



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ  
УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.04.01 Информатика и вычислительная техника  
МАГИСТЕРСКАЯ ПРОГРАММА 09.04.01/07 Интеллектуальные системы анализа,  
обработки и интерпретации больших данных

## О Т Ч Е Т

по лабораторной работе №3

Название: Классы. Наследование. Полиморфизм

Дисциплина: Языки программирования для работы с большими  
данными

Студент

ИУ6-22М

(Группа)

(Подпись, дата)

А.М. Панфилкин

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2022 г.

Весь приведенный ниже код также доступен в следующем репозитории:

<https://github.com/SilkSlime/iu6plfbd>

**Задание 1:** Определить класс Матрица размерности ( $m \times n$ ). 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.

#### Листинг 1 – Задание 1

```
package l3;

public class e1 {

    /**
     * Вариант 1. Задача 5.
     * Определить класс Матрица размерности ( $m \times n$ ). 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.
     */

    public static void main(String[] args) {
        // Define an array of matrixes and fill it using constructors
        Matrix[] matrixes = new Matrix[3];
        matrixes[0] = new Matrix(3, 3);
        matrixes[1] = new Matrix(new int[][] {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}});
        matrixes[2] = new Matrix(new int[][] {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}});

        matrixes[1].print();

        // Change rows with max and min elements of  $k$ -th column
        matrixes[1].changeRowsMinMaxOfK(1);
        matrixes[1].print();

        // Change  $i$ -th matrix by squaring it
        matrixes[1].square(1, matrixes);
        matrixes[1].print();
    }
}

// Define class Matrix with size ( $m \times n$ ) and two constructors
class Matrix {
    private int m;
    private int n;
    private int[][] matrix;

    public Matrix(int m, int n) {
        this.m = m;
```

```

        this.n = n;
        matrix = new int[m][n];
    }

    public Matrix(int[][] matrix) {
        this.matrix = matrix;
        this.m = matrix.length;
        this.n = matrix[0].length;
    }

    // Method for printing matrix using System.out.println()
    public void print() {
        for (int i = 0; i < this.n; i++) {
            for (int j = 0; j < this.m; j++) {
                System.out.printf("%4d", this.matrix[i][j]);
            }
            System.out.println();
        }
    }

    // Method for changing rows with max and min elements of k-th column
    public void changeRowsMinMaxOfK(int k) {
        int max = matrix[0][k];
        int min = matrix[0][k];
        int maxRow = 0;
        int minRow = 0;

        for (int i = 0; i < m; i++) {
            if (matrix[i][k] > max) {
                max = matrix[i][k];
                maxRow = i;
            }
            if (matrix[i][k] < min) {
                min = matrix[i][k];
                minRow = i;
            }
        }

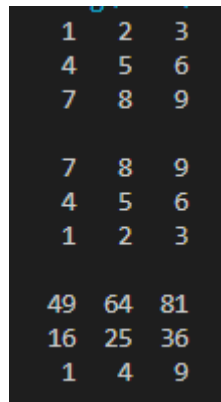
        int[] temp = matrix[maxRow];
        matrix[maxRow] = matrix[minRow];
        matrix[minRow] = temp;
    }

    // Method for changing i-th matrix by squaring it
    public void square(int i, Matrix[] matrixes) {
        int[][] temp = matrixes[i].matrix;
        for (int j = 0; j < m; j++) {
            for (int k = 0; k < n; k++) {
                temp[j][k] *= temp[j][k];
            }
        }
        matrixes[i].matrix = temp;
    }
}

```

```
}
```

Пример результата работы программы показан на рисунке 1.



```
1 2 3
4 5 6
7 8 9

7 8 9
4 5 6
1 2 3

49 64 81
16 25 36
1 4 9
```

Рисунок 1 – Пример результата работы программы

**Задание 2:** Определить класс Цепная дробь. Определить методы сложения, вычитания, умножения, деления. Вычислить значение для заданного  $n$ ,  $x$ ,  $a[n]$ .

Листинг 2 – Задание 2

```
package l3;

public class e2 {

    /**
     * Вариант 1. Задача 6.
     * Определить класс Цепная дробь. Определить методы сложения, вычитания,
     * умножения, деления. Вычислить значение для заданного  $n$ ,  $x$ ,  $a[n]$ .
     */

    public static void main(String[] args) {
        ChainFraction cf1 = new ChainFraction(3, 2, new int[] {1, 2, 3});
        cf1.print();
        System.out.println(cf1.value());

        ChainFraction cf2 = new ChainFraction(1.75, 3, 2);
        cf2.print();
        System.out.println(cf2.value());

        cf1.add(cf2);
        cf1.print();
        System.out.println(cf1.value());
    }
}
```

```

}

class ChainFraction {
    private int n;
    private int x;
    private int[] a;

    public ChainFraction(int n, int x, int[] a) {
        this.n = n;
        this.x = x;
        this.a = a;
    }

    public ChainFraction(double value, int n, int x) {
        this.n = n;
        this.x = x;
        this.a = calculateA(value, n, x);
    }

    private int[] calculateA(double value, int n, int x) {
        this.a = new int[n];
        a[0] = (int) value;
        for (int i = 1; i < n; i++) {
            value = x / (value - a[i-1]);
            a[i] = (int) value;
        }
        return a;
    }

    public ChainFraction add(ChainFraction cf) {
        this.a = calculateA(this.value() + cf.value(), this.n, this.x);
        return null;
    }

    public ChainFraction subtract(ChainFraction cf) {
        this.a = calculateA(this.value() - cf.value(), this.n, this.x);
        return null;
    }

    public ChainFraction multiply(ChainFraction cf) {
        this.a = calculateA(this.value() * cf.value(), this.n, this.x);
        return null;
    }

    public ChainFraction divide(ChainFraction cf) {
        this.a = calculateA(this.value() / cf.value(), this.n, this.x);
        return null;
    }

    public double value() {
        double res = a[n-1];
        for (int i = n-2; i >= 0; i--) {
            res = a[i] + x / res;
        }
    }
}

```

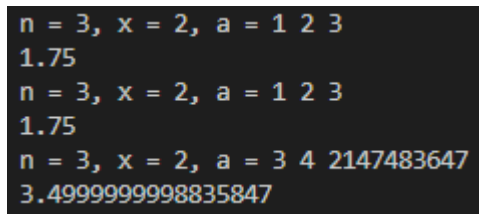
```

    }
    return res;
}

public void print() {
    System.out.printf("n = %d, x = %d, a = ", this.n, this.x);
    for (int i = 0; i < this.n; i++) {
        System.out.printf("%d ", this.a[i]);
    }
    System.out.println();
}
}

```

Пример результата работы программы показан на рисунке 2.



```

n = 3, x = 2, a = 1 2 3
1.75
n = 3, x = 2, a = 1 2 3
1.75
n = 3, x = 2, a = 3 4 2147483647
3.49999999998835847

```

Рисунок 2 – Пример результата работы программы

**Задание 3:** Создать класс Bookk: 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; в) список книг, выпущенных после заданного года. Определить конструкторы и методы setType(), getType(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

Листинг 3 – Задание 3

```

package l3;

public class e3 {

    /**
     * Вариант 2. Задача 5.
     * Создать класс
     * Bookk: 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;
     * в) список книг, выпущенных после заданного года.
     * Определить конструкторы и методы setType(), getType(), toString().
     */
}

```

```

* Определить дополнительно методы в классе, создающем массив объектов.
* Задать критерий выбора данных и вывести эти данные на консоль.
*/

public static void main(String[] args) {
    // Define an array of 5 books and fill it with different data (differe authors,
publishers, years, etc.)
    Bookk[] books = new Bookk[5];
    books[0] = new Bookk(1, "Bookk1", new String[] {"Author1", "Author2"}, "Publisher1",
2000, 100, 10.0, "Hardcover");
    books[1] = new Bookk(2, "Bookk2", new String[] {"Author1", "Author3"}, "Publisher2",
2001, 200, 20.0, "Hardcover");
    books[2] = new Bookk(3, "Bookk3", new String[] {"Author2", "Author3"}, "Publisher3",
2002, 300, 30.0, "Hardcover");
    books[3] = new Bookk(4, "Bookk4", new String[] {"Author1", "Author2"}, "Publisher4",
2003, 400, 40.0, "Hardcover");
    books[4] = new Bookk(5, "Bookk5", new String[] {"Author1", "Author3"}, "Publisher5",
2004, 500, 50.0, "Hardcover");

    // Print all books
    System.out.println("All books:");
    for (Bookk book : books) {
        System.out.println(book);
    }

    // Print books by author
    System.out.println("Bookks by Author1:");
    for (Bookk book : books) {
        for (String author : book.getAuthors()) {
            if (author.equals("Author1")) {
                System.out.println(book);
            }
        }
    }

    // Print books by publisher
    System.out.println("Bookks by Publisher1:");
    for (Bookk book : books) {
        if (book.getPublisher().equals("Publisher1")) {
            System.out.println(book);
        }
    }

    // Print books after year
    System.out.println("Bookks after 2002:");
    for (Bookk book : books) {
        if (book.getYear() > 2002) {
            System.out.println(book);
        }
    }
}
}

```

```

class Bookk {
    private int id;
    private String name;
    private String[] authors;
    private String publisher;
    private int year;
    private int pages;
    private double price;
    private String binding;

    public Bookk(int id, String name, String[] authors, String publisher, int year, int pages,
double price, String binding) {
        this.id = id;
        this.name = name;
        this.authors = authors;
        this.publisher = publisher;
        this.year = year;
        this.pages = pages;
        this.price = price;
        this.binding = binding;
    }

    // method to create array of books with random data and random authors
    public static Bookk[] createRandomBookks(int count) {
        Bookk[] books = new Bookk[count];
        for (int i = 0; i < count; i++) {
            books[i] = new Bookk(i, "Bookk" + i, new String[] {"Author" + (int) (Math.random()
* 10), "Author" + (int) (Math.random() * 10)}, "Publisher" + (int) (Math.random() * 10), (int)
(Math.random() * 100), (int) (Math.random() * 1000), Math.random() * 100, "Hardcover");
        }
        return books;
    }

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

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

```



```

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

public String getPublisher() {
    return publisher;
}

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

public int getYear() {
    return year;
}

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

public int getPages() {
    return pages;
}

public void setPages(int pages) {
    this.pages = pages;
}

public double getPrice() {
    return price;
}

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

public String getBinding() {
    return binding;
}

public void setBinding(String binding) {
    this.binding = binding;
}

// Override toString() method to print the object
@Override
public String toString() {
    String authors = "";
    // join all authors in one string without last comma
    for (int i = 0; i < this.authors.length; i++) {
        if (i == this.authors.length - 1) {
            authors += this.authors[i];
        } else {

```

```

        authors += this.authors[i] + ", ";
    }
}

return name + " by " + authors + " published by " + publisher + " in " + year + " with
" + pages + " pages, " + price + " rub. and " + binding + " binding.";
}
}
}

```

Пример результата работы программы показан на рисунке 3.

```

All books:
Bookk1 by Author1, Author2 published by Publisher1 in 2000 with 100 pages, 10.0 rub. and Hardcover binding.
Bookk2 by Author1, Author3 published by Publisher2 in 2001 with 200 pages, 20.0 rub. and Hardcover binding.
Bookk3 by Author2, Author3 published by Publisher3 in 2002 with 300 pages, 30.0 rub. and Hardcover binding.
Bookk4 by Author1, Author2 published by Publisher4 in 2003 with 400 pages, 40.0 rub. and Hardcover binding.
Bookk5 by Author1, Author3 published by Publisher5 in 2004 with 500 pages, 50.0 rub. and Hardcover binding.
Books by Author1:
Bookk1 by Author1, Author2 published by Publisher1 in 2000 with 100 pages, 10.0 rub. and Hardcover binding.
Bookk2 by Author1, Author3 published by Publisher2 in 2001 with 200 pages, 20.0 rub. and Hardcover binding.
Bookk4 by Author1, Author2 published by Publisher4 in 2003 with 400 pages, 40.0 rub. and Hardcover binding.
Bookk5 by Author1, Author3 published by Publisher5 in 2004 with 500 pages, 50.0 rub. and Hardcover binding.
Books by Publisher1:
Bookk1 by Author1, Author2 published by Publisher1 in 2000 with 100 pages, 10.0 rub. and Hardcover binding.
Books after 2002:
Bookk4 by Author1, Author2 published by Publisher4 in 2003 with 400 pages, 40.0 rub. and Hardcover binding.
Bookk5 by Author1, Author3 published by Publisher5 in 2004 with 500 pages, 50.0 rub. and Hardcover binding.

```

Рисунок 3 – Пример результата работы программы

**Задание 4:** Создать класс 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. Определить конструкторы и методы setType(), getType(), toString(). Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

Листинг 4 – Задание 4

```

package l3;

public class e4 {

    /**
     * Вариант 2. Задача 6.
     * Создать класс
     * 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.
* 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.
*/

public static void main(String[] args) {
    // Create array of houses with random data
    House[] houses = House.createHouses(20);

    // Print all houses with 3 rooms
    System.out.println("All houses with 3 rooms:");
    for (House house : houses) {
        if (house.getRooms() == 3) {
            System.out.println(house);
        }
    }

    // Print all houses with 3 rooms and floor between 10 and 20
    System.out.println("All houses with 3 rooms and floor between 10 and 20:");
    for (House house : houses) {
        if (house.getRooms() == 3 & house.getFloor() >= 10 & house.getFloor() <= 20) {
            System.out.println(house);
        }
    }

    // Print all houses with area more than 50
    System.out.println("All houses with area more than 50:");
    for (House house : houses) {
        if (house.getArea() > 90) {
            System.out.println(house);
        }
    }
}

class House {
    private int id;
    private int apartmentNumber;
    private double area;
    private int floor;
    private int rooms;
    private String street;
    private String buildingType;
    private int exploitationTerm;

    public House(int id, int apartmentNumber, double area, int floor, int rooms, String street,

```

```

String buildingType, int exploitationTerm) {
    this.id = id;
    this.apartmentNumber = apartmentNumber;
    this.area = area;
    this.floor = floor;
    this.rooms = rooms;
    this.street = street;
    this.buildingType = buildingType;
    this.exploitationTerm = exploitationTerm;
}

// Create array of houses with random data
public static House[] createHouses(int count) {
    House[] houses = new House[count];
    for (int i = 0; i < count; i++) {
        houses[i] = new House(i, (int) (Math.random() * 100), Math.random() * 100, (int)
(Math.random() * 100), (int) (Math.random() * 10), "Street " + (int) (Math.random() * 100),
"Type " + (int) (Math.random() * 10), (int) (Math.random() * 100));
    }
    return houses;
}

// Getters and setters
public int getId() {
    return id;
}

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

public int getApartmentNumber() {
    return apartmentNumber;
}

public void setApartmentNumber(int apartmentNumber) {
    this.apartmentNumber = apartmentNumber;
}

public double getArea() {
    return area;
}

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

public int getFloor() {
    return floor;
}

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

```

```

    }

    public int getRooms() {
        return rooms;
    }

    public void setRooms(int rooms) {
        this.rooms = rooms;
    }

    public String getStreet() {
        return street;
    }

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

    public String getBuildingType() {
        return buildingType;
    }

    public void setBuildingType(String buildingType) {
        this.buildingType = buildingType;
    }

    public int getExploitationTerm() {
        return exploitationTerm;
    }

    public void setExploitationTerm(int exploitationTerm) {
        this.exploitationTerm = exploitationTerm;
    }

    @Override
    public String toString() {
        return "House [id=" + id + ", apartmentNumber=" + apartmentNumber + ", area=" + area +
            ", floor=" + floor
                + ", rooms=" + rooms + ", street=" + street + ", buildingType=" + buildingType
            + ", exploitationTerm="
                + exploitationTerm + "]\n";
    }
}

```

Пример результата работы программы показан на рисунке 4.

```

All houses with 3 rooms:
House [id=1, apartmentNumber=92, area=47.219861887690286, floor=27, rooms=3, street=Street 73, buildingType=Type 1, exploitationTerm=22]
All houses with 3 rooms and floor between 10 and 20:
All houses with area more than 50:
House [id=3, apartmentNumber=37, area=98.10801093810481, floor=84, rooms=7, street=Street 52, buildingType=Type 4, exploitationTerm=34]
House [id=6, apartmentNumber=27, area=93.23694557261756, floor=47, rooms=7, street=Street 45, buildingType=Type 0, exploitationTerm=61]
House [id=13, apartmentNumber=46, area=97.63721046147525, floor=96, rooms=9, street=Street 2, buildingType=Type 8, exploitationTerm=40]
House [id=16, apartmentNumber=12, area=97.5149068098847, floor=38, rooms=2, street=Street 9, buildingType=Type 6, exploitationTerm=3]
House [id=18, apartmentNumber=26, area=93.37529850540713, floor=66, rooms=6, street=Street 54, buildingType=Type 0, exploitationTerm=21]

```

## Рисунок 4 – Пример результата работы программы

**Задание 5:** Создать объект класса Роза, используя классы Лепесток, Бутон. Методы: расцвести, завясть, вывести на консоль цвет бутона. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

### Листинг 5 – Задание 5

```
package l3;

import java.util.Arrays;

public class e5 {

    /**
     * Вариант 3. Задача 5.
     * Создать объект класса Роза, используя классы Лепесток, Бутон.
     * Методы: расцвести, завясть, вывести на консоль цвет бутона.
     * Аргументировать принадлежность классу каждого создаваемого метода
     * и корректно переопределить для каждого класса методы equals(), hashCode(), toString().
     */

    public static void main(String[] args) {
        // Example of using Rose class
        Rose rose = new Rose("red", 5);
        System.out.println(rose);
        System.out.println();

        rose.bloom();
        System.out.println(rose);
        System.out.println();
        rose.wither();
        System.out.println(rose);
        System.out.println();

        // Example of roses comparison
        Rose rose1 = new Rose("red", 5);
        Rose rose2 = new Rose("red", 5);
        System.out.println(rose1.equals(rose2));
    }
}

class Petal {
    private boolean isWithered;

    public Petal(boolean isWithered) {
        this.isWithered = isWithered;
    }

    public boolean isWithered() {
```

```

        return isWithered;
    }

    public void setWithered(boolean withered) {
        isWithered = withered;
    }

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

    @Override
    public int hashCode() {
        return (isWithered ? 1 : 0);
    }

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

class Bud {
    private String color;
    private int petalCount;
    private Petal[] petals;

    public Bud(String color, int petalCount) {
        this.color = color;
        this.petalCount = petalCount;
        this.petals = new Petal[petalCount];
        for (int i = 0; i < petalCount; i++) {
            petals[i] = new Petal(false);
        }
    }

    public String getColor() {
        return color;
    }

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

    public int getPetalCount() {
        return petalCount;
    }
}

```

```

    public void setPetalCount(int petalCount) {
        this.petalCount = petalCount;
    }

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

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

    public void bloom() {
        for (Petal petal : petals) {
            petal.setWithered(false);
        }
    }

    public void wither() {
        for (Petal petal : petals) {
            petal.setWithered(true);
        }
    }

    @Override
    public String toString() {
        return "Bud{" +
            "color='" + color + '\'' +
            ", petalCount=" + petalCount +
            ", petals=" + Arrays.toString(petals) +
            '}';
    }

    @Override
    public int hashCode() {
        int result = color.hashCode();
        result = 31 * result + petalCount;
        result = 31 * result + Arrays.hashCode(petals);
        return result;
    }

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

class Rose {
    private String color;

```



```

private int budCount;
private Bud[] buds;

public Rose(String color, int budCount) {
    this.color = color;
    this.budCount = budCount;
    this.buds = new Bud[budCount];
    for (int i = 0; i < budCount; i++) {
        buds[i] = new Bud(color, 5);
    }
}

public String getColor() {
    return color;
}

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

public int getBudCount() {
    return budCount;
}

public void setBudCount(int budCount) {
    this.budCount = budCount;
}

public Bud[] getBuds() {
    return buds;
}

public void setBuds(Bud[] buds) {
    this.buds = buds;
}

public void bloom() {
    for (Bud bud : buds) {
        bud.bloom();
    }
}

public void wither() {
    for (Bud bud : buds) {
        bud.wither();
    }
}

@Override
public String toString() {
    return "Rose{" +
        "color='" + color + '\'' +
        ", budCount=" + budCount +

```

```

        ", buds=" + Arrays.toString(buds) +
        '}';
    }

    @Override
    public int hashCode() {
        int result = color.hashCode();
        result = 31 * result + budCount;
        result = 31 * result + Arrays.hashCode(buds);
        return result;
    }

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

```

Пример результата работы программы показан на рисунке 5.

[illegible]

Рисунок 5 – Пример результата работы программы

**Задание 6:** Создать объект класса Дерево, используя классы Лист. Методы: зацвести, опать листьям, покрыться инеем, пожелтеть листьям. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString()).

### Листинг 6 – Задание 6

```
package l3;

import java.util.Arrays;

public class e6 {

    /**
     * Вариант 3. Задача 6.
     * Создать объект класса Дерево, используя классы Лист.
     * Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.
     */
}
```

```

* Аргументировать принадлежность классу каждого создаваемого метода
* и корректно переопределить для каждого класса методы equals(), hashCode(), toString().
*/

public static void main(String[] args) {
    // Example of using Tree class with array of leaves using Tree constructor
    Leaf[] leaves = new Leaf[5];
    for (int i = 0; i < leaves.length; i++) {
        leaves[i] = new Leaf(false, false, false, false);
    }
    Tree tree = new Tree(leaves);
    System.out.println(tree);
    System.out.println();
    // Example of using Tree methods
    tree.bloom();
    System.out.println(tree);
    System.out.println();
    tree.fall();
    System.out.println(tree);
    System.out.println();
    tree.coverWithIce();
    System.out.println(tree);
    System.out.println();
    tree.yellow();
    System.out.println(tree);
    System.out.println();

    // Example of trees comparison
    Tree tree1 = new Tree(leaves);
    Tree tree2 = new Tree(leaves);
    System.out.println(tree1.equals(tree2));

}

}

class Leaf {
    private boolean isWithered;
    private boolean isFallen;
    private boolean isYellowed;
    private boolean isCoveredWithIce;

    public Leaf(boolean isWithered, boolean isFallen, boolean isYellowed, boolean
isCoveredWithIce) {
        this.isWithered = isWithered;
        this.isFallen = isFallen;
        this.isYellowed = isYellowed;
        this.isCoveredWithIce = isCoveredWithIce;
    }

    public boolean isWithered() {
        return isWithered;
    }
}

```

```

public void setWithered(boolean withered) {
    isWithered = withered;
}

public boolean isFallen() {
    return isFallen;
}

public void setFallen(boolean fallen) {
    isFallen = fallen;
}

public boolean isYellowed() {
    return isYellowed;
}

public void setYellowed(boolean yellowed) {
    isYellowed = yellowed;
}

public boolean isCoveredWithIce() {
    return isCoveredWithIce;
}

public void setCoveredWithIce(boolean coveredWithIce) {
    isCoveredWithIce = coveredWithIce;
}

@Override
public String toString() {
    return "Leaf{" +
        "isWithered=" + isWithered +
        ", isFallen=" + isFallen +
        ", isYellowed=" + isYellowed +
        ", isCoveredWithIce=" + isCoveredWithIce +
        '}';
}

@Override
public int hashCode() {
    int result = (isWithered ? 1 : 0);
    result = 31 * result + (isFallen ? 1 : 0);
    result = 31 * result + (isYellowed ? 1 : 0);
    result = 31 * result + (isCoveredWithIce ? 1 : 0);
    return result;
}

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

    Leaf leaf = (Leaf) o;

```

```

        return hashCode() == leaf.hashCode();
    }
}

class Tree {
    private Leaf[] leaves;

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

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

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

    public void bloom() {
        for (Leaf leaf : leaves) {
            leaf.setWithered(false);
        }
    }

    public void fall() {
        for (Leaf leaf : leaves) {
            leaf.setFallen(true);
        }
    }

    public void coverWithIce() {
        for (Leaf leaf : leaves) {
            leaf.setCoveredWithIce(true);
        }
    }

    public void yellow() {
        for (Leaf leaf : leaves) {
            leaf.setYellowed(true);
        }
    }

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

    @Override
    public int hashCode() {

```

```

        return Arrays.hashCode(leaves);
    }

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

        Tree tree = (Tree) o;

        return hashCode() == tree.hashCode();
    }
}

```

Пример результата работы программы показан на рисунке 6.

```

Tree{leaves=[Leaf{isWithered=false, isFallen=false, isYellowed=false, isCoveredWithIce=false}, Leaf{isWithered=false, isFallen=false, isYellowed=false, isCoveredWithIce=false}, Leaf{isWithered=false, isFallen=false, isYellowed=false, isCoveredWithIce=false}]
Tree{leaves=[Leaf{isWithered=false, isFallen=false, isYellowed=false, isCoveredWithIce=false}, Leaf{isWithered=false, isFallen=false, isYellowed=false, isCoveredWithIce=false}, Leaf{isWithered=false, isFallen=false, isYellowed=false, isCoveredWithIce=false}]
Tree{leaves=[Leaf{isWithered=false, isFallen=true, isYellowed=false, isCoveredWithIce=false}, Leaf{isWithered=false, isFallen=true, isYellowed=false, isCoveredWithIce=false}, Leaf{isWithered=false, isFallen=true, isYellowed=false, isCoveredWithIce=false}]
Tree{leaves=[Leaf{isWithered=false, isFallen=true, isYellowed=false, isCoveredWithIce=true}, Leaf{isWithered=false, isFallen=true, isYellowed=false, isCoveredWithIce=true}, Leaf{isWithered=false, isFallen=true, isYellowed=false, isCoveredWithIce=true}]
Tree{leaves=[Leaf{isWithered=false, isFallen=true, isYellowed=true, isCoveredWithIce=true}, Leaf{isWithered=false, isFallen=true, isYellowed=true, isCoveredWithIce=true}, Leaf{isWithered=false, isFallen=true, isYellowed=true, isCoveredWithIce=true}]
true

```

Рисунок 6 – Пример результата работы программы

**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»..

Листинг 7 – Задание 7

```

package l3;

import java.util.ArrayList;

public class e7 {

    /**
     * Вариант 4. Задача 5.
     * 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».
     */
}

```

```

    */

    public static void main(String[] args) {
        Book book1 = new Book("Java Programming");
        Book book2 = new Book("Database Management");
        Reader reader1 = new Reader("Alice");
        Reader reader2 = new Reader("Bob");
        Librarian librarian = new Librarian();
        Administrator administrator = new Administrator();

        // Test issuing a book for subscription
        librarian.issueBook(book1, reader1, true);

        // Test issuing a book for reading room
        librarian.issueBook(book2, reader2, false);

        // Test returning a book
        librarian.returnBook(book2, reader2);

        // Test blacklisting a reader
        administrator.blacklistReader(reader1);

        // Test preventing a blacklisted reader from issuing a book
        if (!administrator.isReaderBlacklisted(reader1)) {
            librarian.issueBook(book1, reader1, false);
        } else {
            System.out.println("[MAIN] " + reader1.getName() + " is blacklisted");
        }

        // Test returning a book to the library from blacklisted reader
        librarian.returnBook(book1, reader1);
    }
}

class Reader {
    private String name;
    private ArrayList<Order> orders;

    public Reader(String name) {
        this.name = name;
        this.orders = new ArrayList<>();
    }

    public void placeOrder(Book book, boolean forSubscription) {
        Order order = new Order(book, forSubscription);
        orders.add(order);
        System.out.println("[REDR] Order placed by " + name + " for book " + book.getTitle() +
            " (" + (forSubscription ? "subscription" : "reading room") + ")");
    }

    public void returnBook(Book book) {
        for (Order order : orders) {

```

```

        if (order.getBook() == book) {
            orders.remove(order);
            System.out.println("[REDR] "+ name + " returned book " + book.getTitle());
            break;
        }
    }

    public String getName() {
        return name;
    }

    public ArrayList<Order> getOrders() {
        return orders;
    }
}

class Book {
    private String title;
    private boolean available;

    public Book(String title) {
        this.title = title;
        this.available = true;
    }

    public void issue(boolean forSubscription) {
        if (available) {
            available = false;
            System.out.println("[BOOK] Book '" + title + "' issued " + (forSubscription ? "for
subscription" : "for reading room"));
        } else {
            System.out.println("[BOOK] Book '" + title + "' not available");
        }
    }

    public void returnBook() {
        available = true;
        System.out.println("[BOOK] Book '" + title + "' returned");
    }

    public String getTitle() {
        return title;
    }

    public boolean isAvailable() {
        return available;
    }
}

class Order {
    private Book book;
    private boolean forSubscription;

```



```

    public Order(Book book, boolean forSubscription) {
        this.book = book;
        this.forSubscription = forSubscription;
    }

    public Book getBook() {
        return book;
    }

    public boolean isForSubscription() {
        return forSubscription;
    }
}

class Librarian {
    public void issueBook(Book book, Reader reader, boolean forSubscription) {
        book.issue(forSubscription);
        reader.placeOrder(book, forSubscription);
    }

    public void returnBook(Book book, Reader reader) {
        book.returnBook();
        reader.returnBook(book);
    }
}

class Administrator {
    private ArrayList<Reader> blacklistedReaders;

    public Administrator() {
        this.blacklistedReaders = new ArrayList<>();
    }

    public void blacklistReader(Reader reader) {
        blacklistedReaders.add(reader);
        System.out.println("[ADMN] " + reader.getName() + " blacklisted");
    }

    public boolean isReaderBlacklisted(Reader reader) {
        return blacklistedReaders.contains(reader);
    }
}

```

Пример результата работы программы показан на рисунке 7.

```

[BOOK] Book 'Java Programming' issued for subscription
[REDR] Order placed by Alice for book Java Programming (subscription)
[BOOK] Book 'Database Management' issued for reading room
[REDR] Order placed by Bob for book Database Management (reading room)
[BOOK] Book 'Database Management' returned
[REDR] Bob returned book Database Management
[ADMIN] Alice blacklisted
[MAIN] Alice is blacklisted
[BOOK] Book 'Java Programming' returned
[REDR] Alice returned book Java Programming

```

Рисунок 7 – Пример результата работы программы

**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..

Листинг 8 – Задание 8

```

package l3;

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

public class e8 {

    /**
     * Вариант 4. Задача 6.
     * 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.
     */

    public static void main(String[] args) {
        // Заказчик и Конструктор
        Customer customer = new Customer("John");
        Constructor constructor = new Constructor("ABC Construction");

        // Техническое задание на проектирование многоэтажного дома
        TermsOfReference tor = new TermsOfReference("Multi-storey building", 10, 1000);

        // Конструктор регистрирует ТЗ
        constructor.registerTOR(tor);

        // Конструктор определяет стоимость проектирования и строительства, выставляет
        // Заказчику Счет за проектирование
        Invoice invoice = constructor.issueInvoice(customer, constructor.calculateCost(tor));
    }
}

```

```

        // Конструктор создает Бригаду Конструкторов для выполнения Проекта
        List<Constructor> brigade = constructor.createBrigade(tor);

        // Заказчик, Конструктор, Техническое задание, Счет
        System.out.println(customer);
        System.out.println(constructor);
        System.out.println(tor);
        System.out.println(invoice);

        // Бригада Конструкторов
        for (Constructor member : brigade) {
            System.out.println(member);
        }
    }
}

class Customer {
    private final String name;

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

    public String getName() {
        return name;
    }

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

class Constructor {
    private final String name;
    private final List<TermsOfReference> registeredTORs;

    public Constructor(String name) {
        this.name = name;
        this.registeredTORs = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public void registerTOR(TermsOfReference tor) {
        registeredTORs.add(tor);
    }
}

```

```

public int calculateCost(TermsOfReference tor) {
    // Конструктор определяет стоимость проектирования и строительства
    return tor.getNumberOfFloors() * tor.getArea() * 100;
}

public Invoice issueInvoice(Customer customer, int amount) {
    // Конструктор выставляет Заказчику Счет за проектирование
    return new Invoice(customer, this, amount);
}

public List<Constructor> createBrigade(TermsOfReference tor) {
    // Create a brigade of constructors to carry out the project
    // Конструктор создает Бригаду Конструкторов для выполнения Проекта
    List<Constructor> brigade = new ArrayList<>();
    for (int i = 0; i < tor.getNumberOfFloors(); i++) {
        brigade.add(new Constructor(name + " " + (i+1)));
    }
    return brigade;
}

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

}

class TermsOfReference {
    private final String name;
    private final int numberOfFloors;
    private final int area;

    public TermsOfReference(String name, int numberOfFloors, int area) {
        this.name = name;
        this.numberOfFloors = numberOfFloors;
        this.area = area;
    }

    public String getName() {
        return name;
    }

    public int getNumberOfFloors() {
        return numberOfFloors;
    }

    public int getArea() {
        return area;
    }

    @Override
    public String toString() {

```

```

        return "TermsOfReference{" +
            "name='" + name + '\'' +
            ", numberOfFloors=" + numberOfFloors +
            ", area=" + area +
            '}';
    }
}

class Invoice {
    private final Customer customer;
    private final Constructor constructor;
    private final int amount;

    public Invoice(Customer customer, Constructor constructor, int amount) {
        this.customer = customer;
        this.constructor = constructor;
        this.amount = amount;
    }

    public Customer getCustomer() {
        return customer;
    }

    public Constructor getConstructor() {
        return constructor;
    }

    public int getAmount() {
        return amount;
    }

    @Override
    public String toString() {
        return "Invoice{" +
            "customer=" + customer.getName() +
            ", constructor=" + constructor.getName() +
            ", amount=" + amount +
            '}';
    }
}

```

Пример результата работы программы показан на рисунке 8.

```
Customer{name='John'}
Constructor{name='ABC Construction'}
TermsOfReference{name='Multi-storey building', numberOfFloors=10, area=1000}
Invoice{customer=John, constructor=ABC Construction, amount=1000000}
Constructor{name='ABC Construction 1'}
Constructor{name='ABC Construction 2'}
Constructor{name='ABC Construction 3'}
Constructor{name='ABC Construction 4'}
Constructor{name='ABC Construction 5'}
Constructor{name='ABC Construction 6'}
Constructor{name='ABC Construction 7'}
Constructor{name='ABC Construction 8'}
Constructor{name='ABC Construction 9'}
Constructor{name='ABC Construction 10'}
```

Рисунок 8 – Пример результата работы программы

**Вывод:** В данной лабораторной работе мы рассмотрели множество тем, связанных с объектно-ориентированным программированием на Java, включая классы, наследование и полиморфизм. Мы реализовали различные классы, в том числе Matrix, Chained Fraction, Book, House, Rose и Tree, и использовали различные методы для обработки и вывода данных. Мы также построили модели двух программных комплексов, библиотечной системы и системного конструкторского бюро. Эта лабораторная работа обеспечила всестороннее понимание концепций объектно-ориентированного программирования на Java и практический опыт их реализации.