Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

с «09» сентября 2024 г. по «09» ноября 2024 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 21-2к  Козлова Дарья Алексеевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2024

Содержание

[Введение 3](#_Toc182475198)

[1.Разработка программных модулей 4](#_Toc182475199)

[Решение задач 4](#_Toc182475200)

[1.1 Задача 1 4](#_Toc182475201)

[1.2 Задание 2 7](#_Toc182475202)

[1.3 Задание 3 11](#_Toc182475203)

[1.4 Задание 4 15](#_Toc182475204)

[1.5 Задание 5 19](#_Toc182475205)

[1.6 Задание 6 21](#_Toc182475206)

[1.7 Задание 7 21](#_Toc182475207)

[1.8 Задание 8 26](#_Toc182475208)

[1.9 Задание 9 30](#_Toc182475209)

[1.10 Задание 10 36](#_Toc182475210)

[2.Разработка АИС 38](#_Toc182475211)

[2.1 Техническое задание 38](#_Toc182475212)

[2.2 Назначение системы 39](#_Toc182475213)

[2.3. Требования к системе 39](#_Toc182475214)

[2.4. Требования к интерфейсу пользователя 40](#_Toc182475215)

[2.5. Требования к безопасности 40](#_Toc182475216)

[2.6 Краткая спецификация разрабатываемой системы 41](#_Toc182475217)

[2.7 Сущности системы: 41](#_Toc182475218)

[2.8 Основные функции системы: 42](#_Toc182475219)

[3.Разработка мобильного приложения 49](#_Toc182475220)

[4.Разработка сайта 51](#_Toc182475221)

[Заключение 52](#_Toc182475222)

[Список используемой литературы 53](#_Toc182475223)

# Введение

Данный отчёт служит для представления результатов прохождения учебной практики по программированию. Она состоит из пяти дисциплин разработка программных модулей, разработка мобильных приложений, поддержка и тестирование программных модулей, программирование 1С, системное программирование. Так же практика включает создание собственного WEB-сайта.

Цели:

Расширить свои познания и улучшить практические навыки по дисциплинам, создать собственный сайт и решить поставленные задачи для проверки усвоения материала дисциплин.

# 1.Разработка программных модулей

## Решение задач

### 1.1 Задача 1

10. Найти максимальный элемент (ы) в матрице и удалить из матрицы все строки и столбцы, его содержащие.

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| scanner | Scanner | Объект для считывания ввода с клавиатуры. |
| n | int | Размерность создаваемой матрицы. |
| matrix | int[][] | Двумерный массив для хранения элементов матрицы. |
| random | Random | Объект для генерации случайных чисел. |
| max | int | Максимальное значение в матрице. |
| rowsToRemove | List<Integer> | Список индексов строк, которые нужно удалить. |
| colsToRemove | List<Integer> | Список индексов столбцов, которые нужно удалить. |
| updatedMatrix | int[][] | Новая матрица после удаления строк и столбцов. |
| newRow, newCol | int | Вспомогательные переменные для создания новой матрицы. |

Код:

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import java.util.Scanner;

public class MatrixMaxElementRemoval {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Ввод размерности матрицы

System.out.print("Введите размерность матрицы n: ");

int n = scanner.nextInt();

// Создание и заполнение матрицы случайными значениями от -n до n

int[][] matrix = new int[n][n];

Random random = new Random();

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

matrix[i][j] = random.nextInt(2 \* n + 1) - n; // Генерация значений от -n до n

}

}

System.out.println("Сгенерированная матрица:");

printMatrix(matrix);

// Найдем максимальный элемент и его позиции

int max = Integer.MIN\_VALUE;

List<Integer> rowsToRemove = new ArrayList<>();

List<Integer> colsToRemove = new ArrayList<>();

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (matrix[i][j] > max) {

max = matrix[i][j];

rowsToRemove.clear();

colsToRemove.clear();

rowsToRemove.add(i);

colsToRemove.add(j);

} else if (matrix[i][j] == max) {

if (!rowsToRemove.contains(i)) {

rowsToRemove.add(i);

}

if (!colsToRemove.contains(j)) {

colsToRemove.add(j);

}

}

}

}

System.out.println("\nМаксимальный элемент: " + max);

// Удаляем строки и столбцы, содержащие максимальный элемент

int[][] updatedMatrix = removeRowsAndColumns(matrix, rowsToRemove, colsToRemove);

System.out.println("\nМатрица после удаления строк и столбцов:");

printMatrix(updatedMatrix);

scanner.close();

}

// Метод для удаления строк и столбцов

private static int[][] removeRowsAndColumns(int[][] matrix, List<Integer> rowsToRemove, List<Integer> colsToRemove) {

int newRowCount = matrix.length - rowsToRemove.size();

int newColCount = matrix[0].length - colsToRemove.size();

if (newRowCount == 0 || newColCount == 0) {

return new int[0][0]; // Если все строки или столбцы удалены, возвращаем пустую матрицу

}

int[][] newMatrix = new int[newRowCount][newColCount];

int newRow = 0;

for (int i = 0; i < matrix.length; i++) {

if (rowsToRemove.contains(i)) continue;

int newCol = 0;

for (int j = 0; j < matrix[i].length; j++) {

if (colsToRemove.contains(j)) continue;

newMatrix[newRow][newCol] = matrix[i][j];

newCol++;

}

newRow++;

}

return newMatrix;

}

// Метод для печати матрицы

private static void printMatrix(int[][] matrix) {

if (matrix.length == 0) {

System.out.println("Матрица пуста");

return;

}

for (int[] row : matrix) {

for (int elem : row) {

System.out.print(elem + "\t");

}

System.out.println();

}

}

}

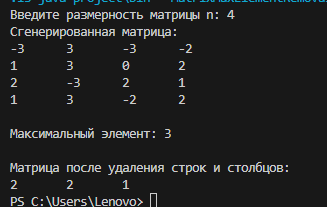


Рисунок 1 – результат задания 1

### 1.2 Задание 2

2. Определить класс Комплекс. Создать массив/список/множество размерности n из комплексных координат. Передать его в метод, который выполнит сложение/умножение его элементов.

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| real, imag | double | Реальная и мнимая части комплексного числа соответственно. |
| other | Complex | Объект класса Complex, используемый при выполнении операций над комплексными числами. |
| newReal, newImag | double | Временные переменные для хранения результатов вычислений при умножении комплексных чисел. |
| scanner | Scanner | Объект для считывания ввода с клавиатуры. |
| n | int | Размерность массива комплексных чисел. |
| complexNumbers | List<Complex> | Список для хранения созданных комплексных чисел. |
| i | int | Индекс цикла для итерации по списку комплексных чисел. |
| sumResult, productResult | Complex | Переменные для хранения результатов сложения и умножения всех комплексных чисел. |
| sum, product | Complex | Временные переменные для накопления результатов сложения и умножения в соответствующих методах. |

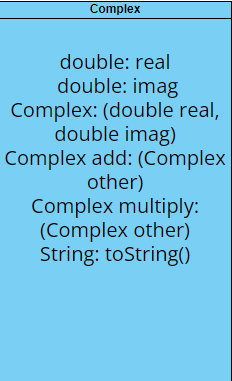


Рисунок 2 - Диаграмма классов задания 2

Код:

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Complex {

private double real; // Реальная часть

private double imag; // Мнимая часть

// Конструктор для создания комплексного числа

public Complex(double real, double imag) {

this.real = real;

this.imag = imag;

}

// Метод для сложения двух комплексных чисел

public Complex add(Complex other) {

return new Complex(this.real + other.real, this.imag + other.imag);

}

// Метод для умножения двух комплексных чисел

public Complex multiply(Complex other) {

double newReal = this.real \* other.real - this.imag \* other.imag;

double newImag = this.real \* other.imag + this.imag \* other.real;

return new Complex(newReal, newImag);

}

// Метод для вывода комплексного числа в строковом формате

@Override

public String toString() {

return String.format("%.2f + %.2fi", real, imag);

}

}

public class ComplexCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Ввод размерности массива

System.out.print("Введите размерность массива комплексных чисел n: ");

int n = scanner.nextInt();

// Создание и заполнение массива комплексных чисел

List<Complex> complexNumbers = new ArrayList<>();

for (int i = 0; i < n; i++) {

System.out.print("Введите реальную часть для комплексного числа " + (i + 1) + ": ");

double real = scanner.nextDouble();

System.out.print("Введите мнимую часть для комплексного числа " + (i + 1) + ": ");

double imag = scanner.nextDouble();

complexNumbers.add(new Complex(real, imag));

}

// Выполнение сложения всех комплексных чисел

Complex sumResult = addAllComplexNumbers(complexNumbers);

System.out.println("Сумма всех комплексных чисел: " + sumResult);

// Выполнение умножения всех комплексных чисел

Complex productResult = multiplyAllComplexNumbers(complexNumbers);

System.out.println("Произведение всех комплексных чисел: " + productResult);

scanner.close();

}

// Метод для сложения всех элементов списка комплексных чисел

public static Complex addAllComplexNumbers(List<Complex> numbers) {

Complex sum = new Complex(0, 0); // Начальное значение для сложения

for (Complex complex : numbers) {

sum = sum.add(complex);

}

return sum;

}

// Метод для умножения всех элементов списка комплексных чисел

public static Complex multiplyAllComplexNumbers(List<Complex> numbers) {

Complex product = new Complex(1, 0); // Начальное значение для умножения

for (Complex complex : numbers) {

product = product.multiply(complex);

}

return product;

}

}

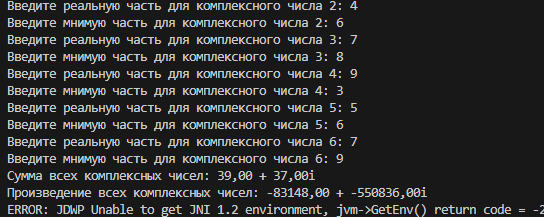


Рисунок 3 – решение задания 2

### 1.3 Задание 3

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

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| fraction1, fraction2, sum, difference, product, quotient | Fraction | Экземпляры класса Fraction, представляющие различные дроби. |
| numerator, denominator | Number | Экземпляры класса Number, хранящие числитель и знаменатель дроби соответственно. |
| value | int | Целочисленное значение, хранящееся в объекте класса Number. |
| args | String[] | Массив строк, содержащий аргументы командной строки (не используется в данном коде). |
| newNumerator, newDenominator | int | Временные переменные для хранения вычисленных числителя и знаменателя при арифметических операциях над дробями. |

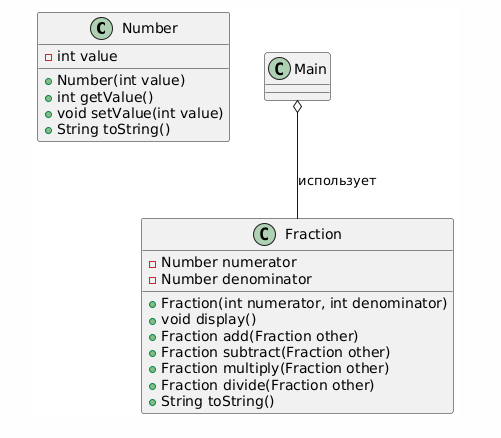


Рисунок 4 – Диаграмма задания 3

Код:

Класс Main:  
public class Main {  
    public static void main(String[] args) {  
        Fraction fraction1 = new Fraction(1, 2);  
        Fraction fraction2 = new Fraction(3, 4);  
  
        System.out.println("Дробь 1: ");  
        fraction1.display();  
  
        System.out.println("Дробь 2: ");  
        fraction2.display();  
  
        System.out.println("Сложение: ");  
        Fraction sum = fraction1.add(fraction2);  
        sum.display();  
  
        System.out.println("Вычитание: ");  
        Fraction difference = fraction1.subtract(fraction2);  
        difference.display();  
  
        System.out.println("Умножение: ");  
        Fraction product = fraction1.multiply(fraction2);  
        product.display();  
  
        System.out.println("Деление: ");  
        Fraction quotient = fraction1.divide(fraction2);  
        quotient.display();  
    }  
}

Класс Number:  
public class Number {  
    private int value;  
  
    public Number(int value) {  
        this.value = value;  
    }  
  
    public int getValue() {  
        return value;  
    }  
  
    public void setValue(int value) {  
        this.value = value;  
    }  
  
    @Override  
    public String toString() {  
        return Integer.toString(value);  
    }  
}

Класс Fraction:  
public class Fraction {  
    private Number numerator;  
    private Number denominator;  
  
    public Fraction(int numerator, int denominator) {  
        if (denominator == 0) {  
            throw new IllegalArgumentException("Denominator cannot be zero");  
        }  
        this.numerator = new Number(numerator);  
        this.denominator = new Number(denominator);  
    }  
  
    // Display the fraction  
    public void display() {  
        System.out.println(this);  
    }  
  
    // Addition  
    public Fraction add(Fraction other) {  
        int newNumerator = (this.numerator.getValue() \* other.denominator.getValue()) +  
                (other.numerator.getValue() \* this.denominator.getValue());  
        int newDenominator = this.denominator.getValue() \* other.denominator.getValue();  
        return new Fraction(newNumerator, newDenominator);  
    }  
  
    // Subtraction  
    public Fraction subtract(Fraction other) {  
        int newNumerator = (this.numerator.getValue() \* other.denominator.getValue()) -  
                (other.numerator.getValue() \* this.denominator.getValue());  
        int newDenominator = this.denominator.getValue() \* other.denominator.getValue();  
        return new Fraction(newNumerator, newDenominator);  
    }  
  
    // Multiplication  
    public Fraction multiply(Fraction other) {  
        int newNumerator = this.numerator.getValue() \* other.numerator.getValue();  
        int newDenominator = this.denominator.getValue() \* other.denominator.getValue();  
        return new Fraction(newNumerator, newDenominator);  
    }  
  
    // Division  
    public Fraction divide(Fraction other) {  
        int newNumerator = this.numerator.getValue() \* other.denominator.getValue();  
        int newDenominator = this.denominator.getValue() \* other.numerator.getValue();  
        if (newDenominator == 0) {  
            throw new ArithmeticException("Division by zero");  
        }  
        return new Fraction(newNumerator, newDenominator);  
    }  
  
    @Override  
    public String toString() {  
        return numerator + "/" + denominator;  
    }  
}

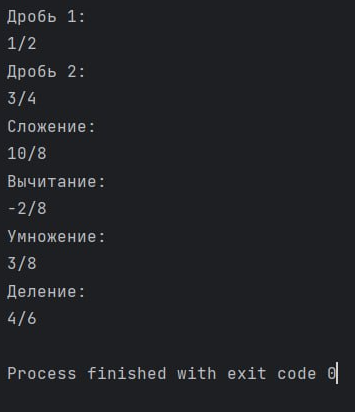


Рисунок 4 – результат задания 3

### 1.4 Задание 4

14. Создать класс Справочная Служба Общественного Транспорта с внутренним классом, с помощью объектов которого можно хранить информацию о времени, линиях маршрутов и стоимости проезда.

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| routeNumber, workingHours, lines, fare | String | Атрибуты класса Route, представляющие номер маршрута, время работы, линии и тариф соответственно. |
| routes | List<Route> | Список для хранения объектов класса Route. |
| route | Route | Временная переменная для хранения текущего маршрута при итерации по списку. |
| service | TransportInfoService | Объект класса TransportInfoService, представляющий справочную службу общественного транспорта. |

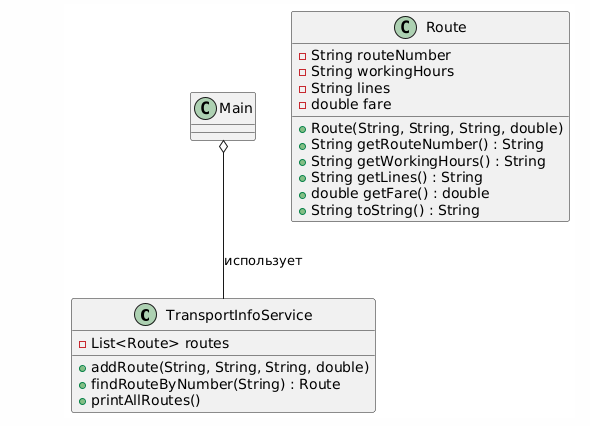


Рисунок 5 – диаграмма задания 4

Код:

java.util.ArrayList;

import java.util.List;

// Класс Справочная Служба Общественного Транспорта

public class TransportInfoService {

// Внутренний класс Route

public class Route {

private String routeNumber;

private String workingHours;

private String lines;

private double fare;

// Конструктор

public Route(String routeNumber, String workingHours, String lines, double fare) {

this.routeNumber = routeNumber;

this.workingHours = workingHours;

this.lines = lines;

this.fare = fare;

}

// Геттеры

public String getRouteNumber() {

return routeNumber;

}

public String getWorkingHours() {

return workingHours;

}

public String getLines() {

return lines;

}

public double getFare() {

return fare;

}

@Override

public String toString() {

return "Route{" +

"routeNumber='" + routeNumber + '\'' +

", workingHours='" + workingHours + '\'' +

", lines='" + lines + '\'' +

", fare=" + fare +

'}';

}

}

// Хранилище маршрутов

private List<Route> routes;

// Конструктор

public TransportInfoService() {

routes = new ArrayList<>();

}

// Метод для добавления маршрута

public void addRoute(String routeNumber, String workingHours, String lines, double fare) {

routes.add(new Route(routeNumber, workingHours, lines, fare));

}

// Метод для поиска маршрутов по номеру

public Route findRouteByNumber(String routeNumber) {

for (Route route : routes) {

if (route.getRouteNumber().equals(routeNumber)) {

return route;

}

}

return null;

}

// Метод для вывода всех маршрутов

public void printAllRoutes() {

for (Route route : routes) {

System.out.println(route);

}

}

// Главный метод для тестирования

public static void main(String[] args) {

TransportInfoService service = new TransportInfoService();

// Добавляем несколько маршрутов

service.addRoute("101", "06:00 - 22:00", "Main St, 1st Ave, 2nd Ave", 1.5);

service.addRoute("102", "07:00 - 23:00", "3rd Ave, 4th Ave, 5th Ave", 2.0);

service.addRoute("103", "08:00 - 20:00", "6th Ave, 7th Ave, 8th Ave", 1.8);

// Ищем маршрут по номеру

System.out.println("Details for route 101:");

Route route = service.findRouteByNumber("101");

if (route != null) {

System.out.println(route);

} else {

System.out.println("Route not found.");

}

// Выводим все маршруты

System.out.println("\nAll routes:");

service.printAllRoutes();

}

}

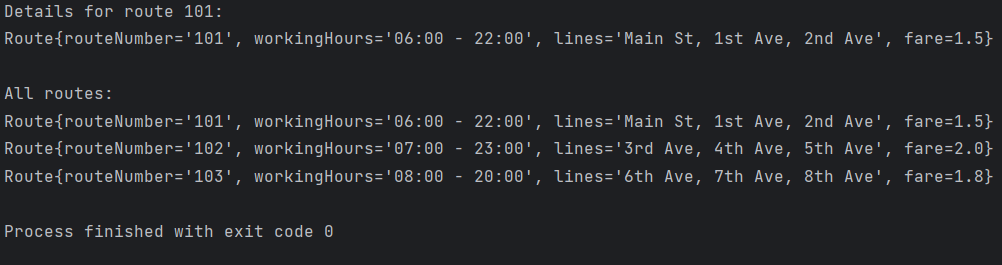


Рисунок 6 – результат задания 4

### 1.5 Задание 5

14. В тексте найти и напечатать все слова максимальной и все слова минимальной длины.

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| text | String | Исходный текст, в котором нужно найти слова. |
| words | String[] | Массив слов, полученный после разбиения текста. |
| minLength, maxLength | int | Переменные для хранения минимальной и максимальной длины слов. |
| minLengthWords, maxLengthWords | List<String> | Списки для хранения слов минимальной и максимальной длины соответственно. |
| word | String | Временная переменная для хранения текущего слова при итерации по массиву слов. |
| length | int | Длина текущего слова. |
| scanner | Scanner | Объект для считывания ввода с клавиатуры. |

Код:

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class WordLengthFinder {

// Метод для нахождения слов максимальной и минимальной длины

public static void findAndPrintWordsByLength(String text) {

// Разделяем текст на слова

String[] words = text.split("\\s+");

if (words.length == 0) {

System.out.println("No words found in the text.");

return;

}

// Инициализируем переменные для минимальной и максимальной длины

int minLength = Integer.MAX\_VALUE;

int maxLength = Integer.MIN\_VALUE;

// Списки для хранения слов минимальной и максимальной длины

List<String> minLengthWords = new ArrayList<>();

List<String> maxLengthWords = new ArrayList<>();

// Определяем минимальную и максимальную длину

for (String word : words) {

int length = word.length();

if (length < minLength) {

minLength = length;

minLengthWords.clear();

minLengthWords.add(word);

} else if (length == minLength) {

minLengthWords.add(word);

}

if (length > maxLength) {

maxLength = length;

maxLengthWords.clear();

maxLengthWords.add(word);

} else if (length == maxLength) {

maxLengthWords.add(word);

}

}

// Выводим слова минимальной длины

System.out.println("Words of minimal length (" + minLength + "):");

for (String word : minLengthWords) {

System.out.println(word);

}

// Выводим слова максимальной длины

System.out.println("\nWords of maximal length (" + maxLength + "):");

for (String word : maxLengthWords) {

System.out.println(word);

}

}

// Главный метод для тестирования

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter text:");

String text = scanner.nextLine();

scanner.close();

findAndPrintWordsByLength(text);

}

}

### 1.6 Задание 6

### 1.7 Задание 7

Составить программу получения на экране рисунка, изображённого в таблице:

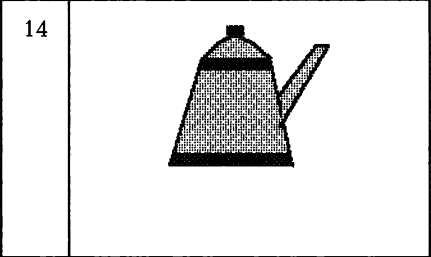


Рисунок 7 - изображение к заданию 7

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| g | Graphics | Объект для рисования на компоненте. |
| g2d | Graphics2D | Расширенный объект для рисования с дополнительными возможностями. |
| xPoints, yPoints | int[] | Массивы координат для рисования многоугольника. |
| noseBaseX, noseBaseY | int | Координаты начала носика чайника. |
| L, W, H | double | Длина, ширина и высота параллелепипеда. |
| p1, p2, p3 | Point | Точки, определяющие плоскость. |
| diagonalPlane | Plane | Объект, представляющий плоскость диагонального сечения. |
| vertices | Point[] | Массив всех вершин параллелепипеда. |
| Переменная | double[] | Векторы, используемые для вычисления нормали к плоскости. |
| g | double | Коэффициенты уравнения плоскости. |

Код:

package com.example.demo2;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class TeapotDrawing extends JPanel {  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 Graphics2D g2d = (Graphics2D) g;  
  
 // Устанавливаем цвет для всего чайника  
 g2d.setColor(Color.*DARK\_GRAY*);  
  
 // Рисуем крышку чайника, по ширине корпуса  
 g2d.fillArc(90, 40, 120, 60, 0, 180); // Купол крышки  
 g2d.setColor(Color.*BLACK*);  
 g2d.fillRect(145, 30, 10, 10); // Кнопка на крышке  
  
 // Увеличиваем толщину края под крышкой  
 g2d.setStroke(new BasicStroke(5)); // Установка большей толщины  
 g2d.setColor(Color.*BLACK*);  
 g2d.drawLine(90, 70, 210, 70); // Прямой край под крышкой  
 g2d.setStroke(new BasicStroke(3)); // Сброс толщины на стандартное значение для остальных частей  
  
 // Рисуем корпус чайника (трапеция с расширением вверх)  
 g2d.setColor(Color.*DARK\_GRAY*);  
 int[] xPoints = {90, 210, 230, 70}; // Точки для верхней части  
 int[] yPoints = {70, 70, 200, 200}; // Высота корпуса  
 g2d.fillPolygon(xPoints, yPoints, 4); // Корпус  
  
 // Рисуем носик (широкий и длинный)  
 g2d.setColor(Color.*DARK\_GRAY*); // Цвет носика  
 int noseBaseX = 210; // Позиция, откуда будет исходить носик (справа)  
 int noseBaseY = 120; // Высота, на которой начинается носик  
  
 // Заполняем носик  
 g2d.fillPolygon(new int[]{noseBaseX, noseBaseX + 60, noseBaseX + 60, noseBaseX},  
 new int[]{noseBaseY, 80, 130, 130}, 4); // Носик в форме трапеции  
  
 // Рисуем нижнюю часть чайника (подставка)  
 g2d.setColor(Color.*DARK\_GRAY*); // Цвет подставки  
 g2d.fillRect(70, 200, 160, 10); // Толстая подставка снизу  
  
 // Рисуем черные контуры  
 g2d.setColor(Color.*BLACK*);  
 g2d.setStroke(new BasicStroke(3)); // Устанавливаем толщину линий для контуров  
 g2d.drawArc(90, 40, 120, 60, 0, 180); // Крышка  
 g2d.drawLine(90, 70, 210, 70); // Прямой край под крышкой  
 g2d.drawPolygon(xPoints, yPoints, 4); // Корпус  
 g2d.drawLine(noseBaseX, noseBaseY, noseBaseX + 60, 80); // Верхняя часть носика  
 g2d.drawLine(noseBaseX + 60, 80, noseBaseX + 60, 110); // Прямой носик  
 g2d.drawLine(noseBaseX + 60, 110, noseBaseX, 130); // Плавный переход к корпусу  
 }  
  
 public static void main(String[] args) {  
 // Создаем окно  
 JFrame frame = new JFrame("Teapot Drawing");  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setSize(400, 300);  
  
 // Добавляем панель с рисунком чайника  
 frame.add(new TeapotDrawing());  
  
 // Делаем окно видимым  
 frame.setVisible(true);  
 }  
}

class Point {

double x, y, z;

Point(double x, double y, double z) {

this.x = x;

this.y = y;

this.z = z;

}

@Override

public String toString() {

return "(" + x + ", " + y + ", " + z + ")";

}

}

class Plane {

double a, b, c, d;

// Уравнение плоскости: ax + by + cz + d = 0

Plane(Point p1, Point p2, Point p3) {

// Векторы для нахождения нормали к плоскости

double[] v1 = {p2.x - p1.x, p2.y - p1.y, p2.z - p1.z};

double[] v2 = {p3.x - p1.x, p3.y - p1.y, p3.z - p1.z};

// Векторное произведение для нормали плоскости

a = v1[1] \* v2[2] - v1[2] \* v2[1];

b = v1[2] \* v2[0] - v1[0] \* v2[2];

c = v1[0] \* v2[1] - v1[1] \* v2[0];

// Уравнение плоскости: a \* x + b \* y + c \* z + d = 0

d = -(a \* p1.x + b \* p1.y + c \* p1.z);

}

// Проверка точки на плоскости

public boolean isOnPlane(Point p) {

return Math.abs(a \* p.x + b \* p.y + c \* p.z + d) < 1e-9;

}

@Override

public String toString() {

return a + "x + " + b + "y + " + c + "z + " + d + " = 0";

}

}

public class DiagonalSection {

public static void main(String[] args) {

// Размеры параллелепипеда

double L = 4.0; // длина

double W = 3.0; // ширина

double H = 2.0; // высота

// Вершины параллелепипеда

Point p1 = new Point(0, 0, 0);

Point p2 = new Point(L, W, 0);

Point p3 = new Point(0, 0, H);

// Создаем плоскость через три точки

Plane diagonalPlane = new Plane(p1, p2, p3);

// Выводим уравнение плоскости

System.out.println("Уравнение плоскости диагонального сечения: " + diagonalPlane);

// Проверяем, лежат ли другие вершины на этой плоскости

Point[] vertices = {

new Point(0, 0, 0), new Point(L, 0, 0), new Point(0, W, 0), new Point(L, W, 0),

new Point(0, 0, H), new Point(L, 0, H), new Point(0, W, H), new Point(L, W, H)

};

System.out.println("Точки на диагональной плоскости:");

for (Point vertex : vertices) {

if (diagonalPlane.isOnPlane(vertex)) {

System.out.println(vertex + " лежит на плоскости");

} else {

System.out.println(vertex + " не лежит на плоскости");

}

}

}

}

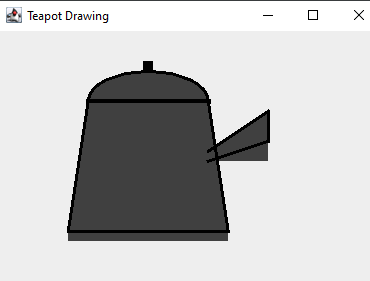


Рисунок 8 – результат задания 7

### 1.8 Задание 8

14. Составить описание класса для определения одномерных массивов строк фиксированной длины. Предусмотреть при этом возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы индексов, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, печати (вывода на экран) элементов массива и всего массива.

|  |  |  |  |
| --- | --- | --- | --- |
| Переменная | Тип данных | Описание | Использование |
| array | String[] | Массив для хранения строк. | Основной массив данных класса. |
| size | int | Размер массива. | Определяет максимальное количество элементов, которые могут быть хранимы в массиве. |
| index | int | Индекс элемента в массиве. | Используется для доступа к элементам массива. |
| value | String | Значение, которое устанавливается в массив. | Новое значение для элемента массива. |
| set | HashSet<String> | Множество для хранения уникальных элементов. | Используется для удаления дубликатов при слиянии массивов. |
| mergedArray, concatenatedArray | FixedSizeStringArray | Новые объекты класса, создаваемые при слиянии и сцеплении массивов. | Результат операций слияния и сцепления. |
| i | int | Счетчик цикла. | Используется для итерации по элементам массива. |

Код:

package com.example.demo1;

import java.util.HashSet;

import java.util.Set;

public class FixedSizeStringArray {

private String[] array;

private int size;

// Конструктор, принимающий размер массива

public FixedSizeStringArray(int size) {

this.size = size;

this.array = new String[size];

}

// Метод для получения строки по индексу

public String get(int index) {

if (index < 0 || index >= size) {

throw new IndexOutOfBoundsException("Индекс выходит за пределы массива: " + index);

}

return array[index];

}

// Метод для установки строки по индексу

public void set(int index, String value) {

if (index < 0 || index >= size) {

throw new IndexOutOfBoundsException("Индекс выходит за пределы массива: " + index);

}

array[index] = value;

}

// Метод для печати элементов массива

public void printArray() {

for (String s : array) {

System.out.println(s);

}

}

// Метод для слияния двух массивов с исключением повторяющихся элементов

public FixedSizeStringArray merge(FixedSizeStringArray other) {

Set<String> set = new HashSet<>();

for (String s : this.array) {

if (s != null) {

set.add(s);

}

}

for (String s : other.array) {

if (s != null) {

set.add(s);

}

}

FixedSizeStringArray mergedArray = new FixedSizeStringArray(set.size());

int index = 0;

for (String s : set) {

mergedArray.set(index++, s);

}

return mergedArray;

}

// Метод для поэлементного сцепления двух массивов

public FixedSizeStringArray concatenate(FixedSizeStringArray other) {

FixedSizeStringArray concatenatedArray = new FixedSizeStringArray(this.size + other.size);

for (int i = 0; i < this.size; i++) {

concatenatedArray.set(i, this.array[i]);

}

for (int i = 0; i < other.size; i++) {

concatenatedArray.set(this.size + i, other.array[i]);

}

return concatenatedArray;

}

// Основной метод для тестирования класса

public static void main(String[] args) {

FixedSizeStringArray array1 = new FixedSizeStringArray(3);

array1.set(0, "Привет");

array1.set(1, "Мир");

array1.set(2, "Java");

FixedSizeStringArray array2 = new FixedSizeStringArray(3);

array2.set(0, "Java");

array2.set(1, "Программирование");

array2.set(2, "Язык");

System.out.println("Массив 1:");

array1.printArray();

System.out.println("\nМассив 2:");

array2.printArray();

FixedSizeStringArray mergedArray = array1.merge(array2);

System.out.println("\nОбъединенный массив (без дубликатов):");

mergedArray.printArray();

FixedSizeStringArray concatenatedArray = array1.concatenate(array2);

System.out.println("\nСцепленный массив:");

concatenatedArray.printArray();

}

}

Результат:

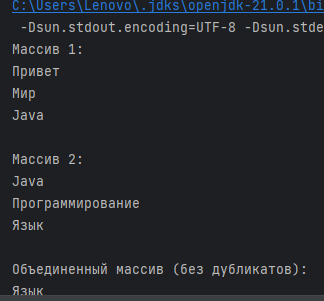


Рисунок 9 – результат задания 8

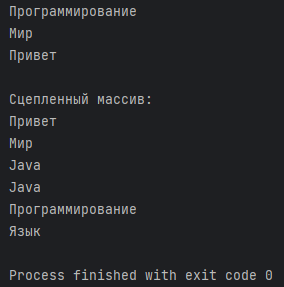


Рисунок 10 – результат задания 8

### 1.9 Задание 9

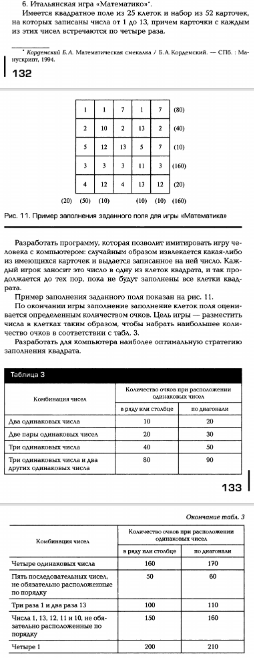


Рисунок 11 – задание 9

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| size | final int | Размер игрового поля (5x5). |
| numbers | final int[] | Массив чисел от 0 до 13, которые используются для заполнения поля. |
| field | int[][] | Двумерный массив для хранения чисел, которыми заполняется игровое поле. |
| filled | boolean[][] | Двумерный массив, указывающий, заполнена ли клетка на поле (true - заполнена, false - нет). |
| counts | final int[] | Массив, задающий количество каждого числа от 0 до 13, которое можно использовать в игре. |
| remainingCounts | int[] | Копия массива counts, используемая для отслеживания оставшихся доступных чисел. |
| scanner | Scanner | Объект для ввода данных от игрока (используется для ввода координат и числа). |
| random | Random | Объект для генерации случайных чисел для хода компьютера. |
| row | int | Переменная для хранения координаты строки, вводимой игроком. |
| col | int | Переменная для хранения координаты столбца, вводимой игроком. |
| number | int | Число, которое вводит игрок или выбирает компьютер для размещения на поле. |
| isPlayerTurn | boolean | Флаг, указывающий, чей сейчас ход (true - ход игрока, false - ход компьютера). |
| playerPoints | int | Количество очков игрока, подсчитанных в конце игры. |
| hasEmptyCells | boolean | Флаг для проверки, остались ли на поле пустые клетки (true - есть пустые клетки). |

Код:

import java.util.Arrays;  
import java.util.Random;  
import java.util.Scanner;  
  
public class MathGame {  
 static final int *SIZE* = 5; // Размер игрового поля (5x5)  
 static final int[] *NUMBERS* = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}; // Числа от 0 до 13  
 static int[][] *field* = new int[*SIZE*][*SIZE*]; // Поле игры  
 static boolean[][] *filled* = new boolean[*SIZE*][*SIZE*]; // Заполненные клетки  
  
 // Таблица 3: Количество чисел в каждой группе  
 static final int[] *counts* = {4, 4, 4, 4, 4, 3, 3, 3, 3, 2, 2, 2, 2, 1};  
  
 static int[] *remainingCounts* = Arrays.*copyOf*(*counts*, *counts*.length); // Оставшиеся количества для чисел  
 static Scanner *scanner* = new Scanner(System.*in*);  
  
 // Инициализация пустого поля  
 public static void initializeField() {  
 for (int i = 0; i < *SIZE*; i++) {  
 for (int j = 0; j < *SIZE*; j++) {  
 *field*[i][j] = -1; // Изначально все клетки пустые (обозначаем -1)  
 }  
 }  
 }  
  
 // Печать игрового поля  
 public static void printField() {  
 for (int i = 0; i < *SIZE*; i++) {  
 for (int j = 0; j < *SIZE*; j++) {  
 if (*field*[i][j] == -1) {  
 System.*out*.print(".\t"); // Пустая клетка  
 } else {  
 System.*out*.print(*field*[i][j] + "\t"); // Заполненная клетка  
 }  
 }  
 System.*out*.println();  
 }  
 }  
  
 // Проверка, можно ли добавить число (остались ли свободные ячейки для числа)  
 public static boolean canAddNumber(int number) {  
 int index = Arrays.*binarySearch*(*NUMBERS*, number);  
 return *remainingCounts*[index] > 0;  
 }  
  
 // Получение случайного доступного числа  
 public static int getRandomNumber() {  
 Random random = new Random();  
 int index;  
 do {  
 index = random.nextInt(*NUMBERS*.length);  
 } while (*remainingCounts*[index] == 0); // Ищем число, которое можно использовать  
 *remainingCounts*[index]--;  
 return *NUMBERS*[index];  
 }  
  
 // Ход игрока (человека)  
 public static void playerMove() {  
 System.*out*.println("Ваш ход. Введите координаты (строка и столбец):");  
 int row, col;  
 do {  
 System.*out*.print("Строка (0-4): ");  
 row = *scanner*.nextInt();  
 System.*out*.print("Столбец (0-4): ");  
 col = *scanner*.nextInt();  
 } while (row < 0 || row >= *SIZE* || col < 0 || col >= *SIZE* || *filled*[row][col]); // Проверка на корректность хода  
  
 int number;  
 do {  
 System.*out*.print("Введите число от 0 до 13: ");  
 number = *scanner*.nextInt();  
 } while (number < 0 || number > 13 || !*canAddNumber*(number)); // Проверка, можно ли использовать это число  
  
 *field*[row][col] = number;  
 *filled*[row][col] = true; // Отмечаем клетку как заполненную  
 }  
  
 // Ход компьютера  
 public static void computerMove() {  
 System.*out*.println("Ход компьютера:");  
 Random random = new Random();  
 int row, col;  
 do {  
 row = random.nextInt(*SIZE*);  
 col = random.nextInt(*SIZE*);  
 } while (*filled*[row][col]); // Ищем свободную клетку  
  
 int number = *getRandomNumber*(); // Получаем случайное доступное число  
 *field*[row][col] = number;  
 *filled*[row][col] = true; // Отмечаем клетку как заполненную  
 System.*out*.println("Компьютер поставил " + number + " в клетку (" + row + ", " + col + ")");  
 }  
  
 // Подсчет очков за комбинации  
 public static int calculatePoints() {  
 int points = 0;  
  
 // Проверка на четыре одинаковых числа в строке  
 for (int i = 0; i < *SIZE*; i++) {  
 for (int j = 0; j <= *SIZE* - 4; j++) {  
 if (*field*[i][j] == *field*[i][j + 1] && *field*[i][j] == *field*[i][j + 2] && *field*[i][j] == *field*[i][j + 3]) {  
 points += 160; // Награда за 4 одинаковых числа в ряду  
 }  
 }  
 }  
  
 // Можно добавить дополнительные проверки (столбцы, диагонали и т.д.)  
 return points;  
 }  
  
 // Основной игровой цикл  
 public static void playGame() {  
 *initializeField*(); // Инициализация пустого поля  
 boolean isPlayerTurn = true; // Ход игрока  
  
 // Игра продолжается, пока есть пустые клетки  
 while (true) {  
 *printField*(); // Печать текущего состояния поля  
  
 if (isPlayerTurn) {  
 *playerMove*(); // Ход игрока  
 } else {  
 *computerMove*(); // Ход компьютера  
 }// Проверяем, остались ли свободные клетки  
 boolean hasEmptyCells = false;  
 for (boolean[] row : *filled*) {  
 for (boolean cell : row) {  
 if (!cell) {  
 hasEmptyCells = true;  
 break;  
 }  
 }  
 }  
 if (!hasEmptyCells) break; // Если нет свободных клеток, игра заканчивается  
  
 // Смена хода  
 isPlayerTurn = !isPlayerTurn;  
 }  
  
 // Подсчет очков после игры  
 int playerPoints = *calculatePoints*();  
 System.*out*.println("Игра завершена.");  
 *printField*(); // Печать финального состояния поля  
 System.*out*.println("Ваши очки: " + playerPoints);  
 }  
  
 public static void main(String[] args) {  
 *playGame*(); // Запуск игры  
 }  
}

Результат:

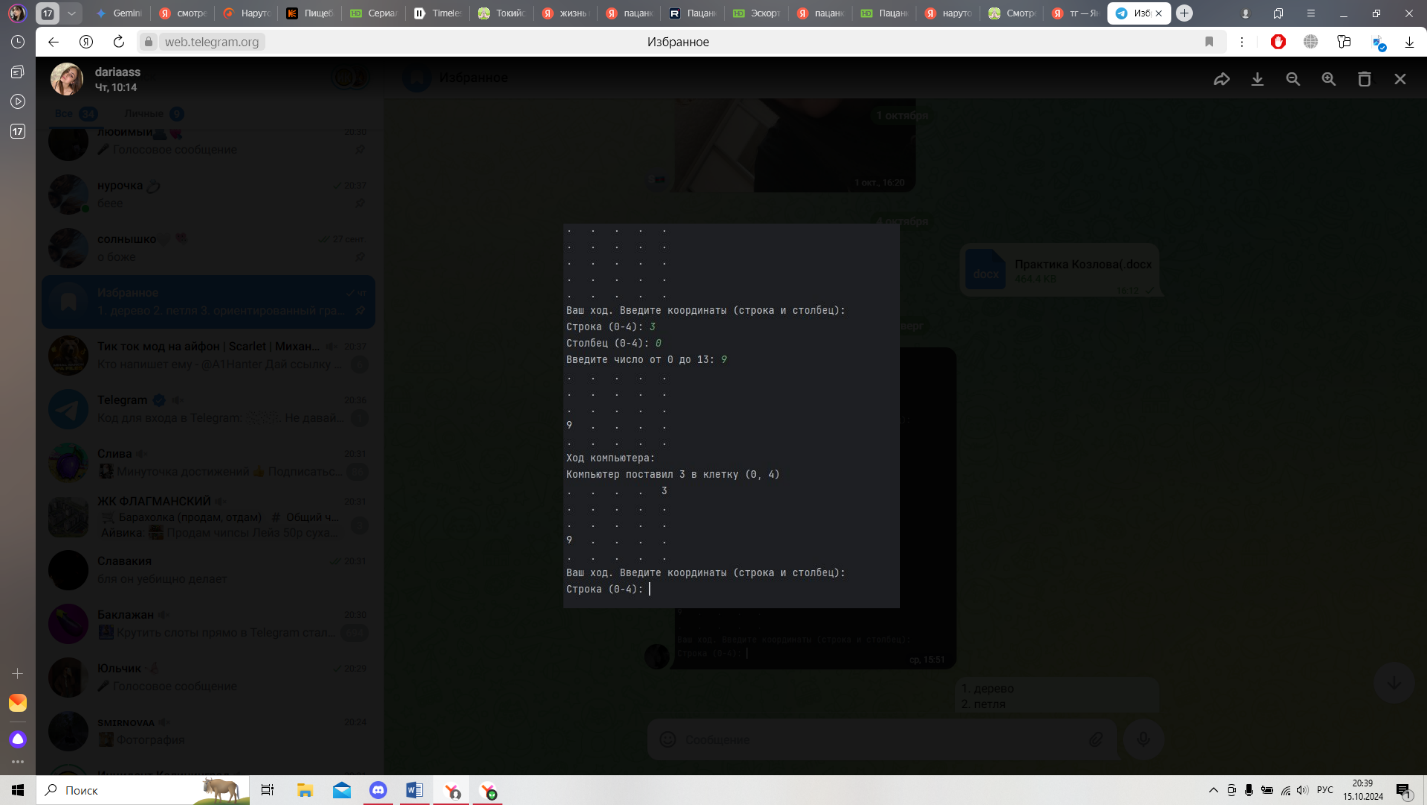
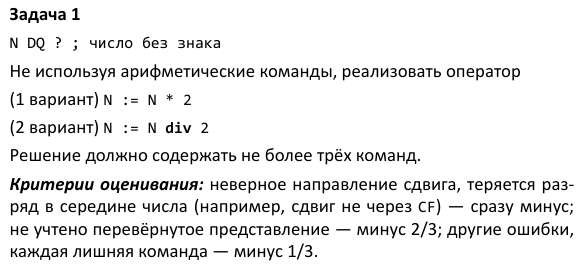


Рисунок 12 – решение задания 9

### 1.10 Задание 10



Вариант 2

# 2.Разработка АИС

Вариант 14 – Деканат

    В качестве серверной части разработать базу данных в СУБД MySQL в соответствии с вариантом. База данных содержит таблицу, состоящую из не менее чем 5-ти полей и 20-ти записей.

    Клиентская часть должна быть разработана на языке Java. Каждый запрос должен быть реализован отдельным методом. Все методы могут принадлежать одному классу. Отдельно должен быть реализован класс, содержащий метод main().

    Таким образом, структура клиента должна быть такой:

1. Класс, содержащий статический метод main() для демонстрации возможностей программы.
2. Класс, содержащий поля и методы для работы с удалённой базой данных:
3. Обязательные методы (запросы) клиента:

- выборка содержимого таблиц базы данных;

- запись в файл результата выборки содержимого таблиц БД.

- вывод в консоль результатов выполнения каждого метода.

1. Дополнительные методы (запросы) клиента
   * добавление записи в таблицу.
   * удаление записи из таблицы.
   * обновление записи в таблице.
   * поиск записи по признаку (на ваш выбор).

### 2.1 Техническое задание

1. Общие сведения

1.1. Название системы: «Деканат»  
1.2. Разработчик: Козлова Дарья Алексеевна  
1.3. Основание для разработки: необходимость автоматизации процесса управления учебным процессом, учета студентов, преподавателей, учебных курсов и успеваемости.

### 2.2 Назначение системы

2.2.1 Система предназначена для автоматизации процессов управления учебным процессом в деканате, учета студентов, преподавателей, учебных курсов и успеваемости.  
2.2.2 Система должна обеспечивать:

* Ведение базы данных студентов, преподавателей и учебных курсов;
* Учёт и анализ успеваемости студентов, включая информацию о пройденных дисциплинах, оценках и посещаемости;
* Хранение информации о студентах, преподавателях, учебных группах и расписании занятий.

### 2.3. Требования к системе

2.3.1 Функциональные требования:

1. Модуль управления студентами:

* Хранение информации о студентах, включая ФИО, дату рождения, адрес, контактные данные, группу, учебный курс, статус (обучается, отчислен, в академическом отпуске и т.д.).

1. Модуль управления преподавателями:

* Ведение базы данных преподавателей, содержащей ФИО, должность, стаж, контактные данные и специализацию.

1. Модуль управления учебными курсами:

* Ведение списка учебных курсов, который включает информацию о названии курса, преподавателе, количестве часов, кредиты, учебной группе, расписании занятий.

1. Модуль учета успеваемости:

* Учёт оценок студентов по каждому предмету, включая даты экзаменов и зачётов;
* Формирование сводных отчетов по успеваемости за семестр, курсов и экзаменационных оценок студентов.

1. Модуль расписания:

* Хранение данных о расписании занятий, включая предмет, аудиторию, время и дни проведения.

2.3.2 Нефункциональные требования

* Удобный пользовательский интерфейс для быстрого доступа к информации о студентах, преподавателях и курсах;
* Возможность интеграции с системами учёта учебных достижений и автоматизированными платформами управления учебными процессами;
* Сохранность данных и регулярное резервное копирование базы данных.

### 2.4. Требования к интерфейсу пользователя

2.4.1. Система должна иметь простой и интуитивно понятный интерфейс с меню для удобного переключения между различными модулями (студенты, преподаватели, курсы, успеваемость, расписание).  
2.4.2. Интерфейс должен быть адаптирован для работы на компьютерах и поддерживать одновременную работу нескольких пользователей с различными уровнями доступа.

### 2.5. Требования к безопасности

2.5.1. Доступ к системе должен быть защищён системой аутентификации пользователей (логин и пароль).  
2.5.2. Должна быть предусмотрена возможность разграничения прав доступа (администратор, преподаватель, сотрудник деканата) для работы с данными студентов, преподавателей и учебных курсов.

### 2.6 Краткая спецификация разрабатываемой системы

Цель системы:

Создание системы управления учебными процессами в деканате, которая позволит автоматизировать процессы учёта студентов, преподавателей, курсов и успеваемости.

### 2.7 Сущности системы:

1. Студенты

* Хранит данные о студентах, включая ФИО, дату рождения, адрес, контактные данные, учебную группу, курс, статус (обучается, отчислен и т.д.).

1. Преподаватели

* Хранит данные о преподавателях, такие как ФИО, должность, стаж, контактные данные и специальность.

1. Учебные курсы

* Включает информацию о курсах, которые студенты изучают в течение семестра: название курса, преподаватель, количество часов, кредиты, учебные группы.

1. Успеваемость

* Содержит информацию о текущей успеваемости студентов: оценках по каждому курсу, посещаемости, дате экзаменов и зачётов.

1. Расписание

* Хранит данные о расписании занятий по курсам, аудиториях, времени и днях проведения.

Входные данные:

* Студенты: данные о зачисленных студентах, их учебных группах, курсе и статусе.
* Преподаватели: данные о преподавателях (ФИО, должность, стаж, контактные данные и специализация).
* Учебные курсы: информация о курсах, преподавателях, учебных часах и кредитах.
* Успеваемость: информация об оценках, посещаемости и датах экзаменов по курсам.
* Расписание: данные о днях, времени проведения занятий и аудиториях.

Выходные данные:

* Отчёты по успеваемости: информация о текущих оценках студентов по курсам и итоговых результатах.
* Информация по студентам: данные о зачисленных студентах, их учебных группах и статусах.
* Информация по преподавателям: данные о преподавателях, их занятиях и специализации.
* Расписание: сводное расписание занятий по курсам, с указанием времени, аудиторий и преподавателей.

### 2.8 Основные функции системы:

1. Управление данными студентов, их учебным статусом и группами;
2. Управление данными преподавателей, включая назначение на курсы и учёт специальностей;
3. Управление учебными курсами, их расписанием и аудиториями;
4. Учёт успеваемости студентов, включая экзамены, зачёты и сводные отчёты по оценкам.

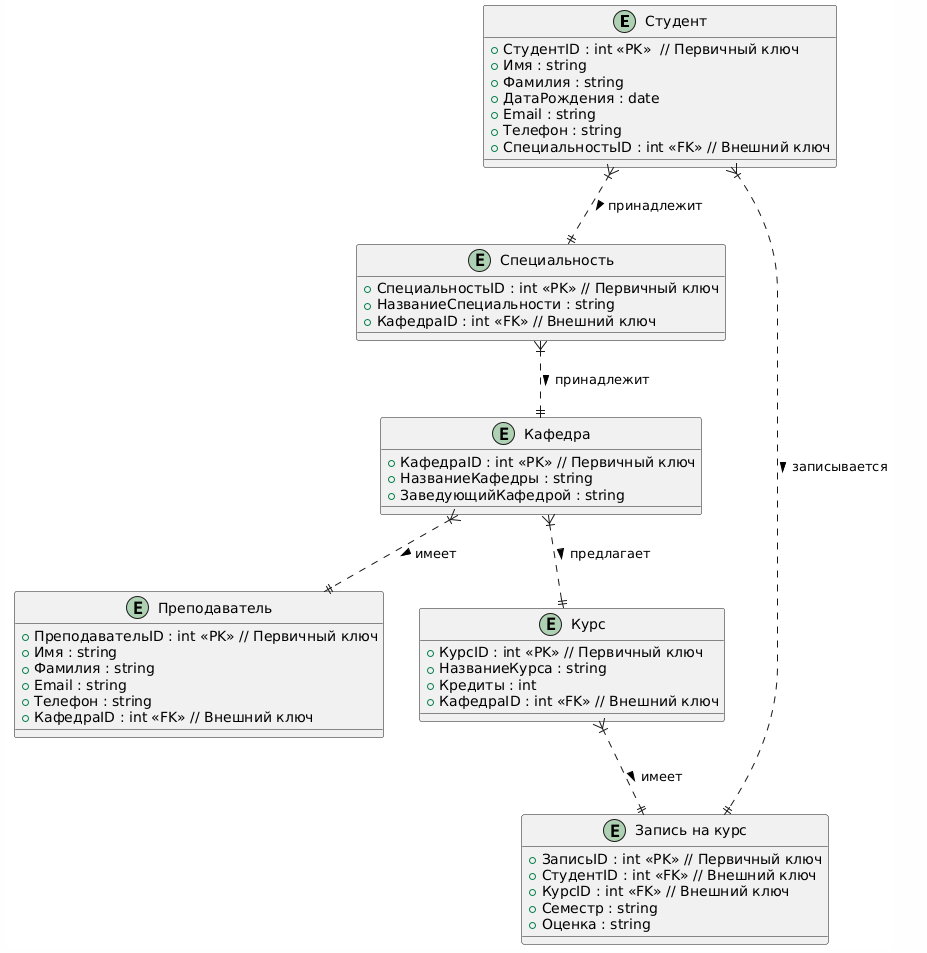


Рисунок 13 – ER-диаграмма

Код:

package org.example;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.util.ArrayList;

import java.util.List;

class Student {

int studentID;

String name;

String surname;

String birthDate;

String email;

String phone;

int specialtyID;

public Student(int studentID, String name, String surname, String birthDate, String email, String phone, int specialtyID) {

this.studentID = studentID;

this.name = name;

this.surname = surname;

this.birthDate = birthDate;

this.email = email;

this.phone = phone;

this.specialtyID = specialtyID;

}

}

class Specialty {

int specialtyID;

String name;

int departmentID;

public Specialty(int specialtyID, String name, int departmentID) {

this.specialtyID = specialtyID;

this.name = name;

this.departmentID = departmentID;

}

}

class Department {

int departmentID;

String name;

String head;

public Department(int departmentID, String name, String head) {

this.departmentID = departmentID;

this.name = name;

this.head = head;

}

}

class Teacher {

int teacherID;

String name;

String surname;

String email;

String phone;

int departmentID;

public Teacher(int teacherID, String name, String surname, String email, String phone, int departmentID) {

this.teacherID = teacherID;

this.name = name;

this.surname = surname;

this.email = email;

this.phone = phone;

this.departmentID = departmentID;

}

}

class Course {

int courseID;

String name;

int credits;

int departmentID;

public Course(int courseID, String name, int credits, int departmentID) {

this.courseID = courseID;

this.name = name;

this.credits = credits;

this.departmentID = departmentID;

}

}

class Enrollment {

int enrollmentID;

int studentID;

int courseID;

String semester;

String grade;

public Enrollment(int enrollmentID, int studentID, int courseID, String semester, String grade) {

this.enrollmentID = enrollmentID;

this.studentID = studentID;

this.courseID = courseID;

this.semester = semester;

this.grade = grade;

}

}

public class DeanOfficeDatabaseGUI extends JFrame {

private List<Student> students = new ArrayList<>();

private DefaultTableModel studentTableModel;

public DeanOfficeDatabaseGUI() {

super("Деканат: База данных");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(800, 600);

setLocationRelativeTo(null);

// Таблица для студентов

studentTableModel = new DefaultTableModel(new Object[]{"ID", "Имя", "Фамилия", "Дата рождения", "Email", "Телефон", "Специальность ID"}, 0);

JTable studentTable = new JTable(studentTableModel);

JScrollPane studentScrollPane = new JScrollPane(studentTable);

// Панель кнопок

JPanel buttonPanel = new JPanel();

JButton addButton = new JButton("Добавить студента");

JButton editButton = new JButton("Редактировать студента");

JButton deleteButton = new JButton("Удалить студента");

buttonPanel.add(addButton);

buttonPanel.add(editButton);

buttonPanel.add(deleteButton);

getContentPane().add(studentScrollPane, BorderLayout.CENTER);

getContentPane().add(buttonPanel, BorderLayout.SOUTH);

addButton.addActionListener(e -> addStudent());

editButton.addActionListener(e -> editStudent());

deleteButton.addActionListener(e -> deleteStudent());

setVisible(true);

}

private void addStudent() {

String name = JOptionPane.showInputDialog(this, "Имя:");

String surname = JOptionPane.showInputDialog(this, "Фамилия:");

String birthDate = JOptionPane.showInputDialog(this, "Дата рождения:");

String email = JOptionPane.showInputDialog(this, "Email:");

String phone = JOptionPane.showInputDialog(this, "Телефон:");

int specialtyID = Integer.parseInt(JOptionPane.showInputDialog(this, "Специальность ID:"));

int newID = students.size() + 1; // Генерация ID

Student newStudent = new Student(newID, name, surname, birthDate, email, phone, specialtyID);

students.add(newStudent);

studentTableModel.addRow(new Object[]{newID, name, surname, birthDate, email, phone, specialtyID});

}

private void editStudent() {

JTable studentTable = new JTable();

int selectedRow = studentTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Пожалуйста, выберите студента для редактирования.");

return;

}

Student student = students.get(selectedRow);

student.name = JOptionPane.showInputDialog(this, "Имя:", student.name);

student.surname = JOptionPane.showInputDialog(this, "Фамилия:", student.surname);

student.birthDate = JOptionPane.showInputDialog(this, "Дата рождения:", student.birthDate);

student.email = JOptionPane.showInputDialog(this, "Email:", student.email);

student.phone = JOptionPane.showInputDialog(this, "Телефон:", student.phone);

student.specialtyID = Integer.parseInt(JOptionPane.showInputDialog(this, "Специальность ID:", student.specialtyID));

studentTableModel.setValueAt(student.name, selectedRow, 1);

studentTableModel.setValueAt(student.surname, selectedRow, 2);

studentTableModel.setValueAt(student.birthDate, selectedRow, 3);

studentTableModel.setValueAt(student.email, selectedRow, 4);

studentTableModel.setValueAt(student.phone, selectedRow, 5);

studentTableModel.setValueAt(student.specialtyID, selectedRow, 6);

}

private void deleteStudent() {

JTable studentTable = null;

int selectedRow = studentTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Пожалуйста, выберите студента для удаления.");

return;

}

students.remove(selectedRow);

studentTableModel.removeRow(selectedRow);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(DeanOfficeDatabaseGUI::new);

}

}

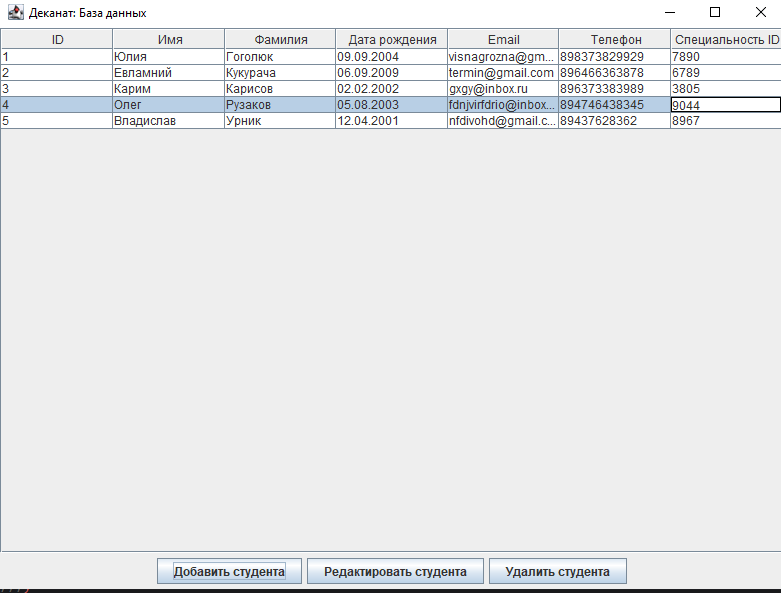


Рисунок 13 – результат разработки АИС

# 3.Разработка мобильного приложения



Рисунок 14 – добавление студента

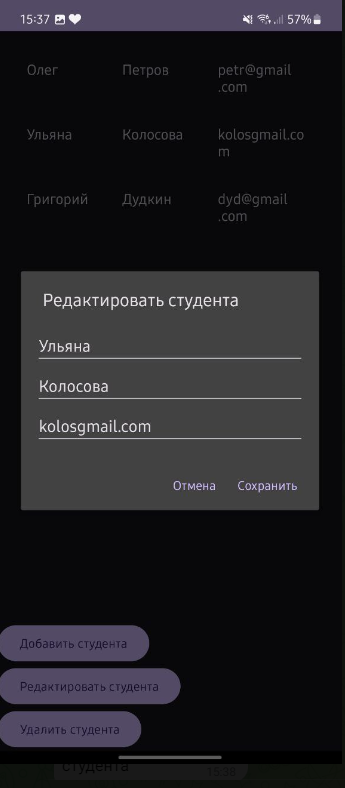


Рисунок 15 – редактирование студента



Рисунок 16 – обновление данных о студенте



Рисунок 17 – удаление студента

# 4.Разработка сайта

<https://yc4745.craftum.io/>

Заключение  
В ходе учебной практики по программированию удалось достичь поставленных целей и укрепить знания по основным дисциплинам. В рамках практики были разработаны программные модули, создано мобильное приложение, проведено тестирование и поддержка кода, а также решены задачи в области системного программирования. Все задания выполнены в соответствии с требованиями, а результаты работы представлены на сайте, созданном для демонстрации проделанной работы. Практика позволила не только углубить теоретические знания, но и приобрести важный практический опыт, необходимый для дальнейшего профессионального развития в сфере программирования.

# Список используемой литературы

1. Иванова, Е. П. Разработка мобильных приложений для Android: учебное пособие / Е. П. Иванова. – 3-е изд., перераб. – Санкт-Петербург: Питер, 2022. – 280 с.
2. Смирнов, Д. С. Базы данных и системы управления ими: учебное пособие для вузов / Д. С. Смирнов, В. И. Поляков. – Москва: Академия, 2021. – 240 с.
3. Михайлов, А. В. Программирование на Java для начинающих: учебное пособие / А. В. Михайлов, Л. Г. Козлова. – Санкт-Петербург: Лань, 2023. – 320 с.
4. Соколова, О. Н. Основы мобильной разработки на платформе Android: учебное пособие для студентов / О. Н. Соколова, Р. П. Исаев. – Москва: КноРус, 2020. – 210 с.
5. Трофимов, П. И. Проектирование баз данных: учебное пособие / П. И. Трофимов, И. А. Захаров. – Екатеринбург: УрФУ, 2019. – 185 с.
6. Беляева, Н. М. Программирование и разработка приложений на Java: учебное пособие / Н. М. Беляева. – Москва: Форум, 2023. – 276 с.
7. Лебедев, С. Н. Основы работы с базами данных в MySQL: учебное пособие / С. Н. Лебедев. – Новосибирск: НГТУ, 2022. – 195 с.
8. Сидоров, Ю. Б. Введение в мобильную разработку на Android: учебное пособие / Ю. Б. Сидоров, А. В. Кузнецов. – Москва: РИПОЛ классик, 2021. – 158 с.
9. Петрова, Е. Л. Управление данными в современных базах данных: учебное пособие / Е. Л. Петрова, И. О. Коваленко. – Казань: КазГТУ, 2020. – 208 с.
10. Филиппов, К. Р. Android-разработка: руководство для начинающих / К. Р. Филиппов, Л. А. Головин. – Санкт-Петербург: БХВ-Петербург, 2023. – 310 с.