



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 Интеллектуальные системы анализа,
обработки и интерпретации больших данных

ОТЧЕТ

по лабораторной работе № 3

Вариант 13

Название: Классы, наследование и полиморфизм

Дисциплина: Языки программирования для работы с большими данными

Студент

ИУ6-22М

(Группа)

(Подпись, дата)

В.А.Ловцов

(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 Вектор в R^3 . Реализовать методы для проверки векторов на ортогональность, проверки пересечения не ортогональных векторов, сравнения векторов. Создать массив из m объектов. Определить, какие из векторов компланарны.

Код:

```
import java.util.Scanner;

class Point {
    double x, y, z;

    public Point(double x, double y, double z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }

    public void printPoint() {
        System.out.print("(" + this.x + ", " + this.y + ", " +
this.z + ")", ");
    }
}

class PointReader {
    public Point readPoint(Scanner scanner){
//        System.out.println("");
        System.out.print("\tx=");
        double x = scanner.nextDouble();
        System.out.print("\ty=");
        double y = scanner.nextDouble();
        System.out.print("\tz=");
        double z = scanner.nextDouble();

        return new Point(x, y, z);
    }
}

class Vector {
    Point a, b;

    public Vector(Point a, Point b) {
        this.a = a;
```

```

        this.b = b;
    }

    double getLen() {
        return Math.sqrt(this.getLenX() * this.getLenX() +
            this.getLenY() * this.getLenY() +
            this.getLenZ() * this.getLenZ());
    }

    public double getLenX() {
        return this.b.x - this.a.x;
    }

    public double getLenY() {
        return this.b.y - this.a.y;
    }

    public double getLenZ() {
        return this.b.z - this.a.z;
    }

    public double getScalar(Vector other) {
        return (this.getLenX() * other.getLenX() +
            this.getLenY() * other.getLenY() +
            this.getLenZ() * other.getLenZ());
    }

    public double getSin(Vector other) {
        return this.getScalar(other) / (this.getLen() *
other.getLen());
    }

    public double getVectMul(Vector other) {
        return (this.getLen() * other.getLen()) /
this.getSin(other);
    }

    public Vector getVectorVector(Vector other) {
        double tmpXb = this.getLenY() * other.getLenZ();
        double tmpXa = this.getLenZ() * other.getLenY();

        double tmpYb = this.getLenZ() * other.getLenX();
        double tmpYa = this.getLenX() * other.getLenZ();

        double tmpZb = this.getLenX() * other.getLenY();

```

```

        double tmpZa = this.getLenY() * other.getLenX();

        Point tmpB = new Point(tmpXb, tmpYb, tmpZb);
        Point tmpA = new Point(tmpXa, tmpYa, tmpZa);

        return new Vector(tmpA, tmpB);
    }

    public double getMixMul(Vector other1, Vector other2) {
        Vector tmpBC = other1.getVectorVector(other2);
        return this.getScalar(tmpBC);
    }

    public boolean isComplanar(Vector other1, Vector other2) {
        return (this.getMixMul(other1, other2) == 0);
    }

    public boolean isCross(Vector other) {
        double v1 = other.getVectMul(new Vector(other.a,
this.a));
        double v2 = other.getVectMul(new Vector(other.a,
this.b));

        double v3 = this.getVectMul(new Vector(this.a,
other.a));
        double v4 = this.getVectMul(new Vector(this.a,
other.b));

        return (v1 * v2 < 0 & v3 * v4 < 0);
    }

    public boolean isOrt(Vector other) {
        return (this.getScalar(other) == 0);
    }

    public void printVector() {
        this.a.printPoint();
        this.b.printPoint();
        System.out.println();
    }
}

public class Third_1 {
    public static void main(String[] args) {

```

```

Scanner scanner = new Scanner(System.in);
System.out.print("Введите n: ");
int n = scanner.nextInt();

scanner.nextLine();
Vector[] array = new Vector[n];
PointReader reader = new PointReader();
for (int i = 0; i < n; i++){
    System.out.println(i + 1 + " ");

    System.out.println( "Точка A:");
    Point tmpA = reader.readPoint(scanner);
    System.out.println( "Точка B:");
    Point tmpB = reader.readPoint(scanner);
    array[i] = new Vector(tmpA, tmpB);
}
for (int i = 0; i < n - 2; i++){
    for (int j = i + 1; j < n - 1; j++){
        for (int k = j + 1; k < n; k++) {
            if
                (array[i].isComplanar(array[j],
array[k])) {

                System.out.println("Компланарные:");
                array[i].printVector();
                array[j].printVector();
                array[k].printVector();

            }
        }
    }
}
}
}

```

Работа программы показана на рисунке 1.

```

Введите n: 4
1)
Точка A:
    x=1
    y=1
    z=2
Точка B:
    x=1, 1
    y=3
    z=2
2)
Точка A:
    x=2
    y=4
    z=3
Точка B:
    x=5
    y=7
    z=6
3)
Точка A:
    x=4
    y=6
    z=5
Точка B:
    x=7
    y=9
    z=8
4)
Точка A:
    x=5
    y=3
    z=6
Точка B:
    x=4
    y=2
    z=5
Компланарные:
(1.0, 1.0, 2.0), (1.1, 3.0, 2.0),
(2.0, 4.0, 3.0), (5.0, 7.0, 6.0),
(4.0, 6.0, 5.0), (7.0, 9.0, 8.0),
Компланарные:
(1.0, 1.0, 2.0), (1.1, 3.0, 2.0),
(2.0, 4.0, 3.0), (5.0, 7.0, 6.0),
(5.0, 3.0, 6.0), (4.0, 2.0, 5.0),
Компланарные:
(1.0, 1.0, 2.0), (1.1, 3.0, 2.0),
(4.0, 6.0, 5.0), (7.0, 9.0, 8.0),
(5.0, 3.0, 6.0), (4.0, 2.0, 5.0),
Компланарные:
(2.0, 4.0, 3.0), (5.0, 7.0, 6.0),
(4.0, 6.0, 5.0), (7.0, 9.0, 8.0),
(5.0, 3.0, 6.0), (4.0, 2.0, 5.0),

```

Рисунок 1 – Работа программы

Задание 2: определить класс Матрица размерности (n x n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения матриц. Объявить массив объектов.

$$\|a\|_1 = \max_{1 \leq i \leq n} \sum_{j=1}^n (a_{ij}), \|a\|_2 = \max_{1 \leq j \leq n} \sum_{i=1}^n (a_{ij})$$

Создать методы, вычисляющие первую и вторую нормы матрицы

Определить, какая из матриц имеет наименьшую первую и вторую нормы.

Код:

```
import java.util.Arrays;
import java.util.Scanner;

class Matrix {
    int[][] mat;
    int n;
    public Matrix(int n) {
        this.n = n;
        mat = new int[n][n];
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n ; j++) {
                mat[i][j] = (int) (Math.random() * 10);
            }
    }

    public Matrix(int n, int val) {
        this.n = n;
        mat = new int[n][n];
        for (int i = 0; i < n; i++)
            for (int j = 0; j < n ; j++) {
                mat[i][j] = val;
            }
    }

    public Matrix(int[][] arr) {
        this.n = arr.length;
        mat = new int[this.n][this.n];
        for (int i = 0; i < n; i++)
            System.arraycopy(arr[i], 0, mat[i], 0, n);
    }

    public Matrix add(Matrix other) {
        int[][] summa = new int[this.n][this.n];

        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                summa[i][j] = this.mat[i][j] + other.mat[i][j];
            }
    }
}
```

```

        }
        return new Matrix(summa);
    }

    public Matrix sub(Matrix other) {
        int[][] sub = new int[this.n][this.n];

        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                sub[i][j] = this.mat[i][j] - other.mat[i][j];
            }
        return new Matrix(sub);
    }

    public Matrix mul(Matrix other) {
        int[][] mul = new int[this.n][this.n];

        for (int i = 0; i < n; i++)
            for (int j = 0; j < n; j++) {
                int line_sum = 0;
                for (int k = 0; k < n; k++){
                    line_sum      +=      this.mat[i][k]      *
other.mat[k][j];
                }
                mul[i][j] = line_sum;
            }
        return new Matrix(mul);
    }

    public int l1() {
        int max = -999999999;
        for (int i = 0; i < n; i++) {
            int cur_sum = 0;
            for (int j = 0; j < n; j++) {
                cur_sum += mat[i][j];
            }
            if (cur_sum > max)
                max = cur_sum;
        }
        return max;
    }

    public int l2() {
        int max = -999999999;
        for (int j = 0; j < n; j++) {
            int cur_sum = 0;
            for (int i = 0; i < n; i++) {
                cur_sum += mat[i][j];
            }
            if (cur_sum > max)
                max = cur_sum;
        }
    }

```



```

    }
    return max;
}

public void printMat() {
    for (int i = 0; i < this.n; i++)
        System.out.println(Arrays.toString(this.mat[i]));
    System.out.println();
}
}

public class Fourth_1 {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.print("Введите n: ");
        int n = scanner.nextInt();

        System.out.print("Введите m: ");
        int m = scanner.nextInt();

        scanner.nextLine();
        Matrix[] array = new Matrix[m];

        int l1_min = 999999999;
        int l2_min = 999999999;
        int l1_id = -1, l2_id = -1;

        for (int i = 0; i < m; i++) {
            array[i] = new Matrix(n);

            int l1 = array[i].l1();
            if (l1 < l1_min) {
                l1_min = l1;
                l1_id = i;
            }

            int l2 = array[i].l2();
            if (l2 < l2_min) {
                l2_min = l2;
                l2_id = i;
            }

            System.out.println("Matrix " + (i + 1));
            array[i].printMat();
        }

        System.out.println("Min l1: " + l1_min + ". It has matrix number " + (l1_id + 1));
        System.out.println("Min l2: " + l2_min + ". It has matrix number " + (l2_id + 1));
    }
}

```

```

Matrix add = array[0].add(array[1]);
System.out.println("Matrix 1 add 2");
add.printMat();
Matrix sub = array[0].sub(array[2]);
System.out.println("Matrix 1 subtract 3:");
sub.printMat();
Matrix mul = array[1].mul(array[2]);
System.out.println("Matrix 2 mul 3: ");
mul.printMat();
}
}

```

Работа программы показана на рисунке 2.

```

Matrix 1
[7, 9, 5, 3]
[1, 4, 8, 6]
[2, 0, 0, 3]
[1, 2, 1, 6]

Matrix 2
[0, 5, 1, 2]
[3, 9, 6, 1]
[2, 0, 4, 5]
[6, 4, 1, 8]

Matrix 3
[8, 8, 9, 2]
[6, 4, 9, 6]
[3, 2, 0, 6]
[1, 8, 8, 2]

Min l1: 19. It has matrix number 2
Min l1: 18. It has matrix number 1
Matrix 1 add 2
[7, 14, 6, 5]
[4, 13, 14, 7]
[4, 0, 4, 8]
[7, 6, 2, 14]

Matrix 1 subtract 3:
[-1, 1, -4, 1]
[-5, 0, -1, 0]
[-1, -2, 0, -3]
[0, -6, -7, 4]

Matrix 2 mul 3:
[35, 38, 61, 40]
[97, 80, 116, 98]
[33, 64, 58, 38]
[83, 130, 154, 58]

```

Рисунок 2 – Работа программы

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. `Patient`: `id`, `Фамилия`, `Имя`, `Отчество`, `Адрес`, `Телефон`, `Номер медицинской карты`, `Диагноз`. Создать массив объектов. Вывести: а) список пациентов, имеющих данный диагноз; б) список пациентов, номер медицинской карты у которых находится в заданном интервале.

Код:

```
import java.util.Objects;
import java.util.Scanner;
import java.util.concurrent.atomic.AtomicInteger;

class PatientList {
    public Patient[] createPatients(int n, Scanner scanner) {
        Patient[] listPat = new Patient[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Пациент " + i);
            System.out.print("\tФамилия: ");
            String surname = scanner.nextLine();

            System.out.print("\tИмя: ");
            String name = scanner.nextLine();

            System.out.print("\tОтчество: ");
            String father_name = scanner.nextLine();

            System.out.print("\tАдрес: ");
            String address = scanner.nextLine();

            System.out.print("\tТелефон: ");
            String phone = scanner.nextLine();

            System.out.print("\tНомер карты: ");
            int card = scanner.nextInt();
            scanner.nextLine();

            System.out.print("\tДиагноз: ");
            String diagnose = scanner.nextLine();

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

```

        listPat[i] = new Patient(surname, name,
father_name, address, phone, card, diagnose);

    }
    return listPat;
}

}

class Patient {
    private static final AtomicInteger count = new
AtomicInteger(0);
    int id;
    String surname;
    String name;
    String father_name;
    String address;
    String phone;
    int card;
    String diagnose;

    public Patient(
        String surname, String name, String fatherName,
String address, String phone, int card, String diagnose
    ) {
        this.id = count.incrementAndGet();
        this.surname = surname;
        this.name = name;
        this.father_name = fatherName;
        this.address = address;
        this.phone = phone;
        this.card = card;
        this.diagnose = diagnose;
    }

    public void print() {
        System.out.println("Пациент id: " + this.id );
        System.out.print("    |  Фамилия: " + this.surname );

        System.out.print("    |  Имя: " + this.name );

        System.out.print("    |  Отчество: " + this.father_name
);

        System.out.print("    |  Адрес: " + this.address );

        System.out.print("    |  Телефон: " + this.phone );

        System.out.print("    |  Номер карты: " + this.card );

        System.out.println("    |  Диагноз: " + this.diagnose );
    }
}

```

```

        System.out.println();
    }
}

public class Third_2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Введите n: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Введите искомый диагноз: ");
        String diag = scanner.nextLine();

        System.out.println("Введите интервал медицинских карт:");

        System.out.print("\tНачало: ");
        int start = scanner.nextInt();
        System.out.print("\tКонец: ");
        int stop = scanner.nextInt();
        scanner.nextLine();

        PatientList patientCreator = new PatientList();
        Patient[] patients = patientCreator.createPatients(n,
scanner);

        for (int i = 0; i < n; i++){
            patients[i].print();
        }

        System.out.println("Пациенты с диагнозом \"" + diag +
"\");");
        for (int i = 0; i < n; i++){
            if (Objects.equals(patients[i].diagnose, diag))
                patients[i].print();
        }

        System.out.println("Пациенты с номерами медкарт в
диапазоне " + start + " - " + stop);
        for (int i = 0; i < n; i++){
            if (patients[i].card >= start & patients[i].card <=
stop)
                patients[i].print();
        }
    }
}

```

Работа программы показана на рисунке 3.

```

Пациент id: 1
  | Фамилия: lsdn | Имя: sdLkfj | Отчество: sjf | Адрес: kwjnesdfkL | Телефон: 238543 | Номер карты: 50 | Диагноз: djdj

Пациент id: 2
  | Фамилия: jsLrhgdg | Имя: oijergf | Отчество: werk | Адрес: welFn | Телефон: 124 | Номер карты: 101 | Диагноз: xxx

Пациенты с диагнозом "xxx"
Пациент id: 2
  | Фамилия: jsLrhgdg | Имя: oijergf | Отчество: werk | Адрес: welFn | Телефон: 124 | Номер карты: 101 | Диагноз: xxx

Пациенты с номерами медкарт в диапазоне 10 - 100
Пациент id: 1
  | Фамилия: lsdn | Имя: sdLkfj | Отчество: sjf | Адрес: kwjnesdfkL | Телефон: 238543 | Номер карты: 50 | Диагноз: djdj

Process finished with exit code 0
|

```

Рисунок 3 – Работа программы

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. Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки. Создать массив объектов. Вывести: а) список абитуриентов, имеющих неудовлетворительные оценки; б) список абитуриентов, средний балл у которых выше заданного; в) выбрать заданное число n абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

Код:

```

import java.util.Objects;
import java.util.Scanner;
import java.util.concurrent.atomic.AtomicInteger;

class AbiturientList {
    public Abiturient[] createAbiturients(int n, Scanner scanner) {
        Abiturient[] listAb = new Abiturient[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Абитуриент " + i);
            System.out.print("\tФамилия: ");
            String surname = scanner.nextLine();

            System.out.print("\tИмя: ");
            String name = scanner.nextLine();

            System.out.print("\tОтчество: ");

```

```

        String father_name = scanner.nextLine();

        System.out.print("\tАдрес: ");
        String address = scanner.nextLine();

        System.out.print("\tТелефон: ");
        String phone = scanner.nextLine();

        System.out.print("\tОценка по математике: ");
        int math = scanner.nextInt();
        System.out.print("\tОценка по русскому: ");
        int russian = scanner.nextInt();
        System.out.print("\tОценка по информатике: ");
        int informatics = scanner.nextInt();
        scanner.nextLine();

        Marks mark = new Marks(math, russian, informatics);

        System.out.println();

        listAb[i] = new Abiturient(surname, name,
father_name, address, phone, mark);
    }
    return listAb;
}
}

class Marks {
    int math;
    int russian;
    int informatics;

    public Marks(int math, int russian, int informatics) {
        this.math = math;
        this.russian = russian;
        this.informatics = informatics;
    }

    public double getMean(){
        return (double) (math + russian + informatics) / 3;
    }

    public String toString() {
        return "(мат: " + this.math + "; рус: " + this.russian
+ "; инф: " + this.informatics + ")";
    }

    public boolean hasBad() {
        return (this.math < 3 | this.russian < 3 |
this.informatics < 3);
    }
}

```

```

    }
}

class Abiturient {
    private static final AtomicInteger count = new
AtomicInteger(0);
    int id;
    String surname;
    String name;
    String father_name;
    String address;
    String phone;
    Marks marks;

    public Abiturient(
        String surname, String name, String fatherName,
String address, String phone, Marks marks
    ) {
        this.id = count.incrementAndGet();
        this.surname = surname;
        this.name = name;
        this.father_name = fatherName;
        this.address = address;
        this.phone = phone;
        this.marks = marks;
    }

    public void print() {
        System.out.println("Абитуриент id: " + this.id );
        System.out.print("    |  Фамилия: " + this.surname );

        System.out.print("    |  Имя: " + this.name );

        System.out.print("    |  Отчество: " + this.father_name
);

        System.out.print("    |  Адрес: " + this.address );

        System.out.print("    |  Телефон: " + this.phone );

        System.out.print("    |  Оценки: " +
this.marks.toString());

        System.out.println();
    }
}

public class Fourth_2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Введите число абитуриентов: ");
    }
}

```



```

int m = scanner.nextInt();
scanner.nextLine();

System.out.print("Искомый средний балл: ");
double mean_mark = scanner.nextDouble();
System.out.print("Введите топ-абитуриентов (n): ");
int n = scanner.nextInt();
scanner.nextLine();

AbiturientList abiturientCreator = new
AbiturientList();
Abiturient[] abiturients =
abiturientCreator.createAbiturients(m, scanner);

for (int i = 0; i < m; i++){
    abiturients[i].print();
}

System.out.println("Абитуриенты с плохими оценками:");
for (int i = 0; i < m; i++){
    if (abiturients[i].marks.hasBad())
        abiturients[i].print();
}

System.out.println("Абитуриенты со средним баллом выше
" + mean_mark);
for (int i = 0; i < m; i++){
    if (abiturients[i].marks.getMean() > mean_mark)
        abiturients[i].print();
}

double[] mean_marks = new double[m];
for (int i = 0; i < m; i++){
    for (int j = i; j < m; j++) {
        if (abiturients[i].marks.getMean() <
abiturients[j].marks.getMean()){
            Abiturient tmp = abiturients[i];
            abiturients[i] = abiturients[j];
            abiturients[j] = tmp;
        }
    }
}

System.out.println("Список абитуриентов топ-" + n);
for (int i = 0; i < n; i++){
    abiturients[i].print();
}

System.out.println("Абитуриенты с полупроходным
баллом:");

```

```

        double half_entrance_mark = abiturients[n-
1].marks.getMean();
        for (int i = 0; i < m; i++){
            if (abiturients[i].marks.getMean() ==
half_entrance_mark)
                abiturients[i].print();
        }
    }
}

```

Работа программы показана на рисунке 4.

```

Абитуриент id: 1
| Фамилия: хех | Имя: хох | Отчество: хах | Адрес: ergh | Телефон: 239857 | Оценки: (мат: 9; рус: 8; инф: 7)
Абитуриент id: 2
| Фамилия: jkrhf | Имя: jdfg | Отчество: rdfg | Адрес: erng | Телефон: 345 | Оценки: (мат: 7; рус: 6; инф: 6)
Абитуриент id: 3
| Фамилия: hjfgd | Имя: ejrng | Отчество: geng | Адрес: edngr | Телефон: 345 | Оценки: (мат: 10; рус: 5; инф: 5)
Абитуриент id: 4
| Фамилия: dfljh | Имя: dlnsgf | Отчество: lan | Адрес: slnf | Телефон: 2104393 | Оценки: (мат: 4; рус: 4; инф: 4)
Абитуриенты с плохими оценками:
Абитуриенты со средним баллом выше 7.0
Абитуриент id: 1
| Фамилия: хех | Имя: хох | Отчество: хах | Адрес: ergh | Телефон: 239857 | Оценки: (мат: 9; рус: 8; инф: 7)
Список абитуриентов топ-2
Абитуриент id: 1
| Фамилия: хех | Имя: хох | Отчество: хах | Адрес: ergh | Телефон: 239857 | Оценки: (мат: 9; рус: 8; инф: 7)
Абитуриент id: 3
| Фамилия: hjfgd | Имя: ejrng | Отчество: geng | Адрес: edngr | Телефон: 345 | Оценки: (мат: 10; рус: 5; инф: 5)
Абитуриенты с полупроходным баллом:
Абитуриент id: 3
| Фамилия: hjfgd | Имя: ejrng | Отчество: geng | Адрес: edngr | Телефон: 345 | Оценки: (мат: 10; рус: 5; инф: 5)

Process finished with exit code 0
|

```

Рисунок 4 – Работа программы

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 Простая дробь, используя класс Число. Методы: вывод на экран, сложение, вычитание, умножение, деление.

Код:

```

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

class Number{
    int num;

    public Number(int num) {
        this.num = num;
    }
}

```

```

    }

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

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

    @Override
    public String toString() {
        return String.valueOf(num);
    }

    public Number add(Number other) {
        return new Number(this.num + other.num);
    }

    public Number sub(Number other) {
        return new Number(this.num - other.num);
    }

    public Number mul(Number other) {
        return new Number(this.num * other.num);
    }

    public Number div(Number other) {
        return new Number(this.num / other.num);
    }
}

class Fraction{
    Number nominator;
    Number denominator;

    public Fraction(Number nominator, Number denominator) {
        this.nominator = nominator;
        this.denominator = denominator;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) return true;

```

```

        if (obj == null || getClass() != obj.getClass()) return
false;
        Fraction frac = (Fraction) obj;
        return      (this.nominator.equals(frac.nominator)      &
this.denominator.equals(((Fraction) obj).denominator) );
    }

    @Override
    public String toString() {
        return "Fraction{ " + nominator + " / " + denominator +
" }";
    }

    @Override
    public int hashCode() {
        return Objects.hash(nominator, denominator);
    }

    public void print(){
        System.out.println(this.toString());
    }

    public void makeSimple() {
        int      count      =      Math.min(Math.abs(nominator.num),
Math.abs(denominator.num));

        for (int i = count; i >= 1; i--){
            if (nominator.num % i == 0 && denominator.num % i
== 0){
                count = i;
                break;
            }
        }
        Number nod = new Number(count);
        this.nominator = this.nominator.div(nod);
        this.denominator = this.denominator.div(nod);
    }

    public Fraction add(Fraction other) {
        Number      nom_res_1      =
this.nominator.mul(other.denominator);
        Number      nom_res_2      =
other.nominator.mul(this.denominator);
        Number nom_res = nom_res_1.add(nom_res_2);
        Number      denom_res      =
this.denominator.mul(other.denominator);
        Fraction result = new Fraction(nom_res, denom_res);
        result.makeSimple();
        return result;
    }

```

```

        public Fraction sub(Fraction other) {
            Number          nom_res_1          =
this.nominator.mul(other.denominator);
            Number          nom_res_2          =
other.nominator.mul(this.denominator);
            Number nom_res = nom_res_1.sub(nom_res_2);
            Number          denom_res          =
this.denominator.mul(other.denominator);
            Fraction result = new Fraction(nom_res, denom_res);
            result.makeSimple();
            return result;
        }

        public Fraction mul(Fraction other) {
            Number nom_res = this.nominator.mul(other.nominator);
            Number          denom_res          =
this.denominator.mul(other.denominator);
            Fraction result = new Fraction(nom_res, denom_res);
            result.makeSimple();
            return result;
        }

        public Fraction div(Fraction other) {
            Number nom_res = this.nominator.mul(other.denominator);
            Number          denom_res          =
this.denominator.mul(other.nominator);
            Fraction result = new Fraction(nom_res, denom_res);
            result.makeSimple();
            return result;
        }
    }
}

```

```

public class Third_3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Первая дробь: ");
        System.out.print("\tЧислитель: ");
        Number nominator1 = new Number(scanner.nextInt());
        System.out.print("\tЗнаменатель: ");
        Number denominator1 = new Number(scanner.nextInt());
        Fraction fraction1 = new Fraction(nominator1,
denominator1);

        System.out.println("Вторая дробь: ");
        System.out.print("\tЧислитель: ");
        Number nominator2 = new Number(scanner.nextInt());
        System.out.print("\tЗнаменатель: ");
        Number denominator2 = new Number(scanner.nextInt());
    }
}

```

```

        Fraction    fraction2    =    new    Fraction(nominator2,
denominator2);

        System.out.println("Результат опреации +:");
        fraction1.add(fraction2).print();

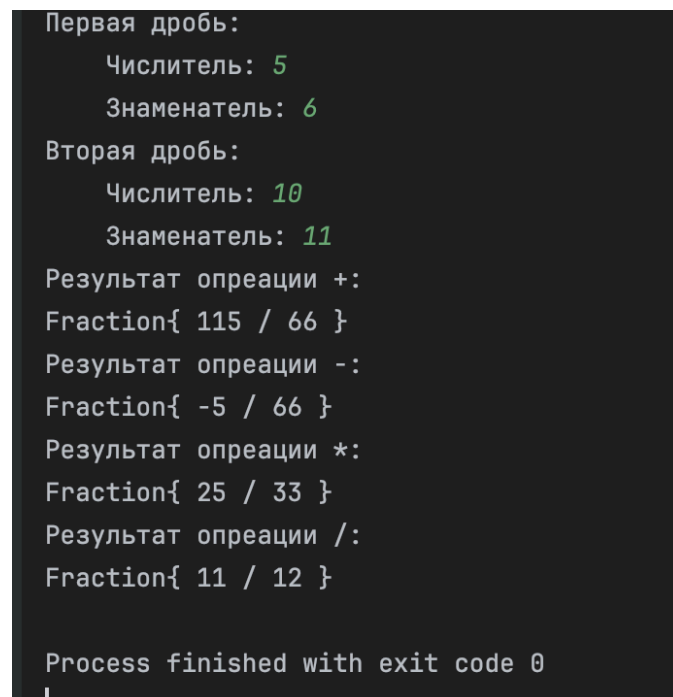
        System.out.println("Результат опреации -:");
        fraction1.sub(fraction2).print();

        System.out.println("Результат опреации *:");
        fraction1.mul(fraction2).print();

        System.out.println("Результат опреации /:");
        fraction1.div(fraction2).print();
    }
}

```

Работа программы показана на рисунке 5.



```

Первая дробь:
    Числитель: 5
    Знаменатель: 6
Вторая дробь:
    Числитель: 10
    Знаменатель: 11
Результат опреации +:
Fraction{ 115 / 66 }
Результат опреации -:
Fraction{ -5 / 66 }
Результат опреации *:
Fraction{ 25 / 33 }
Результат опреации /:
Fraction{ 11 / 12 }

Process finished with exit code 0

```

Рисунок 5 – Работа программы

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

Код:

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

class ObjectToOpen {
    boolean isClosed;
    int width;
    int height;

    ObjectToOpen(int width, int height){
        this.width = width;
        this.height = height;
        this.isClosed = true;
    }

    @Override
    public int hashCode() {
        return Objects.hash(width, height);
    }

    public void close() {
        this.isClosed = true;
    }

    public void open() {
        this.isClosed = false;
    }
}

class Window extends ObjectToOpen{
    Window(int width, int height) {
        super(width, height);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;
        Window window = (Window) o;
        return width == window.width && height == window.height;
    }

    @Override
    public String toString() {
        return "Window{" +
            "isClosed=" + isClosed +
            ", width=" + width +
            ", height=" + height +

```

```

        '}'';
    }
}

class Door extends ObjectToOpen {

    Door(int width, int height) {
        super(width, height);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;
        Door door = (Door) o;
        return width == door.width && height == door.height;
    }

    @Override
    public String toString() {
        return "Door{" +
            "isClosed=" + isClosed +
            ", width=" + width +
            ", height=" + height +
            '}';
    }
}

class House {
    Door[] doors;
    Window[] windows;
    double square;

    public House(Window[] windows, Door[] doors, double square)
{
        this.windows = windows;
        this.doors = doors;
        this.square = square;
    }

    @Override
    public String toString() {
        return "House{" +
            "doors=" + Arrays.toString(doors) +
            ", windows=" + Arrays.toString(windows) +
            ", square=" + square +
            '}';
    }

    @Override

```



```

    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;
        House house = (House) o;
        return Double.compare(square, house.square) == 0 &&
Arrays.equals(doors, house.doors) && Arrays.equals(windows,
house.windows);
    }

    @Override
    public int hashCode() {
        int result = Objects.hash(square);
        result = 31 * result + Arrays.hashCode(doors);
        result = 31 * result + Arrays.hashCode(windows);
        return result;
    }

    public void close(){
        for (Door door : doors) door.close();
        for (Window window : windows) window.close();
    }

    public void open(){
        for (Door door : doors) door.open();
        for (Window window : windows) window.open();
    }

    public int doorsCount(){
        return doors.length;
    }

    public void printDoors(){
        System.out.println("Число          дверей:          "          +
this.doorsCount());
    }

    public int windowsCount(){
        return windows.length;
    }

    public void printWindows(){
        System.out.println("Число          окон:          "          +
this.windowsCount());
    }
}

public class Fourth_3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Введите площадь дома (м^2): ");
    }
}

```

```

double square = scanner.nextDouble();

System.out.print("Введите число окон: ");
int n_win = scanner.nextInt();

Window[] windows = new Window[n_win];
for (int i = 0; i < n_win; i++){
    System.out.print("\tВведите ширину и высоту (через
пробел) окна " + (i+1) + ": ");
    int width = scanner.nextInt();
    int height = scanner.nextInt();
    windows[i] = new Window(width, height);
}

System.out.print("Введите число дверей: ");
int n_door = scanner.nextInt();
Door[] doors = new Door[n_door];
for (int i = 0; i < n_door; i++){
    System.out.print("\tВведите ширину и высоту (через
пробел) двери " + (i+1) + ": ");
    int width = scanner.nextInt();
    int height = scanner.nextInt();
    doors[i] = new Door(width, height);
}

House house = new House(windows, doors, square);
house.open();
System.out.println(house);
house.close();
System.out.println(house);
house.printWindows();
house.printDoors();
}
}

```

Работа программы показана на рисунке 6.



```

Введите площадь дома (м²): 150
Введите число окон: 2
Введите ширину и высоту (через пробел) окна 1: 80 120
Введите ширину и высоту (через пробел) окна 2: 90 120
Введите число дверей: 2
Введите ширину и высоту (через пробел) двери 1: 100 200
Введите ширину и высоту (через пробел) двери 2: 90 200
House{doors=[Door{isClosed=false, width=100, height=200}, Door{isClosed=false, width=90, height=200}], windows=[Window{isClosed=false, width=80, height=120}, Window{isClosed=false, width=90, height=120}]}
House{doors=[Door{isClosed=true, width=100, height=200}, Door{isClosed=true, width=90, height=200}], windows=[Window{isClosed=true, width=80, height=120}, Window{isClosed=true, width=90, height=120}]}
Число окон: 2
Число дверей: 2

```

Рисунок 6 – Работа программы

Задание 7: построить модель программной системы. Система Больница. Пациенту назначается лечащий Врач. Врач может сделать назначение Пациенту (процедуры, лекарства, операции). Медсестра или другой Врач выполняют назначение. Пациент может быть выписан из

Больницы по окончании лечения, при нарушении режима или при иных обстоятельствах.

Код:

```
// Класс Пациент
class Patient {
    String name;
    Doctor doctor;
    boolean discharged;

    String treatment;

    public Patient(String name, Doctor doctor) {
        this.name = name;
        this.doctor = doctor;
        this.discharged = false;
    }

    public void setDischarged(boolean discharged) {
        this.discharged = discharged;
        System.out.println("Пациент " + this.name + " выписан");
    }
}

// Класс Медсестра
class Medical {
    String name;

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

    public void performTreatment(Patient patient) {
        patient.treatment += " - done. | ";
        System.out.println(
            "Пациенту " + patient.name + " выполнено
назначение: " + patient.treatment
            + " медсестрой " + this.name
        );
    }

    public void punish(Patient patient) {
        System.out.println(
            "Пациент " + patient.name + " наказан за
нарушение режима медсестрой " + this.name
        );
        patient.setDischarged(true);
        patient.treatment = "go home";
    }
}
```

```

}

// Класс Врач
class Doctor extends Medical {

    public Doctor(String name) {
        super(name);
    }

    public void setHealthy(Patient patient) {
        System.out.println(
            "Пациент " + patient.name + " выписан"
            + " врачом " + this.name
        );
        patient.setDischarged(true);
        patient.treatment = "healthy";
    }

    public void prescribeTreatment(Patient patient, String
treatment) {
        patient.treatment = treatment;
        System.out.println(
            "Пациенту " + patient.name + " выписано
назначение: " + patient.treatment
            + " врачом " + this.name
        );
    }

    @Override
    public void performTreatment(Patient patient) {
        patient.treatment += " - done. | ";
        System.out.println(
            "Пациенту " + patient.name + " выполнено
назначение: " + patient.treatment
            + " врачом " + this.name
        );
    }

    @Override
    public void punish(Patient patient) {
        System.out.println(
            "Пациент " + patient.name + " наказан за
нарушение режима врачом " + this.name
        );
        patient.setDischarged(true);
        patient.treatment = "go home";
    }
}

```

```

public class Third_4 {
    public static void main(String[] args) {
        Doctor doctor = new Doctor("Иванов");
        Doctor doctor_2 = new Doctor("Сидоров");
        Patient patient = new Patient("Смирнов", doctor);
        Patient patient_2 = new Patient("Соколов", doctor_2);
        Medical nurse = new Medical("Борисова");

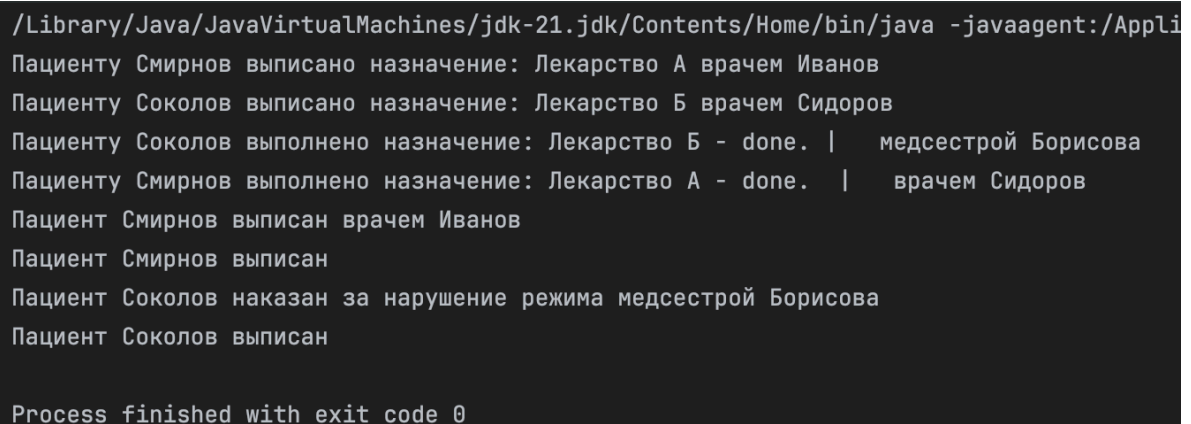
        doctor.prescribeTreatment(patient, "Лекарство А");
        doctor_2.prescribeTreatment(patient_2, "Лекарство Б");

        nurse.performTreatment(patient_2);
        doctor_2.performTreatment(patient);

        doctor.setHealthy(patient);
        nurse.punish(patient_2);
    }
}

```

Работа программы показана на рисунке 7.



```

/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Appli
Пациенту Смирнов выписано назначение: Лекарство А врачом Иванов
Пациенту Соколов выписано назначение: Лекарство Б врачом Сидоров
Пациенту Соколов выполнено назначение: Лекарство Б - done. | медсестрой Борисова
Пациенту Смирнов выполнено назначение: Лекарство А - done. | врачом Сидоров
Пациент Смирнов выписан врачом Иванов
Пациент Смирнов выписан
Пациент Соколов наказан за нарушение режима медсестрой Борисова
Пациент Соколов выписан

Process finished with exit code 0

```

Рисунок 7 – Работа программы

Задание 8: построить модель программной системы. Система Вступительные экзамены. Абитуриент регистрируется на Факультет, сдает Экзамены. Преподаватель выставляет Оценку. Система подсчитывает средний балл и определяет Абитуриентов, зачисленных в учебное заведение.

Код:

```

// Класс Абитуриент
class Applicant {
    String name;

```

```

double mark;
boolean admitted;

Faculty faculty;

public Applicant(String name, Faculty faculty) {
    this.name = name;
    this.mark = 0.0;
    this.admitted = false;
    this.faculty = faculty;
}

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

public void setAdmitted(boolean admitted) {
    this.admitted = admitted;
}
}

// Класс Преподаватель
class Teacher {
    public void giveGrade(Applicant applicant, double mark) {
        applicant.setMark(mark);
    }
}

// Класс Факультет
class Faculty {

    String name;

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

    public double getAverage(Applicant[] applicants) {
        double result = 0;
        int n = 0;
        for (Applicant applicant : applicants) {
            if (applicant.faculty.name == this.name) {
                result += applicant.mark;
                n++;
            }
        }
        return (result / n);
    }

    public void admitApplicants(Applicant[] applicants) {
        double avg = getAverage(applicants);

```

```

        for (Applicant applicant : applicants) {
            if (applicant.faculty.name == this.name &
applicant.mark >=avg) {
                applicant.setAdmitted(true);
            }
        }
    }
}

```

```

public class Fourth_4 {
    public static void main(String[] args) {
        Faculty faculty = new Faculty("Информатика");
        Faculty faculty_2 = new Faculty("Лингвистика");
        Applicant applicant1 = new Applicant("Иванов",
faculty);
        Applicant applicant2 = new Applicant("Петров",
faculty);
        Applicant applicant3 = new Applicant("Смирнов",
faculty);
        Applicant applicant4 = new Applicant("Сидоров",
faculty_2);
        Applicant applicant5 = new Applicant("Борисов",
faculty_2);
        Applicant applicant6 = new Applicant("Звягинцев",
faculty);

```

```

        Teacher teacher = new Teacher();
        teacher.giveGrade(applicant1, 8.5);
        teacher.giveGrade(applicant2, 6.0);
        teacher.giveGrade(applicant3, 7.0);
        teacher.giveGrade(applicant4, 9.0);
        teacher.giveGrade(applicant5, 5.0);
        teacher.giveGrade(applicant6, 7.5);

```

```

        Applicant[] applicants = {
            applicant1,
            applicant2,
            applicant3,
            applicant4,
            applicant5,
            applicant6
        };

```

```

        double avg1 = faculty.getAverage(applicants);
        double avg2 = faculty_2.getAverage(applicants);

```

```

        faculty.admitApplicants(applicants);
        faculty_2.admitApplicants(applicants);

```

```

        System.out.println(

```

```

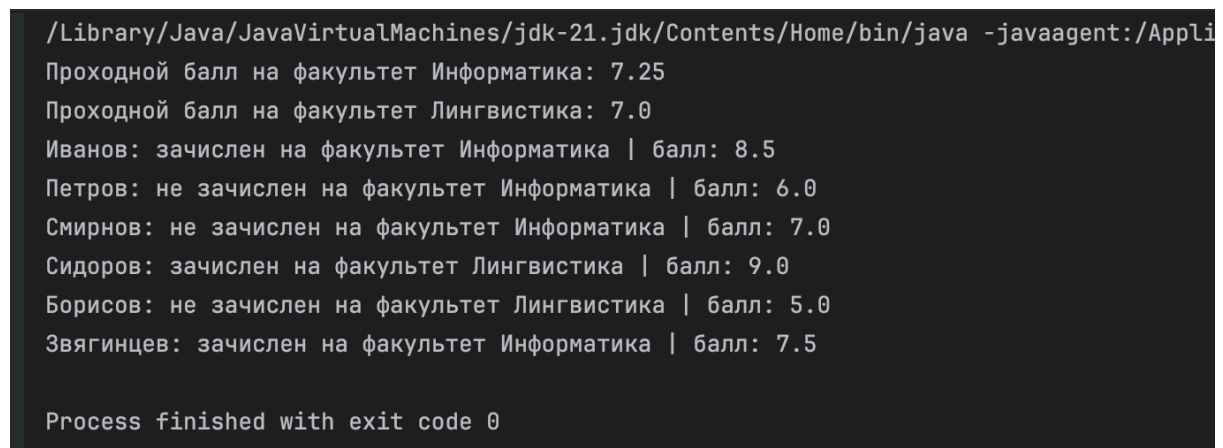
        "Проходной балл на факультет " + faculty.name
        + ": " + avg1
    );

    System.out.println(
        "Проходной балл на факультет " + faculty_2.name
        + ": " + avg2
    );

    for (Applicant applicant : applicants) {
        if (applicant.admitted) {
            System.out.println(
                applicant.name + ": зачислен на
факультет "
                + applicant.faculty.name + " | балл: "
                + applicant.mark
            );
        } else {
            System.out.println(
                applicant.name + ": не зачислен на
факультет "
                + applicant.faculty.name + " |
балл: "
                + applicant.mark
            );
        }
    }
}
}
}

```

Работа программы показана на рисунке 8.



```

/Library/Java/JavaVirtualMachines/jdk-21.jdk/Contents/Home/bin/java -javaagent:/Appli
Проходной балл на факультет Информатика: 7.25
Проходной балл на факультет Лингвистика: 7.0
Иванов: зачислен на факультет Информатика | балл: 8.5
Петров: не зачислен на факультет Информатика | балл: 6.0
Смирнов: не зачислен на факультет Информатика | балл: 7.0
Сидоров: зачислен на факультет Лингвистика | балл: 9.0
Борисов: не зачислен на факультет Лингвистика | балл: 5.0
Звягинцев: зачислен на факультет Информатика | балл: 7.5

Process finished with exit code 0

```

Рисунок 8 – Работа программы

Вывод: были освоены принципы ООП на языке программирования Java