным и минимальным элементами k-го столбца. Создать метод, который изменяет i-ю матрицу путем возведения ее в квадрат.

Код класса Main:

```
public class Main {
    public static void main(String[] args) {
        Matrix[] matrices = new Matrix[4];

        matrices[0] = new Matrix(2, 2);

        double[][] array1 = {{3, 5}, {9, 2}, {13, 4}};
        matrices[1] = new Matrix(3, 2, array1);

        double[][] array2 = {{16, 2, 8}, {7, 5, 1}, {23, 30, 13}};
        matrices[2] = new Matrix(3, 3, array2);

        double[][] array3 = {{1, 12}, {5, 3}};
        matrices[3] = new Matrix(2, 2, array3);

        // задаем номер столбца для метода перестановки строк
        int k = 0;
        for(int i = 0; i < 4; ++i){
            matrices[i].swap_rows(k);
        }
        for(int i = 0; i < 4; ++i) {
            System.out.printf("Matrix № %d\n", i + 1);
            matrices[i].print();
            System.out.println();
        }

        // возведение матрицы в квадрат
        matrices[1].make_squared();

        matrices[2].make_squared();
        System.out.println("Матрица № 3, возведенная в квадрат:");
        matrices[2].print();
    }
}
```

Код класса Matrix:

```
public class Matrix {
    private final int rows;
    private final int columns;
    private double[][] matrix;
    public Matrix(int m, int n) {
```

```

        this.rows = m;
        this.columns = n;
        this.matrix = new double[rows][columns];
        for(int i = 0; i < m; ++i) {
            for(int j = 0; j < n; ++j){
                this.matrix[i][j] = 0;
            }
        }
    }

    public Matrix(int m, int n, double[][] array) {
        this.rows = m;
        this.columns = n;
        this.matrix = new double[rows][columns];

        for(int i = 0; i < m; ++i) {
            System.arraycopy(array[i], 0, this.matrix[i], 0, n);
        }
    }

    public void swap_rows(int k) {
        double min, max;
        min = max = this.matrix[0][k];
        int min_row = 0, max_row = 0;

        // поиск строк с минимальным и максимальным элементом в k-
        столбце
        for(int i = 1; i < this.rows; ++i) {
            if(this.matrix[i][k] > max) {
                max = this.matrix[i][k];
                max_row = i;
            } else if (this.matrix[i][k] < min) {
                min = this.matrix[i][k];
                min_row = i;
            }
        }

        // перестановка строк
        for(int j = 0; j < this.columns; ++j) {
            double temp = this.matrix[min_row][j];
            this.matrix[min_row][j] = this.matrix[max_row][j];
            this.matrix[max_row][j] = temp;
        }
    }

    public void make_squared() {
        if(this.rows != this.columns) {
            System.out.println("Матрицу нельзя возвести в
квадрат, т.к количество строк и столбцов не равно");
            return;
        }
        double[][] temp_matrix = new
double[this.rows][this.columns];

```

```

        for(int i = 0; i < this.rows; ++i) {
            for(int j = 0; j < this.columns; ++j) {
                int sum = 0;
                for(int z = 0; z < this.rows; ++z) {
                    sum += this.matrix[i][z] * this.matrix[z][j];
                }
                temp_matrix[i][j] = sum;
            }
        }

        for(int i = 0; i < this.rows; ++i) {
            System.arraycopy(temp_matrix[i], 0, this.matrix[i],
0, this.columns);
        }
    }

    public void print() {
        for(int i = 0; i < this.rows; ++i) {
            for(int j = 0; j < this.columns; ++j) {
                System.out.printf("%.1f ", this.matrix[i][j]);
            }
            System.out.println();
        }
    }
}

```

Работа программы показана на рисунке 1.

```

C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe "-javaagent:C:\Program Files\JetBr
Matrix № 1
0,0 0,0
0,0 0,0

Matrix № 2
13,0 4,0
9,0 2,0
3,0 5,0

Matrix № 3
16,0 2,0 8,0
23,0 30,0 13,0
7,0 5,0 1,0

Matrix № 4
5,0 3,0
1,0 12,0

Матрицу нельзя возвести в квадрат, т.к количество строк и столбцов не равно
Матрица № 3, возведенная в квадрат:
358,0 132,0 162,0
1149,0 1011,0 587,0
234,0 169,0 122,0

Process finished with exit code 0

```

Рисунок 1 – Работа программы 1

Задание 2: определить класс Цепная дробь:

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

Определить методы сложения, вычитания, умножения, деления.

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

Код класса ContinuedFracion:

```
public class ContinuedFracion {
    private final double constant;
    private final double[] coefficients;
    private final int coef_number;
    public ContinuedFracion(double x, double[] coefs) {
        this.constant = x;
        this.coefficients = new double[coefs.length];
        this.coef_number = coefs.length;
        System.arraycopy(coefs, 0, this.coefficients, 0,
coefs.length);
    }

    public double calculate(int n) {
        double value = this.coefficients[n - 1];
        for(int i = n - 2; i >= 0; --i) {
            value = this.coefficients[i] + this.constant / value;
        }
        return value;
    }

    public double sum(ContinuedFracion fraction) {
        return this.calculate(this.coef_number) +
fraction.calculate(fraction.coef_number);
    }

    public double diff(ContinuedFracion fraction) {
        return this.calculate(this.coef_number) -
fraction.calculate(fraction.coef_number);
    }

    public double multiplication(ContinuedFracion fraction) {
        return this.calculate(this.coef_number) *
fraction.calculate(fraction.coef_number);
    }

    public double divison(ContinuedFracion fraction) {
        return this.calculate(this.coef_number) /
fraction.calculate(fraction.coef_number);
    }
    public static void main(String[] args) {
        double[] coefs1 = {1, 3, 2 ,5};
    }
}
```

```

        ContinuedFraction fraction1 = new ContinuedFraction(2,
coefs1);

        double[] coefs2 = {3, 4, 1, 2};
        ContinuedFraction fraction2 = new ContinuedFraction(2,
coefs2);

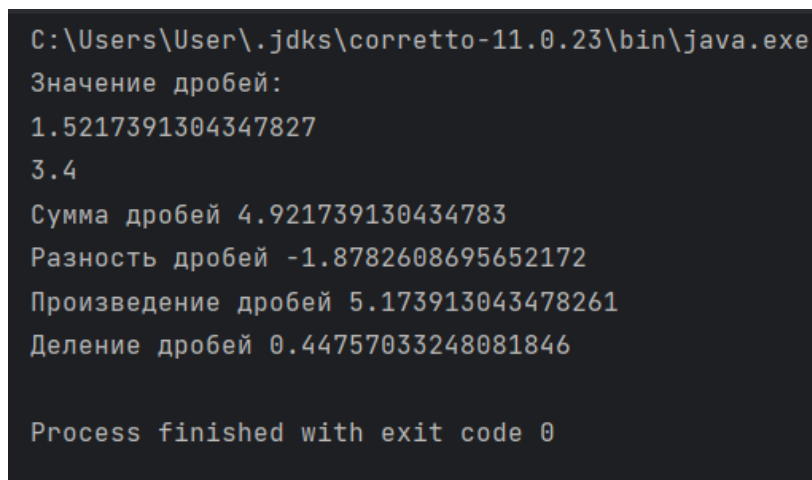
        System.out.println("Значение дробей:");
        System.out.println(fraction1.calculate(4));
        System.out.println(fraction2.calculate(4));

        System.out.println("Сумма          дробей          "          +
fraction1.sum(fraction2));
        System.out.println("Разность          дробей          "          +
fraction1.diff(fraction2));
        System.out.println("Произведение          дробей          "          +
fraction1.multiplication(fraction2));
        System.out.println("Деление          дробей          "          +
fraction1.divison(fraction2));

    }
}

```

Работа программы показана на рисунке 2.



```

C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe
Значение дробей:
1.5217391304347827
3.4
Сумма дробей 4.921739130434783
Разность дробей -1.8782608695652172
Произведение дробей 5.173913043478261
Деление дробей 0.44757033248081846

Process finished with exit code 0

```

Рисунок 2 – Работа программы 2

Задание 3: 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. **Book:** `id`, `Название`, `Автор(ы)`, `Издательство`, `Год издания`, `Количество страниц`, `Цена`, `Переплет`. Создать массив объектов. Вывести: а) список книг заданного автора; б) список книг,

выпущенных заданным издательством; с) список книг, выпущенных после заданного года.

Код класса Main:

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        ArrayList<Book> books = new ArrayList<>();
        books.add(new Book(0, "Энциклопедия", new
String[]{"Иванов"}, "Издательство1",
        2003, 350, 1000, "Твердый"));
        books.add(new Book(1, "Учебник математики", new
String[]{"Петров", "Иванов"}, "Издательство2",
        2007, 200, 570, "Твердый"));
        books.add(new Book(2, "Детектив", new String[]{"Конан
Дойл"}, "Издательство1",
        2010, 167, 400, "Мягкий"));
        books.add(new Book(3, "Словарь", new String[]{"Петров"},
        "Издательство3",
        2001, 300, 250, "Мягкий"));

        Scanner scan = new Scanner(System.in);
        // вывод книг заданного автора
        System.out.println("Введите имя автора:");
        String author = scan.nextLine();
        ArrayList<Book> filtered_author_books =
filter_by_author(books, author);
        if(filtered_author_books.size() == 0) {
            System.out.println("Книг заданного автора нет");
        }
        else {
            for(Book b: filtered_author_books) {
                System.out.println(b.toString());
            }
        }

        // вывод книг заданного издательства
        System.out.println("Введите название издательства:");
        String publisher = scan.nextLine();
        ArrayList<Book> filtered_publisher_books =
filter_by_publisher(books, publisher);
        if(filtered_publisher_books.size() == 0) {
            System.out.println("Книг, выпущенных заданным
издательством, нет");
        }
        else {
            for(Book b: filtered_publisher_books) {
                System.out.println(b.toString());
            }
        }
    }
}
```

```

        // вывод книг, выпущенных после заданного года
        System.out.println("Введите минимальный год выпуска");
        int year = scan.nextInt();
        ArrayList<Book> filtered_year_books =
filter_by_year(books, year);
        if(filtered_author_books.size() == 0) {
            System.out.println("Книг, выпущенных после заданного
года, нет");
        }
        else {
            for(Book b: filtered_year_books) {
                System.out.println(b.toString());
            }
        }
    }

    public static ArrayList<Book>
filter_by_author(ArrayList<Book> books, String author) {
        ArrayList<Book> filtered_books = new ArrayList<>();
        for(Book book: books) {
            boolean found_author = false;
            for(String auth: book.getAuthors()) {
                if(auth.equals(author)) {
                    found_author = true;
                    break;
                }
            }
            if(found_author) {
                filtered_books.add(book);
            }
        }
        return filtered_books;
    }

    public static ArrayList<Book>
filter_by_publisher(ArrayList<Book> books, String publisher) {
        ArrayList<Book> filtered_books = new ArrayList<>();
        for(Book book: books) {
            if(book.getPublisher().equals(publisher)) {
                filtered_books.add(book);
            }
        }
        return filtered_books;
    }

    public static ArrayList<Book> filter_by_year(ArrayList<Book>
books, int year) {
        ArrayList<Book> filtered_books = new ArrayList<>();
        for(Book book: books) {
            if(book.getYear_publication() >= year) {
                filtered_books.add(book);
            }
        }
    }

```



```

        }
        return filtered_books;
    }
}

```

Код класса Book:

```

public class Book {
    private int id;
    private String title;
    private String[] authors;
    private String publisher;
    private int year_publication;
    private int page_number;
    private float price;
    private String cover_type;

    public Book(int id, String title, String[] authors, String
publisher, int year_publication,
                int page_number, float price, String cover_type)
    {
        this.id = id;
        this.title = title;
        this.authors = authors;
        this.publisher = publisher;
        this.year_publication = year_publication;
        this.page_number = page_number;
        this.price = price;
        this.cover_type = cover_type;
    }

    @Override
    public String toString() {
        return "Книга: '" + this.title + "', " + String.join(", ",
this.authors) +
                ", изд." + this.publisher + ", " + " +
this.year_publication + "г., " + this.page_number +
                " стр., переплет: " + this.cover_type + ", " +
this.price + " руб.";
    }
}

```

```

    }

    public int getId() {
        return id;
    }

    public String getTitle() {
        return this.title;
    }

    public String[] getAuthors() {
        return authors;
    }

    public String getPublisher() {
        return publisher;
    }

    public int getYear_publication() {
        return year_publication;
    }

    public int getPage_number() {
        return page_number;
    }

    public float getPrice() {
        return price;
    }

    public String getCover_type() {
        return cover_type;
    }

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

```

```

    }

    public void setTitle(String title) {
        this.title = title;
    }

    public void setAuthors(String[] authors) {
        this.authors = authors;
    }

    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }

    public void setYear_publication(int year_publication) {
        this.year_publication = year_publication;
    }

    public void setPage_number(int page_number) {
        this.page_number = page_number;
    }

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

    public void setCover_type(String cover_type) {
        this.cover_type = cover_type;
    }
}

```

Работа программы показана на рисунке 3.

```

Введите имя автора:
Иванов
Книга: 'Энциклопедия', Иванов, изд.Издательство1, 2003г., 350 стр., переплет: Твердый, 1000.0 руб.
Книга: 'Учебник математики', Петров, Иванов, изд.Издательство2, 2007г., 200 стр., переплет: Твердый, 570.0 руб.
Введите название издательства:
Издательство1
Книг, выпущенных заданным издательством, нет
Введите минимальный год выпуска
2000
Книга: 'Энциклопедия', Иванов, изд.Издательство1, 2003г., 350 стр., переплет: Твердый, 1000.0 руб.
Книга: 'Учебник математики', Петров, Иванов, изд.Издательство2, 2007г., 200 стр., переплет: Твердый, 570.0 руб.
Книга: 'Детектив', Конан Дойл, изд.Издательство1, 2010г., 167 стр., переплет: Мягкий, 400.0 руб.
Книга: 'Словарь', Петров, изд.Издательство3, 2001г., 300 стр., переплет: Мягкий, 250.0 руб.

Process finished with exit code 0

```

Рисунок 3 – Работа программы 3

Задание 4: 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. House: id, Номер квартиры, Площадь, Этаж, Количество комнат, Улица, Тип здания, Срок эксплуатации. Создать массив объектов. Вывести: а) список квартир, имеющих заданное число комнат; б) список квартир, имеющих заданное число комнат и расположенных на этаже, который находится в заданном промежутке; с) список квартир, имеющих площадь, превосходящую заданную.

Код класса Main:

```

import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        ArrayList<House> houses = new ArrayList<>();
        houses.add(new House(0, 18, 60, 5, 2,
            "Бауманская", "Тип1", 10));
        houses.add(new House(1, 10, 77, 3, 3,
            "Тверская", "Тип2", 30));
        houses.add(new House(2, 20, 100, 5, 4,
            "Арбат", "Тип1", 60));
        houses.add(new House(3, 4, 60, 1, 2,
            "Бауманская", "Тип3", 10));

        Scanner scan = new Scanner(System.in);
        // вывод списка квартир с заданным числом комнат
        System.out.println("Введите необходимое число комнат:");
        int num = scan.nextInt();
        ArrayList<House> houses_with_num_rooms =
        filter_by_room_number(houses, num);
    }
}

```

```

        if(houses_with_num_rooms.size() == 0) {
            System.out.println("Квартир с заданным числом комнат
нет");
        }
        else {
            for(House h: houses_with_num_rooms) {
                System.out.println(h.toString());
            }
        }

        // вывод списка квартир с заданным числом комнат и
        расположенных на этаже в интервале
        System.out.println("Введите минимальный возможный
этаж:");
        int min_floor = scan.nextInt();
        System.out.println("Введите максимально возможный
этаж:");
        int max_floor = scan.nextInt();
        ArrayList<House> houses_filtered_floor =
filter_by_floor(houses_with_num_rooms, min_floor, max_floor);
        if(houses_filtered_floor.size() == 0) {
            System.out.println("Квартир с заданным числом комнат
и номером этажа в заданном интервале нет");
        }
        else {
            for(House h: houses_filtered_floor) {
                System.out.println(h.toString());
            }
        }
        // вывод списка квартир, имеющих площадь, превосходящую
        заданную
        System.out.println("Введите минимально возможную
площадь:");
        int min_area = scan.nextInt();
        ArrayList<House> houses_filtered_area =
filter_by_area(houses, min_area);
        if(houses_filtered_area.size() == 0) {
            System.out.println("Квартир с площадью, превосходящую
заданную, нет");
        }
        else {
            for(House h: houses_filtered_area) {
                System.out.println(h.toString());
            }
        }
    }

    public static ArrayList<House>
filter_by_room_number(ArrayList<House> houses, int n) {
        ArrayList<House> filtered_houses = new ArrayList<>();
        for(House house: houses) {
            if(house.getRooms_number() == n) {

```

```

        filtered_houses.add(house);
    }
}
return filtered_houses;
}

    public static ArrayList<House>
filter_by_floor(ArrayList<House> houses, int min_floor, int
max_floor) {
    ArrayList<House> filtered_houses = new ArrayList<>();
    for(House house: houses) {
        if(house.getFloor() >= min_floor && house.getFloor()
<= max_floor) {
            filtered_houses.add(house);
        }
    }
    return filtered_houses;
}

    public static ArrayList<House>
filter_by_area(ArrayList<House> houses, int area) {
    ArrayList<House> filtered_houses = new ArrayList<>();
    for(House house: houses) {
        if(house.getArea() >= area) {
            filtered_houses.add(house);
        }
    }
    return filtered_houses;
}
}

```

Код класса House:

```

public class House {
    private int id;
    private int number;
    private float area;
    private int floor;
    private int rooms_number;
    private String street_name;
    private String building_type;
    private float exploit_period;

    public House(int id, int number, float area, int floor, int
rooms_number,
                String street_name, String building_type, float
exploit_period) {
        this.id = id;
        this.number = number;
        this.area = area;
        this.floor = floor;
        this.rooms_number = rooms_number;
        this.street_name = street_name;
        this.building_type = building_type;
        this.exploit_period = exploit_period;
    }
}

```

```

    }

    @Override
    public String toString() {
        return "Квартира № " + this.number + ", площадь " +
this.area + "м^2, " +
        this.floor + " этаж, " + this.rooms_number + "
комнат, ул." + this.street_name +
        ", типа здания: " + this.building_type + ", срок
эксплуатации: " +
        this.exploit_period + " лет.";
    }

    public int getId() {
        return id;
    }

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

    public int getNumber() {
        return number;
    }

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

    public float getArea() {
        return area;
    }

    public void setArea(float area) {
        this.area = area;
    }

    public int getFloor() {
        return floor;
    }

    public void setFloor(int floor) {
        this.floor = floor;
    }

    public int getRooms_number() {
        return rooms_number;
    }

    public void setRooms_number(int rooms_number) {
        this.rooms_number = rooms_number;
    }
}

```

```

    public String getStreet_name() {
        return street_name;
    }

    public void setStreet_name(String street_name) {
        this.street_name = street_name;
    }

    public String getBuilding_type() {
        return building_type;
    }

    public void setBuilding_type(String building_type) {
        this.building_type = building_type;
    }

    public float getExploit_period() {
        return exploit_period;
    }

    public void setExploit_period(float exploit_period) {
        this.exploit_period = exploit_period;
    }
}

```

Работа программы показана на рисунке 4.

```

C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Ed...
Введите необходимое число комнат:
3
Квартира № 10, площадь 77.0м^2, 3 этаж, 3 комнат, ул.Тверская, типа здания: Тип2, срок эксплуатации: 30.0 лет.
Введите минимальный возможный этаж:
4
Введите максимально возможный этаж:
10
Квартир с заданным числом комнат и номером этажа в заданном интервале нет
Введите минимально возможную площадь:
50
Квартира № 18, площадь 60.0м^2, 5 этаж, 2 комнат, ул.Бауманская, типа здания: Тип1, срок эксплуатации: 10.0 лет.
Квартира № 10, площадь 77.0м^2, 3 этаж, 3 комнат, ул.Тверская, типа здания: Тип2, срок эксплуатации: 30.0 лет.
Квартира № 20, площадь 100.0м^2, 5 этаж, 4 комнат, ул.Арбат, типа здания: Тип1, срок эксплуатации: 60.0 лет.
Квартира № 4, площадь 60.0м^2, 1 этаж, 2 комнат, ул.Бауманская, типа здания: Тип3, срок эксплуатации: 10.0 лет.
Process finished with exit code 0

```

Рисунок 4 – Работа программы 4

Задание 5: 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 Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей.

Код класса Main:

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

public class Main {
    public static void main(String[] args) {
        House house1 = new House();
        house1.addDoor(new Door());
        house1.addWindow(new Window());
        house1.addWindow(new Window());

        ArrayList<Window> windows = new ArrayList<>();
        windows.add(new Window());
        windows.add(new Window());
        ArrayList<Door> doors = new ArrayList<>();
        doors.add(new Door());
        House house2 = new House(windows, doors);

        // проверка работы методов hashCode, equals и toString
        System.out.println("1) " + house1.toString());
        System.out.println("Хэш дома №1: " + house1.hashCode());
        System.out.println("2) " + house2.toString());
        System.out.println("Хэш дома №2: " + house2.hashCode());
        System.out.println("Результат сравнения домов: " +
house1.equals(house2));
        System.out.println();

        house2.lock();
        System.out.println("Дом №1: " + house1.toString());
        System.out.println("Хэш дома №1: " + house1.hashCode());
        System.out.println("Дом №2: " + house2.toString());
        System.out.println("Хэш дома №2: " + house2.hashCode());
        System.out.println("Результат сравнения домов: " +
house1.equals(house2));
        System.out.println();
    }
}
```

Код класса House:

```
import java.util.ArrayList;
import java.util.Objects;

public class House {
    private boolean locked;
    private ArrayList<Window> windows;
    private ArrayList<Door> doors;

    public House() {
        this.locked = false;
        this.windows = new ArrayList<>();
        this.doors = new ArrayList<>();
    }
}
```

```

    public House(ArrayList<Window> windows, ArrayList<Door>
doors) {
        this.locked = false;
        this.windows = windows;
        this.doors = doors;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        House other = (House) o;
        if (Objects.equals(this.windows, other.windows)) return
false;
        if (Objects.equals(this.doors, other.doors)) return
false;
        return this.locked == other.locked;
    }

    @Override
    public int hashCode() {
        return Objects.hash(this.locked, this.doors.size(),
this.windows.size());
    }

    public String toString() {
        String lock_status;
        if (this.locked) lock_status = "закрыт.";
        else lock_status = "открыт.";
        return "Дом с количеством окон: " + this.windows.size() +
" шт. и количеством дверей: " +
            this.doors.size() + " шт. Дом " + lock_status;
    }

    public void lock() {
        this.locked = true;
        System.out.println("Дверь дома была закрыта");
    }

    public void open() {
        this.locked = false;
        System.out.println("Дверь дома была открыта");
    }

    public boolean getLockStatus() {
        return this.locked;
    }

    public void addWindow(Window win) {
        this.windows.add(win);
    }

    public void addDoor(Door door) {

```

```

        this.doors.add(door);
    }

    public void printWindowNumber() {
        System.out.println("Дом имеет " + this.windows.size() + "
окон.");
    }

    public void printDoorNumber() {
        System.out.println("Дом имеет " + this.doors.size() + "
дверей.");
    }
}

```

Код класса Door:

```

public class Door {
    public Door() {

    }
}

```

Код класса Window:

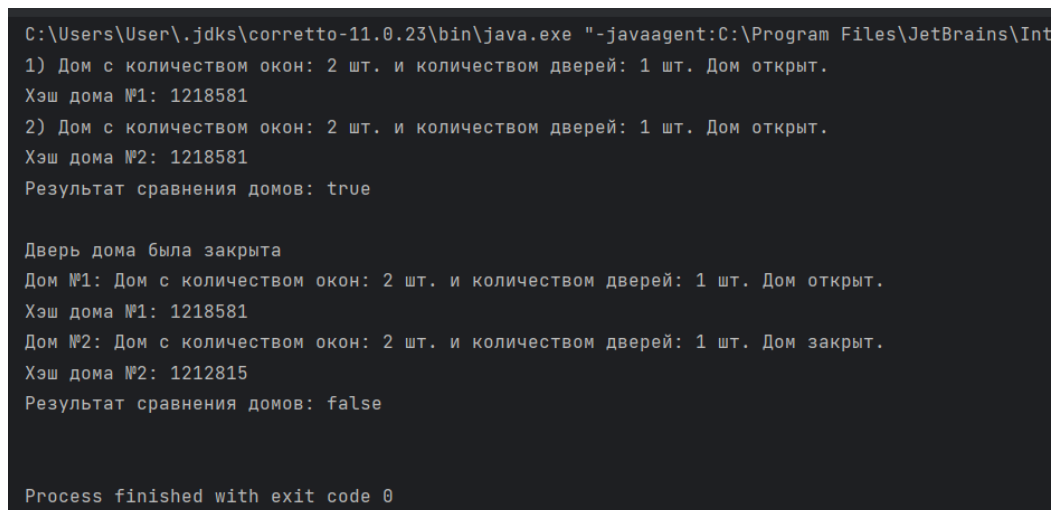
```

public class Window {
    public Window() {

    }
}

```

Работа программы показана на рисунке 5.



```

C:\Users\User\jdk\corretto-11.0.23\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Int
1) Дом с количеством окон: 2 шт. и количеством дверей: 1 шт. Дом открыт.
Хэш дома №1: 1218581
2) Дом с количеством окон: 2 шт. и количеством дверей: 1 шт. Дом открыт.
Хэш дома №2: 1218581
Результат сравнения домов: true

Дверь дома была закрыта
Дом №1: Дом с количеством окон: 2 шт. и количеством дверей: 1 шт. Дом открыт.
Хэш дома №1: 1218581
Дом №2: Дом с количеством окон: 2 шт. и количеством дверей: 1 шт. Дом закрыт.
Хэш дома №2: 1212815
Результат сравнения домов: false

Process finished with exit code 0

```

Рисунок 5 – Работа программы 5

Задание 6: 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.

Код класса Main:

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        Rose rose1 = new Rose();
        rose1.addPetal(new Petal());
        rose1.addPetal(new Petal());
        rose1.bloom();

        ArrayList<Petal> petals = new ArrayList<>();
        petals.add(new Petal());
        petals.add(new Petal());
        Rose rose2 = new Rose(petals, new Bud());
        rose2.bloom();

        rose2.printBudColor();
        System.out.println();

        // Проверка методов hashCode, equals, toString
        System.out.println("Информация о розах: ");
        System.out.println("1) " + rose1.toString());
        System.out.println("Хэш розы 1: " + rose1.hashCode());
        System.out.println("2) " + rose2.toString());
        System.out.println("Хэш розы 2: " + rose2.hashCode());
        System.out.println("Результат сравнения: " +
rose1.equals(rose2));
        System.out.println();

        rose2.wither();
        System.out.println("Информация о розах: ");
        System.out.println("1) " + rose1.toString());
        System.out.println("Хэш розы 1: " + rose1.hashCode());
        System.out.println("2) " + rose2.toString());
        System.out.println("Хэш розы 2: " + rose2.hashCode());
        System.out.println("Результат сравнения: " +
rose1.equals(rose2));

    }
}
```

Код класса Bud:

```
import java.util.Objects;

public class Bud {
    private String color;

    public Bud() {
        this.color = "Розовый";
    }

    public Bud(String color) {
        this.color = color;
    }
}
```

```

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

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

public String getColor() {
    return color;
}

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

```

Код класса Petal:

```

public class Petal {
    public Petal() {

    }
}

```

Код класса Rose:

```

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

public class Rose {
    private ArrayList<Petal> petals;
    private Bud bud;
    private int status;

    public Rose() {
        this.petals = new ArrayList<>();
        this.bud = new Bud();
        this.status = 0;
    }

    public Rose(ArrayList<Petal> p, Bud b) {
        this.petals = p;
        this.bud = b;
        this.status = 0;
    }

    @Override
    public String toString() {
        String rose_status = "";
        if (this.status == 0) rose_status = "не расцвела";
    }
}

```

```

        else if (this.status == 1) rose_status = "расцвела";
        else if (this.status == 2) rose_status = "завяла";

        return "Роза, количество лепестков: " + this.petals.size()
+
        ", цвет бутона: " + this.bud.getColor() + ",
состояние розы: " +
        rose_status + ".";
    }

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

    @Override
    public int hashCode() {
        return Objects.hash(this.petals.size(),
this.bud.getColor(), this.status);
    }

    public void bloom() {
        if (this.status == 0) {
            this.status = 1;
            System.out.println("Роза расцвела");
        } else if (this.status == 1) {
            System.out.println("Роза уже расцвела");
        }
        else {
            System.out.println("Роза уже завяла и не может
расцвести");
        }
    }

    public void wither() {
        if (this.status == 1) {
            this.status = 2;
            System.out.println("Роза завяла");
        } else if (this.status == 0) {
            System.out.println("Роза еще не расцвела");
        }
        else {
            System.out.println("Роза уже завяла");
        }
    }

    public void printBudColor() {
        System.out.println("Цвет бутона " + this.bud.getColor());
    }

```

```

    public void addPetal(Petal p) {
        this.petals.add(p);
    }
}

```

Работа программы показана на рисунке 6.

```

C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe -javaagent:C:\Program Files\JetBrain
Роза расцвела
Роза расцвела
Цвет бутона Розовый

Информация о розах:
1) Роза, количество лепестков: 2, цвет бутона: Розовый, состояние розы: расцвела.
Хэш розы 1: 1485210599
2) Роза, количество лепестков: 2, цвет бутона: Розовый, состояние розы: расцвела.
Хэш розы 2: 1485210599
Результат сравнения: true

Роза завяла
Информация о розах:
1) Роза, количество лепестков: 2, цвет бутона: Розовый, состояние розы: расцвела.
Хэш розы 1: 1485210599
2) Роза, количество лепестков: 2, цвет бутона: Розовый, состояние розы: завяла.
Хэш розы 2: 1485210600
Результат сравнения: false

Process finished with exit code 0

```

Рисунок 6 – Работа программы 6

Задание 7: построить модель программной системы. Система Библиотека. Читатель оформляет Заказ на Книгу. Система осуществляет поиск в Каталоге. Библиотекарь выдает Читателю Книгу на абонемент или в читальный зал. При невозвращении Книги Читателем он может быть занесен Администратором в «черный список».

Код класса Main:

```

import java.util.HashSet;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        Library library = new Library();
        Reader reader_1 = new Reader("Иван");
        Librarian librarian_1 = new Librarian("Петр");

        Book book_1 = new Book("Война и мир", "Л. Н. Толстой",
1000, 1873);
        Book book_2 = new Book("Евгений Онегин", "А. С. Пушкин",
450, 1830);
        Book book_3 = new Book("Мертвые души", "Н. В. Гоголь",
500, 1835);

```

```

HashSet<Book> books_catalog = new HashSet<Book>();
books_catalog.add(book_1);
books_catalog.add(book_2);
library.setCatalog(books_catalog);

librarian_1.issueBook(library, reader_1, book_1, true);
librarian_1.issueBook(library, reader_1, book_3, true);

System.out.println("Читатель " + reader_1.getName() + " в
черном списке: " + reader_1.getOnBlacklist());

System.out.println("Вернули ли читатель книгу?");
String reader_return = scanner.nextLine();
if (reader_return.equalsIgnoreCase("нет")) {
    librarian_1.addToBlackList(reader_1);
}

System.out.println("Читатель " + reader_1.getName() + " в
черном списке: " + reader_1.getOnBlacklist());
}
}

```

Код класса Book:

```

import java.util.Objects;

public class Book {
    private String title;
    private String author;
    private int page_num;
    private int creation_year;

    public Book(String title, String author, int page_num, int
creation_year) {
        this.title = title;
        this.author = author;
        this.page_num = page_num;
        this.creation_year = creation_year;
    }

    @Override
    public String toString() {
        return "Книга '" + this.title + "'", автора: " +
            this.author + ", " + this.creation_year + " г., "
+
            this.page_num + " стр.";
    }

    @Override
    public int hashCode() {
        return Objects.hash(this.title, this.author,
this.page_num, this.creation_year);
    }
}

```



```

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

    Book other = (Book) o;
    return this.title.equals(other.title) &&
this.author.equals(other.author) &&
        this.page_num == other.page_num &&
this.creation_year == other.creation_year;
}

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}

public String getAuthor() {
    return author;
}

public void setAuthor(String author) {
    this.author = author;
}

public int getPage_num() {
    return page_num;
}

public void setPage_num(int page_num) {
    this.page_num = page_num;
}

public int getCreation_year() {
    return creation_year;
}

public void setCreation_year(int creation_year) {
    this.creation_year = creation_year;
}
}

```

Код класса Librarian:

```

public class Librarian {
    private String name;

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

```

```

        public void issueBook(Library lib, Reader reader, Book book,
Boolean issue_type) {
            if (lib.checkBook(book) && !reader.getOnBlacklist()) {
                reader.addBook(book);
                System.out.print("Библиотекарь " + this.name + " выдал
книгу " +
                                book.getTitle() + " читателю " +
reader.getName());
                if (issue_type) {
                    System.out.println(" на дом");
                }
                else {
                    System.out.println(" в читальный зал");
                }
            }
            else {
                System.out.println("Книга " + book.getTitle() + " не
найдена в каталоге");
            }
        }

        public void addToBlackList(Reader reader) {
            reader.setOnBlacklist(true);
            System.out.println("Библиотекарь " + this.name + " добавил
читателя " +
                                reader.getName() + " в черный список");
        }

        public String getName() {
            return name;
        }

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

```

Код класса Library:

```

import java.util.*;

public class Library {
    private HashSet<Book> catalog;
    List<Reader> readers;

    public Library() {
        this.catalog = new HashSet<>();
        this.readers = new ArrayList<>();
    }

    public void addBook(Book new_book) {
        this.catalog.add(new_book);
    }

    public HashSet<Book> getCatalog() {
        return catalog;
    }
}

```

```

    }

    public Boolean checkBook(Book book) {
        return this.catalog.contains(book);
    }

    public void setCatalog(HashSet<Book> catalog) {
        this.catalog = catalog;
    }

    public List<Reader> getReaders() {
        return readers;
    }

    public void setReaders(List<Reader> readers) {
        this.readers = readers;
    }
}

```

Код класса Reader

```

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

public class Reader {
    private String name;
    private List<Book> books;

    private Boolean isOnBlacklist;
    public Reader(String name) {
        this.name = name;
        this.isOnBlacklist = false;
        this.books = new ArrayList<Book>();
    }

    public void addBook(Book b) {
        this.books.add(b);
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {

```

```

        this.name = name;
    }

    public List<Book> getBooks() {
        return books;
    }

    public void setBooks(List<Book> books) {
        this.books = books;
    }

    public Boolean getOnBlacklist() {
        return isOnBlacklist;
    }

    public void setOnBlacklist(Boolean onBlacklist) {
        isOnBlacklist = onBlacklist;
    }
}

```

Работа программы показана на рисунке 7.

```

C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe -javaagent:C:\Pro...
Библиотекарь Петр выдал книгу Война и мир читателю Иван на дом
Книга Мертвые души не найдена в каталоге
Читатель Иван в черном списке: false
Вернули ли читатель книгу?
Нет
Библиотекарь Петр добавил читателя Иван в черный список
Читатель Иван в черном списке: true
Process finished with exit code 0

```

Рисунок 7 – Работа программы 7

Задание 8: построить модель программной системы. 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 Main:

```
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        // инициализация заказчика
        Customer customer = new Customer("Алексей");

        // инициализация технического задания
        Map<String, Double> tasks = new HashMap<>();
        tasks.put("План помещения", 10000.0);
        tasks.put("Проводка", 5000.50);
        tasks.put("Дизайн", 2000.0);
        TechnicalSpecification ts = new
TechnicalSpecification("Конструкция 8-ми этажного здания");
        ts.setTasks(tasks);

        // инициализация конструкторов
        Designer designer1 = new Designer("Иван");
        Designer designer2 = new Designer("Петр");

        // расчет счета
        DesignBureau bureau = new DesignBureau();
        bureau.registerTechnicalAssignment(ts);
        Invoice invoice = bureau.createInvoice(customer, ts);

        // добавление конструкторов в бригаду
        ProjectTeam team = bureau.createProjectTeam();
        bureau.assignDesignerToTeam(team, designer1);
        bureau.assignDesignerToTeam(team, designer2);
    }
}
```

Код класса Customer:

```
public class Customer {
    private String name;

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

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

Код класса DesignBureau:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class DesignBureau {
    private List<TechnicalSpecification> technicalAssignments;
    private List<ProjectTeam> projectTeams;
```

```

    public DesignBureau() {
        this.technicalAssignments = new ArrayList<>();
        this.projectTeams = new ArrayList<>();
    }

    public void registerTechnicalAssignment(TechnicalSpecification ts) {
        technicalAssignments.add(ts);
        System.out.println("Зарегистрировано техническое задание: " + ts.getDescription());
    }

    public double determineCost(TechnicalSpecification ts) {
        double budget = 0;
        // Простой расчет стоимости, исходя из бюджета ТЗ
        for(Map.Entry<String, Double> task: ts.getTasks().entrySet()) {
            budget += task.getValue();
        }
        return budget;
    }

    public Invoice createInvoice(Customer customer, TechnicalSpecification ts) {
        double cost = determineCost(ts);
        Invoice invoice = new Invoice(cost);
        invoice.printInvoice(customer);
        return invoice;
    }

    public ProjectTeam createProjectTeam() {
        ProjectTeam team = new ProjectTeam();
        projectTeams.add(team);
        System.out.println("Создана новая бригада конструкторов.");
        return team;
    }

    public void assignDesignerToTeam(ProjectTeam team, Designer designer) {
        team.addDesigner(designer);
        System.out.println("Назначить конструктора " + designer.getName() + " в бригаду.");
    }
}

```

Код класса Designer:

```

public class Designer {
    private String name;

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

```

```

    }

    public String getName() {
        return name;
    }
}

```

Код класса Invoice:

```

public class Invoice {
    private double amount;

    public Invoice(double amount) {
        this.amount = amount;
    }

    public double getAmount() {
        return amount;
    }

    public void printInvoice(Customer customer) {
        System.out.println("Счет для " + customer.getName() + ":
" + amount + " руб.");
    }
}

```

Код класса ProjectTeam:

```

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

public class ProjectTeam {
    private List<Designer> designers;

    public ProjectTeam() {
        this.designers = new ArrayList<>();
    }

    public void addDesigner(Designer designer) {

```

```

        designers.add(designer);
    }

    public List<Designer> getDesigners() {
        return designers;
    }
}

```

Код класса TechnicalSpecification:

```

import java.util.HashMap;
import java.util.Map;

public class TechnicalSpecification {
    private String description;
    Map<String, Double> tasks;

    public TechnicalSpecification(String description) {
        this.description = description;
        this.tasks = new HashMap<>();
    }

    public void addTask(String title, Double price) {
        tasks.put(title, price);
    }

    public String getDescription() {
        return description;
    }

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

    public Map<String, Double> getTasks() {
        return tasks;
    }
}

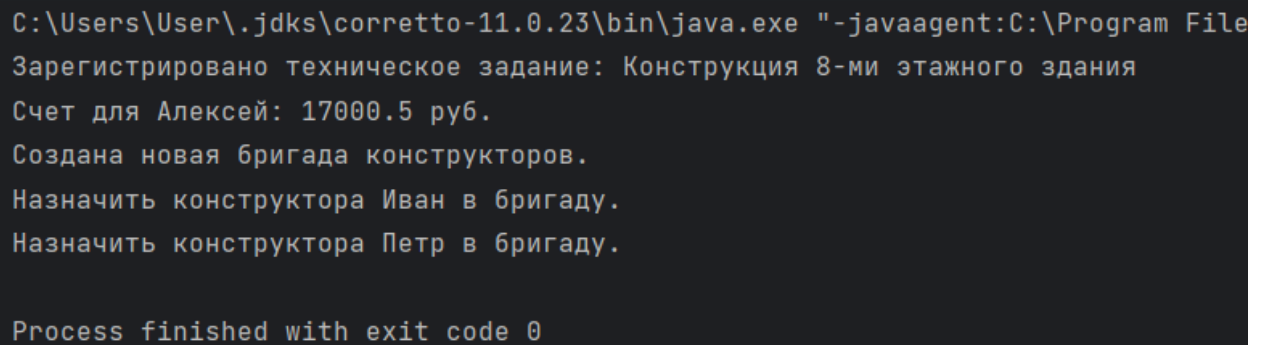
```



```
}

public void setTasks(Map<String, Double> tasks) {
    this.tasks = tasks;
}
}
```

Работа программы показана на рисунке 8.

A screenshot of a Windows command prompt window with a dark background and light-colored text. The text shows the execution of a Java program. The first line is the command: C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe -javaagent:C:\Program File. The subsequent lines are the program's output: 'Зарегистрировано техническое задание: Конструкция 8-ми этажного здания', 'Счет для Алексей: 17000.5 руб.', 'Создана новая бригада конструкторов.', 'Назначить конструктора Иван в бригаду.', and 'Назначить конструктора Петр в бригаду.'. The final line indicates the process finished with exit code 0.

```
C:\Users\User\.jdk\corretto-11.0.23\bin\java.exe -javaagent:C:\Program File
Зарегистрировано техническое задание: Конструкция 8-ми этажного здания
Счет для Алексей: 17000.5 руб.
Создана новая бригада конструкторов.
Назначить конструктора Иван в бригаду.
Назначить конструктора Петр в бригаду.

Process finished with exit code 0
```

Рисунок 8 – Работа программы 8

Вывод: были освоены принципы ООП на языке программирования Java