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 «20» ноября 2023 г. по «09» декабря 2023 г.

Место практики ГБУ КО ПОО «КИТиС»

|  |  |
| --- | --- |
| Выполнил: | студент 4 курса,  группы ИСп 20-1  Шушлебина Александра Сергеевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2023

|  |
| --- |
| Содержание  [Введение 3](#_Toc152935442)  [1 Решение задач 4](#_Toc152935443)  [2.Разработка АИС “Расписание занятий” 41](#_Toc152935444)  [2.1 ER-диаграмма «сущность-связь» 46](#_Toc152935445)  [2.2 «Техническое задание» 49](#_Toc152935446)  [3.Руководство пользователя: Система Расписания 55](#_Toc152935447)  [4.Разработка сайта 58](#_Toc152935448)  [Заключение 60](#_Toc152935449)  [Список использованных источников 61](#_Toc152935450) |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

# Введение

В современном мире информационные системы играют ключевую роль в повседневной деятельности людей и организаций. Они предоставляют инструменты для хранения, обработки и передачи информации, обеспечивая эффективное управление ресурсами, принятие обоснованных решений и обеспечение коммуникаций.

Практика направлена на развитие ключевых профессиональных навыков, таких как анализ требований пользователя, проектирование архитектуры системы, программирование, тестирование, а также управление проектами в контексте информационных технологий.

Мы имеем возможность погрузиться в процесс тестирования программного обеспечения, управления проектами, а также в изучение принципов безопасности информационных систем.

Учебная практика предоставляет возможность не только закрепить теоретические знания, полученные в течение учебы, но и приобрести ценный опыт, который будет полезен в будущей профессиональной деятельности

## 1 Решение задач

Задача 1

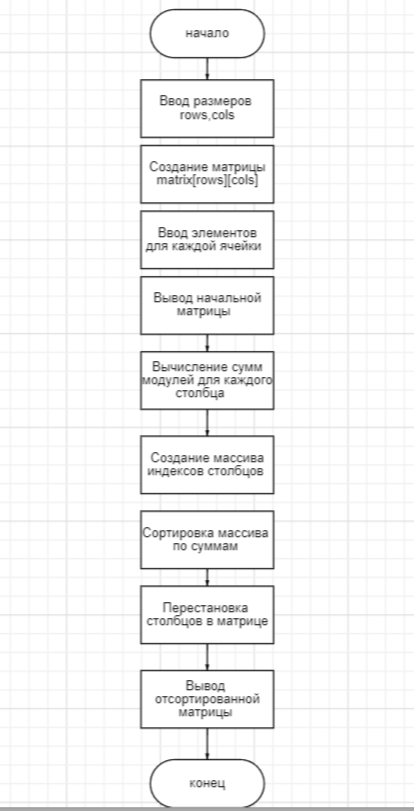
Описание задания:

Вариант 23. Перестроить заданную матрицу, переставляя в ней столбцы так, чтобы значение их характеристик убывали.

Таблица 1 – Таблица идентификаторов:

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Назначение |
| MatrixSorter | Класс | Входная |
| sortMatrixColumns | Метод | Входная |
| matrix | Двумерный массив | Входная |
| rowCount | Целое число | Входная |
| colCount | Целое число | Входная |
| columnSums | Массив | Входная/выходная |
| columnIndices | Массив Integer[] | Входная |
| sortedMatrix | Двумерный массив | Входная |
| scanner | Scanner | Входная |
| rows и cols | Целое число | Входная |
| i и j | Целое число | Входная/выходная |
| printMatrix | Метод | Входная/выходная |

Блок-схема алгоритма:



Код программы:

import java.util.Arrays;

import java.util.Scanner;

public class MatrixSorter {

public static void sortMatrixColumns(int[][] matrix) {

int rowCount = matrix.length;

int colCount = matrix[0].length;

int[] columnSums = new int[colCount];

// Вычисляем суммы модулей элементов для каждого столбца

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

int sum = 0;

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

sum += Math.abs(matrix[i][j]);

}

columnSums[j] = sum;

}

// Создаем массив индексов столбцов и сортируем их по суммам

Integer[] columnIndices = new Integer[colCount];

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

columnIndices[i] = i;

}

Arrays.sort(columnIndices, (a, b) -> Integer.compare(columnSums[b], columnSums[a]));

// Перестраиваем столбцы в матрице

int[][] sortedMatrix = new int[rowCount][colCount];

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

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

sortedMatrix[i][j] = matrix[i][columnIndices[j]];

}

}

// Записываем отсортированную матрицу обратно

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

System.arraycopy(sortedMatrix[i], 0, matrix[i], 0, colCount);

}

}

}

public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
  
 System.out.print("Введите количество строк: ");  
 int rows = scanner.nextInt();  
 System.out.print("Введите количество столбцов: ");  
 int cols = scanner.nextInt();  
  
 int[][] matrix = new int[rows][cols];  
  
 System.out.println("Введите элементы матрицы:");  
  
 for (int i = 0; i < rows; i++) {  
 for (int j = 0; j < cols; j++) {  
 matrix[i][j] = scanner.nextInt();  
 }  
 }  
  
 System.out.println("Начальная матрица:");  
 printMatrix(matrix);  
  
 MatrixSorter.sortMatrixColumns(matrix);  
  
 System.out.println("Матрица после сортировки:");  
 printMatrix(matrix);  
 }  
  
 public static void printMatrix(int[][] matrix) {  
 for (int[] row : matrix) {  
 for (int col : row) {  
 System.out.print(col + "\t");  
 }  
 System.out.println();  
 }  
 }  
}

Результат выполнения программы:

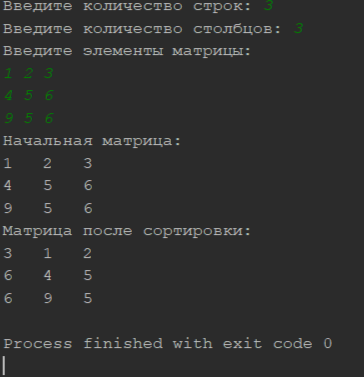


Рисунок 1 – Вывод программы 1

Задача 2

Описание задания:

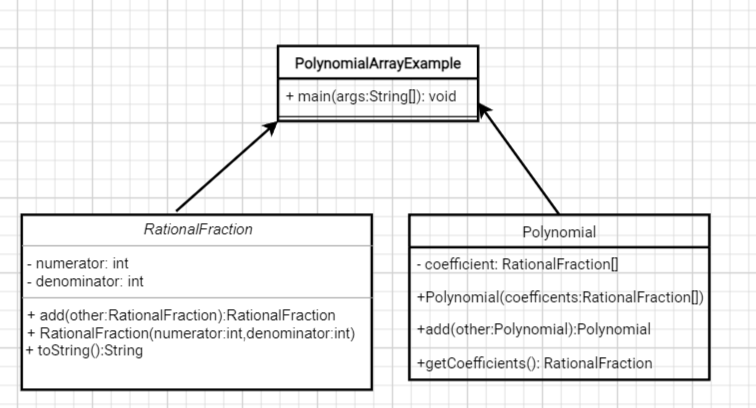
Вариант 23, по учебнику – С1. Определить класс Полином степени n.

Объявить массив/список/множество из m полиномов и определить сумму полиномов массива.

Таблица 2 – Таблица идентификаторов:

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Назначение |
| RationalFraction | Класс | Входная |
| numerator | Целое число | Входная |
| denominator | Целое число | Входная |
| add(RationalFraction) | Метод | Входная/выходная |
| toString() | Метод | Входная/выходная |
| Polynomial | Класс | Входная |
| add(Polynomial) | Метод | Входная |
| getCoefficients() | Метод | Входная |
| PolynomialArrayExample | Класс | Входная/выходная |
| n | Целое число | Входная. |
| polynomials | Класс | Входная |
| main(String[]) | Метод | Входная |
| sum | Целое число | Входная/выходная |

Uml диаграмма классов:



Код программы:

import java.util.Arrays;  
  
class RationalFraction {  
private int numerator;  
private int denominator;  
  
public RationalFraction(int numerator, int denominator) {  
this.numerator = numerator;  
this.denominator = denominator;  
}  
  
// Метод сложения рациональных дробей  
public RationalFraction add(RationalFraction other) {  
int resultNumerator = this.numerator \* other.denominator + other.numerator \* this.denominator;  
int resultDenominator = this.denominator \* other.denominator;  
return new RationalFraction(resultNumerator, resultDenominator);  
}  
  
@Override  
public String toString() {  
return numerator + "/" + denominator;  
}  
}  
  
class Polynomial {  
private RationalFraction[] coefficients;  
  
public Polynomial(RationalFraction[] coefficients) {  
this.coefficients = coefficients;  
}  
  
// Метод сложения полиномов  
public Polynomial add(Polynomial other) {  
int maxLength = Math.max(this.coefficients.length, other.coefficients.length);  
RationalFraction[] resultCoefficients = new RationalFraction[maxLength];  
  
for (int i = 0; i < maxLength; i++) {  
RationalFraction term1 = (i < this.coefficients.length) ? this.coefficients[i] : new RationalFraction(0, 1);  
RationalFraction term2 = (i < other.coefficients.length) ? other.coefficients[i] : new RationalFraction(0, 1);  
  
resultCoefficients[i] = term1.add(term2);  
}  
  
return new Polynomial(resultCoefficients);  
}  
  
public RationalFraction[] getCoefficients() {  
return coefficients;  
}  
}  
  
public class PolynomialArrayExample {  
public static void main(String[] args) {  
int n = 3; // Количество полиномов в массиве  
  
// Создание массива полиномов  
Polynomial[] polynomials = new Polynomial[n];  
for (int i = 0; i < n; i++) {  
// Инициализация каждого полинома   
RationalFraction[] coefficients = {  
new RationalFraction(1, 1),  
new RationalFraction(2, 1),  
new RationalFraction(3, 1)  
};  
polynomials[i] = new Polynomial(coefficients);  
}  
  
// Вычисление суммы полиномов в массиве  
Polynomial sum = Arrays.stream(polynomials)  
.reduce(new Polynomial(new RationalFraction[0]), Polynomial::add);  
  
// Вывод суммы полиномов  
System.out.println("Sum of polynomials:");  
for (RationalFraction coefficient : sum.getCoefficients()) {  
System.out.print(coefficient + "\t");  
}  
}  
}

Результат выполнения программы:

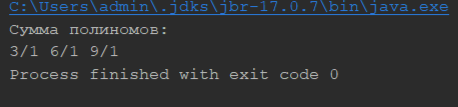


Рисунок 2 – Вывод программы 2

Задача 3

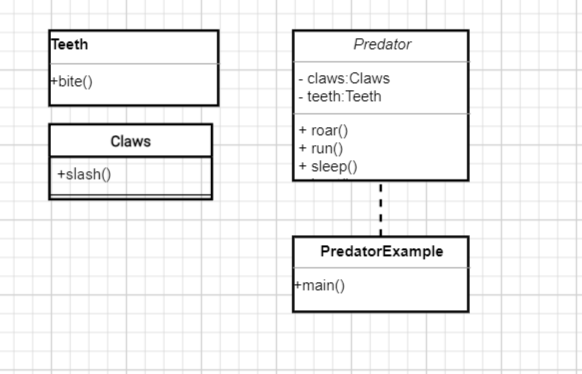
Описание задания:

Вариант 23. Создать объект класса Хищник, используя классы Когти, Зубы. Методы:  
рычать, бежать, спать, добывать пищу.

Таблица 3 – Таблица идентификаторов:

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Назначение |
| Claws | Класс | Входная |
| slash() | Метод | Входная |
| Teeth | Класс | Входная |
| bite() | Метод | Входная |
| roar() | Метод | Входная/Выходная |
| run() | Метод | Входная/Выходная |
| sleep() | Метод | Входная/Выходная |
| hunt() | Метод | Входная/Выходная |
| PredatorExample | Класс | Входная |
| main(String[]) | Метод | Входная |

Uml диаграмма классов:



Код программы:

class Claws {  
public void slash() {  
System.out.println("Когти царапают.");  
}  
}  
  
class Teeth {  
public void bite() {  
System.out.println("Зубы кусают.");  
}  
}  
  
class Predator {  
private Claws;  
private Teeth;  
  
public Predator(Claws, Teeth teeth) {  
this.claws = claws;  
this.teeth = teeth;  
}  
  
public void roar() {  
System.out.println("Хищник рычит.");  
}  
  
public void run() {  
System.out.println("Хищник бежит.");  
}  
  
public void sleep() {  
System.out.println("Хищник спит.");  
}  
  
public void hunt() {  
System.out.println("Хищник добывает пищу:");  
claws.slash();  
teeth.bite();  
}  
}  
  
public class PredatorExample {  
public static void main(String[] args) {  
// Создание объектов Когти и Зубы  
Claws = new Claws();  
Teeth teeth = new Teeth();  
  
// Создание объекта Хищник с использованием Когтей и Зубов  
Predator predator = new Predator(claws, teeth);  
  
// Вызов методов объекта   
predator.roar();  
predator.run();  
predator.sleep();  
predator.hunt();  
}  
}

Результат выполнения программы:

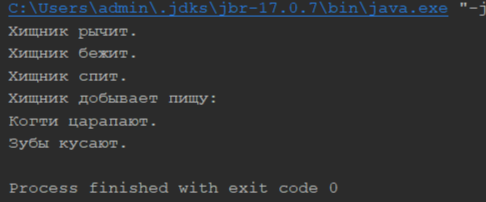


Рисунок 3 – Вывод программы 3

Задача 4

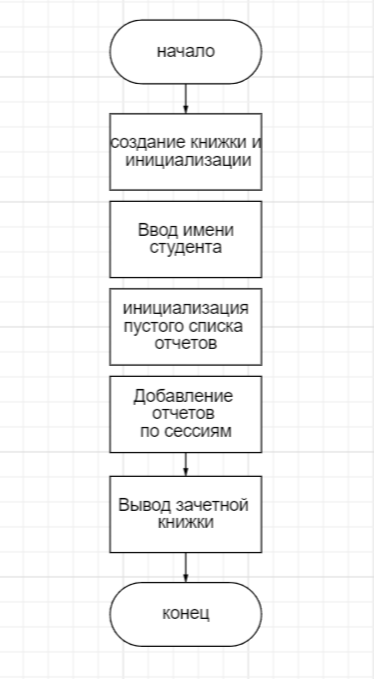
Описание задания:

Вариант 23. Создать класс Зачетная Книжка с внутренним классом, с помощью объектов которого можно хранить информацию о сессиях, зачетах, экзаменах

Таблица 4 – Таблица идентификаторов:

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | назначение |
| subject | String | Входная |
| result | String | Входная |
| getSubject() | Метод | Выходная |
| getResult() | Метод | Выходная |
| toString() | Метод | Выходная |
| Zaochety | Класс | Выходная |
| addZaOtchet(...) | Метод | Входная/ Выходная |
| getZaOtchets() | Метод | Выходная |
| studentName | String | Входная |
| addSessionInfo(...) | Метод | Входная/ Выходная |
| printZachetnayaKnizhka() | Метод | Входная |
| main(String[]) | Метод | Входная |

Блок-схема алгоритма:



Код программы:

import java.util.ArrayList;  
import java.util.List;  
  
class ZaOtchet {  
private String subject;  
private String result;  
  
public ZaOtchet(String subject, String result) {  
this.subject = subject;  
this.result = result;  
}  
  
public String getSubject() {  
return subject;  
}  
  
public String getResult() {  
return result;  
}  
  
@Override  
public String toString() {  
return "Предмет: " + subject + ", Результат: " + result;  
}  
}  
  
class Zaochety {  
private List<ZaOtchet> zaOtchets;  
  
public Zaochety() {  
this.zaOtchets = new ArrayList<>();  
}  
  
public void addZaOtchet(String subject, String result) {  
ZaOtchet = new ZaOtchet(subject, result);  
zaOtchets.add(zaOtchet);  
}  
  
public List<ZaOtchet> getZaOtchets() {  
return zaOtchets;  
}  
}  
  
public class ZachetnayaKnizhka {  
private String studentName;  
private Zaochety;  
  
public ZachetnayaKnizhka(String studentName) {  
this.studentName = studentName;  
this.zaochety = new Zaochety();  
}  
  
public void addSessionInfo(String subject, String result) {  
zaochety.addZaOtchet(subject, result);  
}  
  
public void printZachetnayaKnizhka() {  
System.out.println("Зачетная книжка для студента: " + studentName);  
List<ZaOtchet> zaOtchets = zaochety.getZaOtchets();  
for (ZaOtchet zaOtchet : zaOtchets) {  
System.out.println(zaOtchet);  
}  
}  
  
public static void main(String[] args) {  
ZachetnayaKnizhka studentBook = new ZachetnayaKnizhka("Иван Иванов");  
studentBook.addSessionInfo("Математика", "Зачет");  
studentBook.addSessionInfo("Физика", "Экзамен - отлично");  
studentBook.addSessionInfo("История", "Зачет");  
  
studentBook.printZachetnayaKnizhka();  
}  
}

import java.util.ArrayList;  
import java.util.List;  
  
class ZaOtchet {  
private String subject;  
private String result;  
  
public ZaOtchet(String subject, String result) {  
this.subject = subject;  
this.result = result;  
}  
  
public String getSubject() {  
return subject;  
}  
  
public String getResult() {  
return result;  
}  
  
@Override  
public String toString() {  
return "Предмет: " + subject + ", Результат: " + result;  
}  
}  
  
class Zaochety {  
private List<ZaOtchet> zaOtchets;  
  
public Zaochety() {  
this.zaOtchets = new ArrayList<>();  
}  
  
public void addZaOtchet(String subject, String result) {  
ZaOtchet = new ZaOtchet(subject, result);  
zaOtchets.add(zaOtchet);  
}  
  
public List<ZaOtchet> getZaOtchets() {  
return zaOtchets;  
}  
}  
  
public class ZachetnayaKnizhka {  
private String studentName;  
private Zaochety zaochety;  
  
public ZachetnayaKnizhka(String studentName) {  
this.studentName = studentName;  
this.zaochety = new Zaochety();  
}  
  
public void addSessionInfo(String subject, String result) {  
zaochety.addZaOtchet(subject, result);  
}  
  
public void printZachetnayaKnizhka() {  
System.out.println("Зачетная книжка для студента: " + studentName);  
List<ZaOtchet> zaOtchets = zaochety.getZaOtchets();  
for (ZaOtchet zaOtchet : zaOtchets) {  
System.out.println(zaOtchet);  
}  
}  
  
public static void main(String[] args) {  
ZachetnayaKnizhka studentBook = new ZachetnayaKnizhka("Иван Иванов");  
studentBook.addSessionInfo("Математика", "Зачет");  
studentBook.addSessionInfo("Физика", "Экзамен - отлично");  
studentBook.addSessionInfo("История", "Зачет");  
  
studentBook.printZachetnayaKnizhka();  
}  
}

Результат выполнения программы:

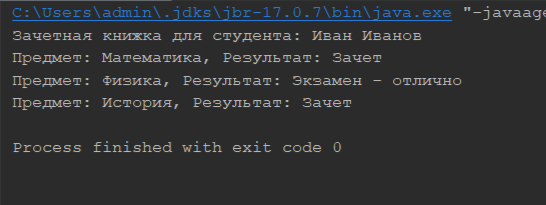


Рисунок 4 – Вывод программы 4

Задача 5

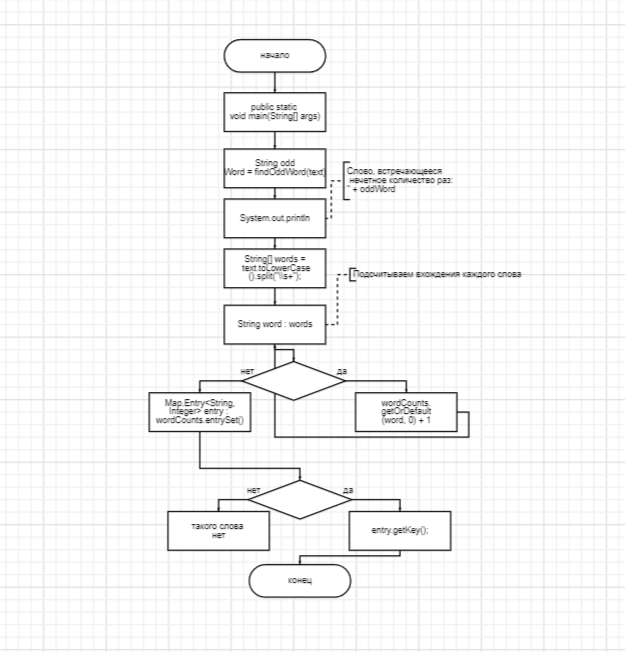
Описание задания:

Вариант – 23. все слова текста встречаются четное количество раз, за исключением одного. Определить это слово.

Таблица 5 – Таблица идентификаторов:

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Назначение |
| findOddWord(String) | Метод | Выходная |
| text | String | Входная |
| oddWord | String | Выходная |
| wordCounts | Map | Входная/Выходная |
| words | String[] | Входная. |
| word | String | Входная |
| entry | Map.Entry | Входная |

Блок-схема алгоритма:



Код программы:

import java.util.HashMap;  
import java.util.Map;  
  
public class Main {  
public static void main(String[] args) {  
String text = "Все слова в этом тексте встречаются четное количество раз, кроме одного. Определим это слово и узнаем, как часто оно встречается.";  
  
String oddWord = findOddWord(text);  
System.out.println("Слово, встречающееся нечетное количество раз: " + oddWord);  
}  
  
public static String findOddWord(String text) {  
Map<String, Integer> wordCounts = new HashMap<>();  
  
// Приводим текст к нижнему регистру и разбиваем на слова  
String[] words = text.toLowerCase().split("\\s+");  
  
// Подсчитываем вхождения каждого слова  
for (String word : words) {  
wordCounts.put(word, wordCounts.getOrDefault(word, 0) + 1);  
}  
  
// Ищем слово, встречающееся нечетное количество раз  
for (Map.Entry<String, Integer> entry : wordCounts.entrySet()) {  
if (entry.getValue() % 2 != 0) {  
return entry.getKey();  
}  
}  
  
return "Такого слова нет";  
}  
}

Результат выполнения программы:

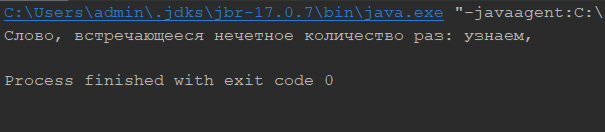


Рисунок 5 – Вывод программы 5

Задача 6

Описание задания:

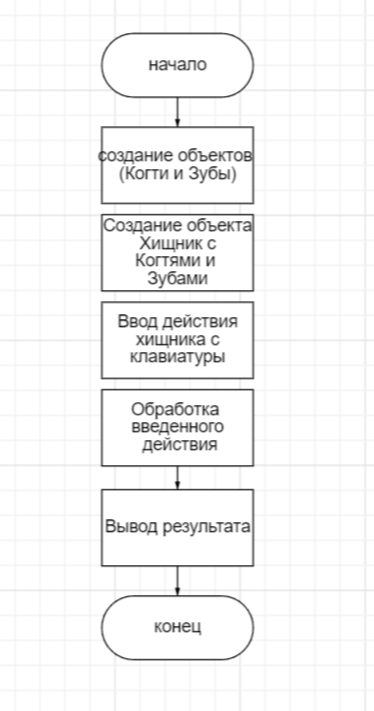
Выполнить задания на основе задачи № 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т. д.

Выполнить задания из задачи № 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Таблица 6 – Таблица идентификаторов:

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Назначение |
| Claws | Класс | Входная |
| Teeth | Класс | Входная |
| Predator(...) | Конструктор | Входная |
| scanner | Переменная | Входная |
| action | Переменная | Входная |

Блок-схема алгоритма:



Код программы:

import java.util.Scanner;  
  
class Claws {  
 public void slash() {  
 System.out.println("Когти царапают.");  
 }  
}  
  
class Teeth {  
 public void bite() {  
 System.out.println("Зубы кусают.");  
 }  
}  
  
class Predator {  
 Claws claws;  
 private Teeth teeth;  
  
 public Predator(Claws claws, Teeth teeth) {  
 this.claws = claws;  
 this.teeth = teeth;  
 }  
  
 public void roar() {  
 System.out.println("Хищник рычит.");  
 }  
  
 public void run() {  
 System.out.println("Хищник бежит.");  
 }  
  
 public void sleep() {  
 System.out.println("Хищник спит.");  
 }  
  
 public void hunt() {  
 System.out.println("Хищник добывает пищу:");  
 claws.slash();  
 teeth.bite();  
 }  
}  
  
public class PredatorExample {  
 public static void main(String[] args) {  
 try (Scanner scanner = new Scanner(System.in)) {  
 // Создание объектов Когти и Зубы  
 Claws claws = new Claws();  
 Teeth teeth = new Teeth();  
  
 // Создание объекта Хищник с использованием Когтей и Зубов  
 Predator predator = new Predator(claws, teeth);  
  
  
  
 // Добавим ввод действий хищника с клавиатуры  
 System.out.println("Введите действие хищника (roar, run, sleep, hunt):");  
 String action = scanner.next();  
  
 switch (action) {  
 case "roar":  
 predator.roar();  
 break;  
 case "run":  
 predator.run();  
 break;  
 case "sleep":  
 predator.sleep();  
 break;  
 case "hunt":  
 predator.hunt();  
 break;  
 default:  
 System.out.println("Некорректное действие.");  
 }  
 } catch (Exception e) {  
 System.err.println("Произошла ошибка: " + e.getMessage());  
 e.printStackTrace();  
 }  
 }  
}

Результат выполнения программы:

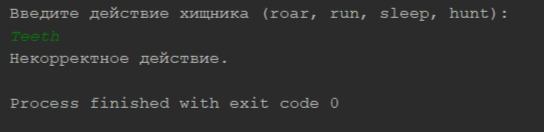


Рисунок 6 – Вывод программы 6

Задача 7

Описание задания:

Вариант 30. Составить программу получения на экране рисунков, изображенных в табл. 2.

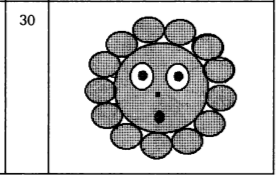
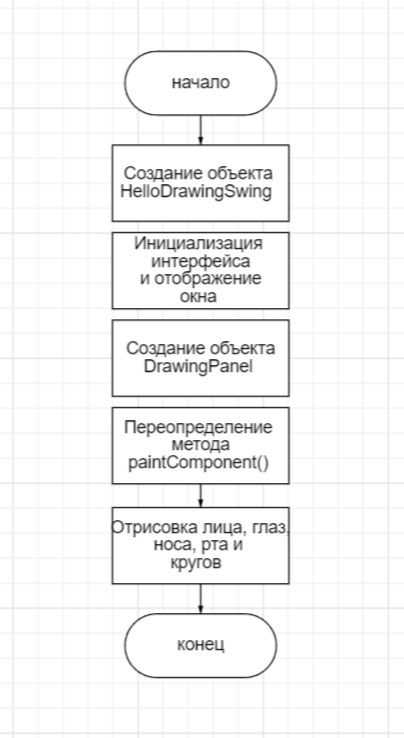


Таблица 7 – Таблица идентификаторов:

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Назначение |
| initUI() | Метод | Выходная |
| main(String[] args) | Метод | Входная /Выходная |
| paintComponent(Graphics g) | Метод | Входная/выходная |
| face | Ellipse2D | Выходная |
| leftEye | Ellipse2D | Выходная |
| rightEye | Ellipse2D | Выходная |
| leftPupil | Ellipse2D | Выходная |
| rightPupil | Ellipse2D | Выходная |
| nose | Rectangle2D | Выходная |
| mouth | Ellipse2D | Выходная |
| smallCircle | Ellipse2D | Выходная |

Блок-схема алгоритма:



Код программы

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.geom.Ellipse2D;  
import java.awt.geom.Rectangle2D;  
  
public class HelloDrawingSwing extends JFrame {  
  
public HelloDrawingSwing() {  
initUI();  
}  
  
private void initUI() {  
setTitle("Emoji Drawing");  
setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
setSize(300, 300);  
setLocationRelativeTo(null);  
  
DrawingPanel panel = new DrawingPanel();  
add(panel);  
  
setVisible(true);  
}  
  
public static void main(String[] args) {  
SwingUtilities.invokeLater(HelloDrawingSwing::new);  
}  
}  
  
class DrawingPanel extends JPanel {  
  
@Override  
protected void paintComponent(Graphics g) {  
super.paintComponent(g);  
Graphics2D g2d = (Graphics2D) g;  
  
// Создаем лицо  
Ellipse2D face = new Ellipse2D.Double(50, 50, 200, 200);  
g2d.setColor(Color.YELLOW);  
g2d.fill(face);  
  
// Создаем глаза  
Ellipse2D leftEye = new Ellipse2D.Double(90, 100, 40, 50);  
g2d.setColor(Color.WHITE);  
g2d.fill(leftEye);  
  
Ellipse2D rightEye = new Ellipse2D.Double(170, 100, 40, 50);  
g2d.fill(rightEye);  
  
// Создаем зрачки в глазах  
Ellipse2D leftPupil = new Ellipse2D.Double(110, 115, 16, 16);  
g2d.setColor(Color.BLACK);  
g2d.fill(leftPupil);  
  
Ellipse2D rightPupil = new Ellipse2D.Double(190, 115, 16, 16);  
g2d.fill(rightPupil);  
  
// Создаем нос  
Rectangle2D nose = new Rectangle2D.Double(145, 150, 10, 10);  
g2d.setColor(Color.ORANGE);  
g2d.fill(nose);  
  
// Создаем рот  
Ellipse2D mouth = new Ellipse2D.Double(140, 180, 20, 30);  
g2d.setColor(Color.BLACK);  
g2d.fill(mouth);  
  
// Создаем 13 кругов вокруг головы  
for (int i = 0; i < 13; i++) {  
double angle = Math.toRadians(360 / 13 \* i);  
double x = 150 + 120 \* Math.cos(angle) - 25;  
double y = 150 + 120 \* Math.sin(angle) - 25;  
  
Ellipse2D smallCircle = new Ellipse2D.Double(x, y, 50, 50);  
g2d.setColor(Color.RED);  
g2d.fill(smallCircle);  
}  
}  
}

Результат программы:

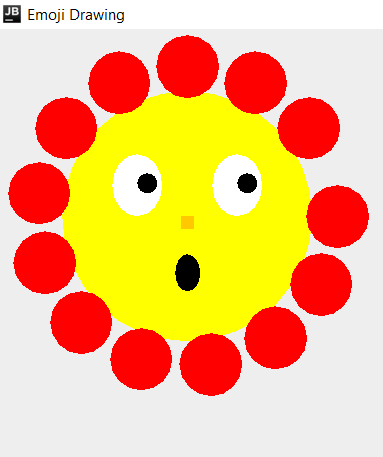


Рисунок 7 – Вывод программы 7

Задача 8

Описание задания:

Позволить щукам (см. задачу 21) поедать карпов, как только они их увидят, для чего следует установить, какого именно карпа видит щука. Проблема решается просмотром всей стаи карпов и определения того из них, чьи координаты близки к координатам данной щуки. Найденного карпа удалить из стаи.

Таблица 8 – Таблица идентификаторов:

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Назначение |
| x, y | double | Входная/выходная |
| speed | double | Входная/выходная |
| size | double | Входная/выходная |
| color | Color | Входная/выходная |
| direction | double | Входная/выходная |
| draw(Graphics2D g) | Метод | Входная/выходная |
| look(FishSchool fishSchool) | Метод | Входная |
| run() | Метод | Выходная |
| eat(FishSchool fishSchool) | Метод | Входная/выходная |
| isInRange(TFish otherFish) | Метод | Входная/выходная |
| addFish(TFish fish) | Метод | Входная/выходная |
| removeIf(Predicate<TFish> predicate) | Метод | Входная/выходная |
| iterator() | Метод | Выходная |
| setFishSchool(FishSchool fishSchool) | Метод | Входная/выходная |
| paintComponent(Graphics g) | Метод | Входная/выходная |

Uml диаграмма:



Код программы:

import javax.swing.\*;  
import java.awt.\*;  
import java.util.Iterator;  
import java.util.function.Predicate;  
  
abstract class TFish {  
double x, y;  
double speed;  
double size;  
Color color;  
double direction;  
TFish next;  
  
abstract void draw(Graphics2D g);  
  
abstract Color look(FishSchool fishSchool);  
  
abstract void run();  
  
void eat(FishSchool fishSchool) {  
Predicate<TFish> predicate = fish -> fish != this && isInRange(fish);  
Color targetColor = look(fishSchool);  
if (targetColor != null) {  
fishSchool.removeIf(predicate.and(fish -> fish.color.equals(targetColor)));  
System.out.println("Fish eaten!");  
}  
}  
  
boolean isInRange(TFish otherFish) {  
double distance = Math.sqrt(Math.pow(x - otherFish.x, 2) + Math.pow(y - otherFish.y, 2));  
return distance < size \* 2;  
}  
}  
  
class TPike extends TFish {  
public TPike(double x, double y, double speed, double size, double direction) {  
this.x = x;  
this.y = y;  
this.speed = speed;  
this.size = size;  
this.color = Color.GREEN;  
this.direction = direction;  
}  
  
@Override  
void draw(Graphics2D g) {  
g.setColor(color);  
g.drawLine((int) x, (int) y, (int) (x + size), (int) y);  
g.drawLine((int) (x + size), (int) y, (int) (x + size / 2), (int) (y - size));  
g.drawLine((int) (x + size / 2), (int) (y - size), (int) x, (int) y);  
}  
  
@Override  
Color look(FishSchool fishSchool) {  
for (TFish fish : fishSchool) {  
if (fish != this && isInRange(fish)) {  
return fish.color;  
}  
}  
return null;  
}  
  
@Override  
void run() {  
x += speed \* Math.cos(Math.toRadians(direction));  
y += speed \* Math.sin(Math.toRadians(direction));  
  
if (Math.random() < 0.1) {  
direction = Math.random() \* 360;  
}  
}  
}  
  
class TKarp extends TFish {  
public TKarp(double x, double y, double speed, double size, double direction) {  
this.x = x;  
this.y = y;  
this.speed = speed;  
this.size = size;  
this.color = Color.RED;  
this.direction = direction;  
}  
  
@Override  
void draw(Graphics2D g) {  
g.setColor(color);  
int[] xPoints = {(int) x, (int) (x + size), (int) (x - size)};  
int[] yPoints = {(int) y, (int) (y + size), (int) (y + size)};  
g.fillPolygon(xPoints, yPoints, 3);  
}  
  
@Override  
Color look(FishSchool fishSchool) {  
return null;  
}  
  
@Override  
void run() {  
x += speed \* Math.cos(Math.toRadians(direction));  
y += speed \* Math.sin(Math.toRadians(direction));  
  
if (Math.random() < 0.1) {  
direction = Math.random() \* 360;  
}  
}  
}  
  
class FishSchool implements Iterable<TFish> {  
TFish head;  
  
void addFish(TFish fish) {  
fish.next = head;  
head = fish;  
}  
  
void removeIf(Predicate<TFish> predicate) {  
TFish currentFish = head;  
TFish previousFish = null;  
  
while (currentFish != null) {  
if (predicate.test(currentFish)) {  
if (previousFish != null) {  
previousFish.next = currentFish.next;  
} else {  
head = currentFish.next;  
}  
return;  
}  
  
previousFish = currentFish;  
currentFish = currentFish.next;  
}  
}  
  
@Override  
public Iterator<TFish> iterator() {  
return new FishIterator();  
}  
  
private class FishIterator implements Iterator<TFish> {  
private TFish current = head;  
  
@Override  
public boolean hasNext() {  
return current != null;  
}  
  
@Override  
public TFish next() {  
TFish fish = current;  
current = current.next;  
return fish;  
}  
}  
}  
  
class AquariumPanel extends JPanel {  
private FishSchool fishSchool;  
  
public void setFishSchool(FishSchool fishSchool) {  
this.fishSchool = fishSchool;  
}  
  
@Override  
protected void paintComponent(Graphics g) {  
super.paintComponent(g);  
Graphics2D g2d = (Graphics2D) g;  
  
g2d.setColor(Color.CYAN);  
g2d.fillRect(0, 0, getWidth(), getHeight());  
  
for (TFish fish : fishSchool) {  
fish.draw(g2d);  
}  
}  
}  
  
public class simulation {  
public static void main(String[] args) {  
SwingUtilities.invokeLater(() -> {  
JFrame frame = new JFrame("Aquarium Simulation");  
frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
AquariumPanel aquariumPanel = new AquariumPanel();  
frame.getContentPane().add(aquariumPanel);  
frame.setSize(800, 600);  
frame.setVisible(true);  
  
FishSchool fishSchool = new FishSchool();  
for (int i = 0; i < 6; i++) {  
fishSchool.addFish(new TPike(Math.random() \* 800, Math.random() \* 600, 2, 20, Math.random() \* 360));  
}  
for (int i = 0; i < 10; i++) {  
fishSchool.addFish(new TKarp(Math.random() \* 800, Math.random() \* 600, 3, 30, Math.random() \* 360));  
}  
  
aquariumPanel.setFishSchool(fishSchool);  
  
Timer timer = new Timer(50, e -> {  
for (TFish fish : fishSchool) {  
fish.run();  
fish.eat(fishSchool);  
}  
aquariumPanel.repaint();  
});  
timer.start();  
});  
}  
}

Результат программы:

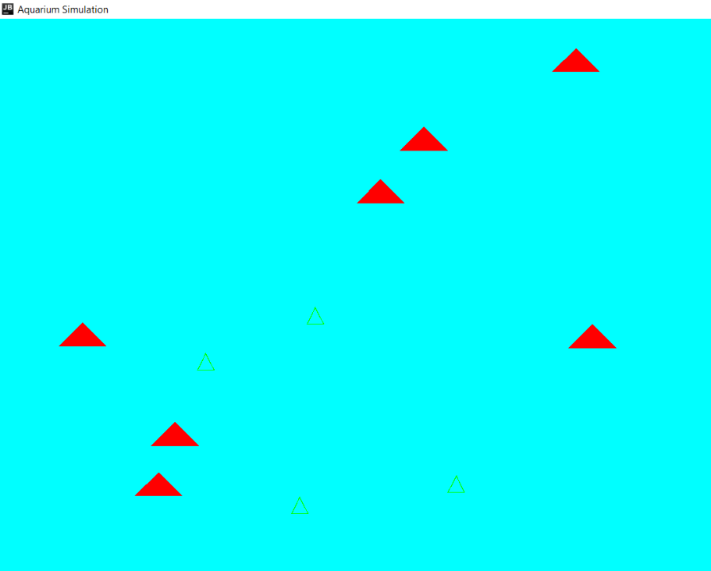


Рисунок 8 – Вывод программы 8

## 2.Разработка АИС “Расписание занятий”

Код программы :

import tkinter as tk  
from tkinter import ttk  
import sqlite3  
  
class ScheduleApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Расписание занятий")  
  
 # Создаем соединение с базой данных  
 self.conn = sqlite3.connect('расписание.db')  
 self.cursor = self.conn.cursor()  
  
 # Создаем таблицу для предметов  
 self.cursor.execute('''  
 CREATE TABLE IF NOT EXISTS предметы (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 название TEXT NOT NULL,  
 день\_недели TEXT NOT NULL,  
 аудитория TEXT NOT NULL  
 )  
 ''')  
  
 # Создаем таблицу для преподавателей  
 self.cursor.execute('''  
 CREATE TABLE IF NOT EXISTS преподаватели (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 ФИО TEXT NOT NULL,  
 предмет\_id INTEGER NOT NULL,  
 количество\_пар\_в\_неделю INTEGER NOT NULL,  
 количество\_студентов INTEGER NOT NULL,  
 FOREIGN KEY (предмет\_id) REFERENCES предметы (id)  
 )  
 ''')  
  
 # Commit изменений  
 self.conn.commit()  
  
 # Вкладка для предметов  
 self.tab\_subjects = ttk.Frame(root)  
 self.tab\_subjects.pack(expand=1, fill="both")  
 self.create\_subjects\_tab()  
  
 # Вкладка для преподавателей  
 self.tab\_teachers = ttk.Frame(root)  
 self.tab\_teachers.pack(expand=1, fill="both")  
 self.create\_teachers\_tab()  
  
 def create\_subjects\_tab(self):  
 frame\_subjects = ttk.Frame(self.tab\_subjects, padding="10")  
 frame\_subjects.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 ttk.Label(frame\_subjects, text="Название предмета:").grid(row=0, column=0, sticky=tk.W)  
 self.entry\_subject\_name = ttk.Entry(frame\_subjects)  
 self.entry\_subject\_name.grid(row=0, column=1, sticky=(tk.W, tk.E))  
  
 ttk.Label(frame\_subjects, text="День недели:").grid(row=1, column=0, sticky=tk.W)  
 self.entry\_subject\_day = ttk.Entry(frame\_subjects)  
 self.entry\_subject\_day.grid(row=1, column=1, sticky=(tk.W, tk.E))  
  
 ttk.Label(frame\_subjects, text="Аудитория:").grid(row=2, column=0, sticky=tk.W)  
 self.entry\_subject\_room = ttk.Entry(frame\_subjects)  
 self.entry\_subject\_room.grid(row=2, column=1, sticky=(tk.W, tk.E))  
  
 ttk.Button(frame\_subjects, text="Добавить предмет", command=self.add\_subject).grid(row=3, column=0, columnspan=2, pady=10)  
  
 # Создаем и отображаем Treeview для предметов  
 self.subjects\_tree = ttk.Treeview(frame\_subjects, columns=("ID", "Название", "День недели", "Аудитория"), show="headings")  
 self.subjects\_tree.heading("ID", text="ID")  
 self.subjects\_tree.heading("Название", text="Название")  
 self.subjects\_tree.heading("День недели", text="День недели")  
 self.subjects\_tree.heading("Аудитория", text="Аудитория")  
 self.subjects\_tree.grid(row=4, column=0, columnspan=2, pady=10)  
  
 # Обновляем Treeview  
 self.update\_subjects\_tree()  
  
 def create\_teachers\_tab(self):  
 frame\_teachers = ttk.Frame(self.tab\_teachers, padding="10")  
 frame\_teachers.grid(row=0, column=0, sticky=(tk.W, tk.E, tk.N, tk.S))  
  
 ttk.Label(frame\_teachers, text="ФИО преподавателя:").grid(row=0, column=0, sticky=tk.W)  
 self.entry\_teacher\_name = ttk.Entry(frame\_teachers)  
 self.entry\_teacher\_name.grid(row=0, column=1, sticky=(tk.W, tk.E))  
  
 ttk.Label(frame\_teachers, text="Название предмета:").grid(row=1, column=0, sticky=tk.W)  
 self.entry\_teacher\_subject = ttk.Entry(frame\_teachers)  
 self.entry\_teacher\_subject.grid(row=1, column=1, sticky=(tk.W, tk.E))  
  
 ttk.Label(frame\_teachers, text="Количество пар в неделю:").grid(row=2, column=0, sticky=tk.W)  
 self.entry\_teacher\_lectures = ttk.Entry(frame\_teachers)  
 self.entry\_teacher\_lectures.grid(row=2, column=1, sticky=(tk.W, tk.E))  
  
 ttk.Label(frame\_teachers, text="Количество студентов:").grid(row=3, column=0, sticky=tk.W)  
 self.entry\_teacher\_students = ttk.Entry(frame\_teachers)  
 self.entry\_teacher\_students.grid(row=3, column=1, sticky=(tk.W, tk.E))  
  
 ttk.Button(frame\_teachers, text="Добавить преподавателя", command=self.add\_teacher).grid(row=4, column=0, columnspan=2, pady=10)  
  
 # Создаем и отображаем Treeview для преподавателей  
 self.teachers\_tree = ttk.Treeview(frame\_teachers, columns=("ID", "ФИО", "Название предмета", "Количество пар в неделю", "Количество студентов"), show="headings")  
 self.teachers\_tree.heading("ID", text="ID")  
 self.teachers\_tree.heading("ФИО", text="ФИО")  
 self.teachers\_tree.heading("Название предмета", text="Название предмета")  
 self.teachers\_tree.heading("Количество пар в неделю", text="Количество пар в неделю")  
 self.teachers\_tree.heading("Количество студентов", text="Количество студентов")  
 self.teachers\_tree.grid(row=5, column=0, columnspan=2, pady=10)  
  
 # Обновляем Treeview  
 self.update\_teachers\_tree()  
  
 def add\_subject(self):  
 name = self.entry\_subject\_name.get()  
 day = self.entry\_subject\_day.get()  
 room = self.entry\_subject\_room.get()  
  
 if name and day and room:  
 self.cursor.execute('INSERT INTO предметы (название, день\_недели, аудитория) VALUES (?, ?, ?)', (name, day, room))  
 self.conn.commit()  
 self.update\_subjects\_tree()  
  
 def add\_teacher(self):  
 name = self.entry\_teacher\_name.get()  
 subject = self.entry\_teacher\_subject.get()  
 lectures = self.entry\_teacher\_lectures.get()  
 students = self.entry\_teacher\_students.get()  
  
 if name and subject and lectures and students:  
 # Получаем id предмета по его названию  
 self.cursor.execute('SELECT id FROM предметы WHERE название = ?', (subject.strip(),))  
 subject\_id = self.cursor.fetchone()  
  
 if subject\_id:  
 # Вставляем преподавателя в таблицу  
 self.cursor.execute('INSERT INTO преподаватели (ФИО, предмет\_id, количество\_пар\_в\_неделю, количество\_студентов) VALUES (?, ?, ?, ?)',  
 (name, subject\_id[0], lectures, students))  
 self.conn.commit()  
 self.update\_teachers\_tree()  
 else:  
 print(f"Error adding teacher: Subject '{subject}' not found.")  
 else:  
 print("Error adding teacher: Please fill in all fields.")  
  
 def update\_subjects\_tree(self):  
 # Очищаем Treeview  
 for row in self.subjects\_tree.get\_children():  
 self.subjects\_tree.delete(row)  
  
 # Выбираем данные из базы данных  
 self.cursor.execute('SELECT \* FROM предметы')  
 subjects = self.cursor.fetchall()  
  
 # Вставляем данные в Treeview  
 for subject in subjects:  
 self.subjects\_tree.insert("", "end", values=subject)  
  
 def update\_teachers\_tree(self):  
 # Очищаем Treeview  
 for row in self.teachers\_tree.get\_children():  
 self.teachers\_tree.delete(row)  
  
 # Выбираем данные из базы данных  
 self.cursor.execute('SELECT преподаватели.id, преподаватели.ФИО, предметы.название, преподаватели.количество\_пар\_в\_неделю, преподаватели.количество\_студентов FROM преподаватели JOIN предметы ON преподаватели.предмет\_id = предметы.id')  
 teachers = self.cursor.fetchall()  
  
 # Вставляем данные в Treeview  
 for teacher in teachers:  
 self.teachers\_tree.insert("", "end", values=teacher)  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = ScheduleApp(root)  
 root.mainloop()

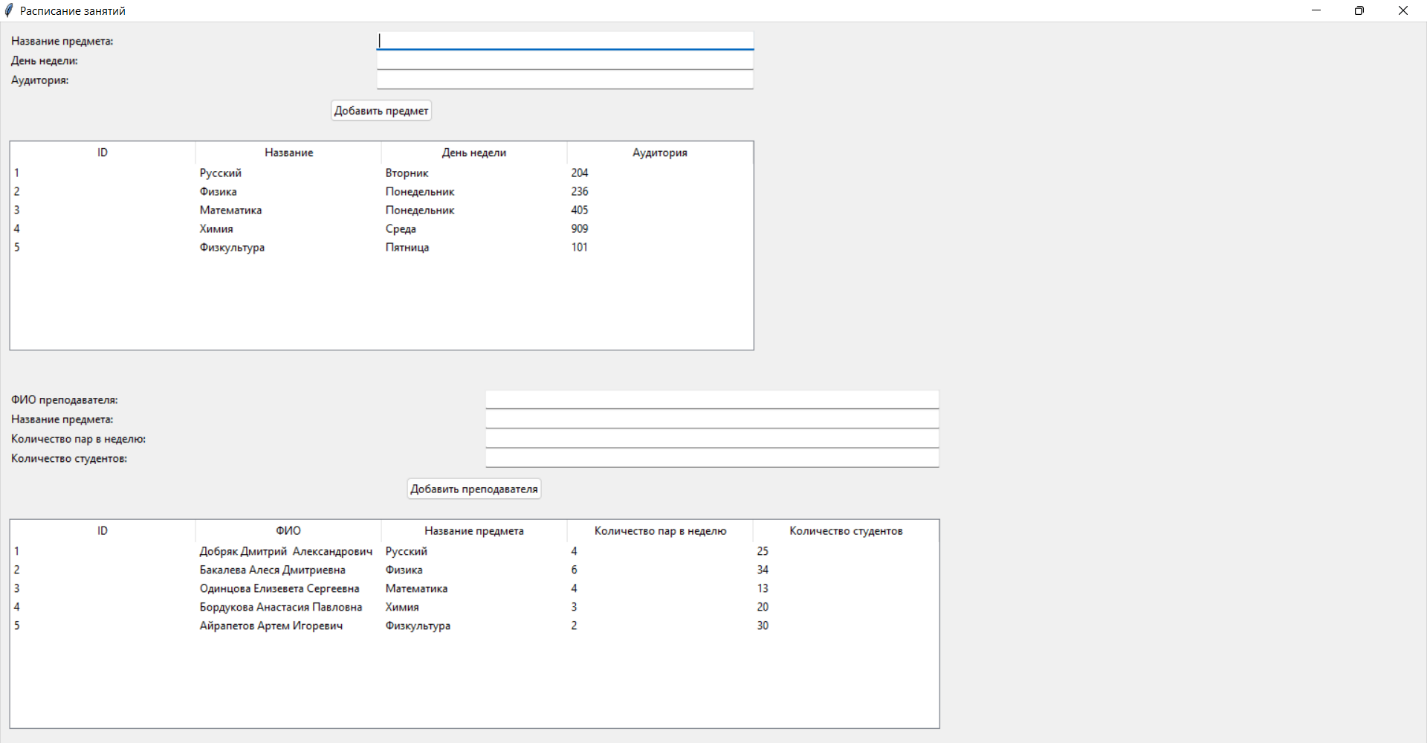
Результат:

Рисунок 9 – Расписание занятий

## 2.1 ER-диаграмма «сущность-связь»

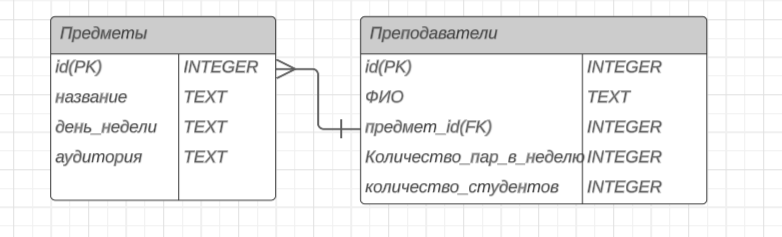


Рисунок 10 - ER-диаграмма «сущность-связь»

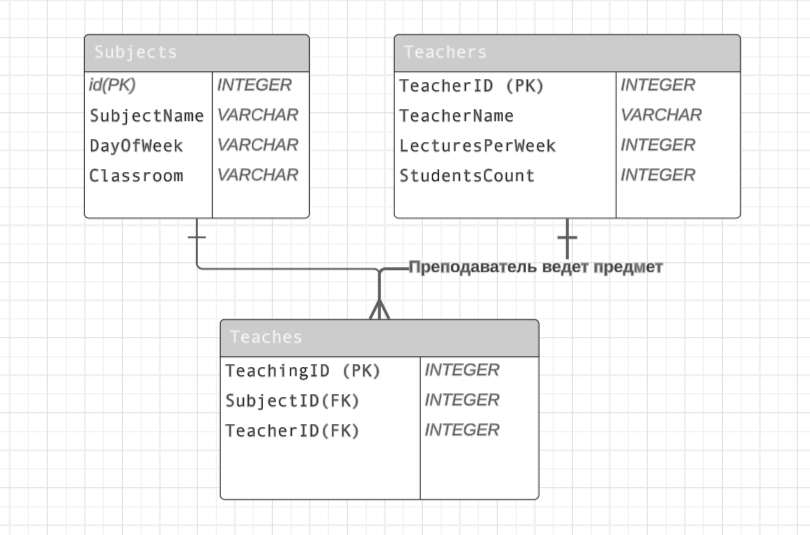


Рисунок 11 - ERD модель

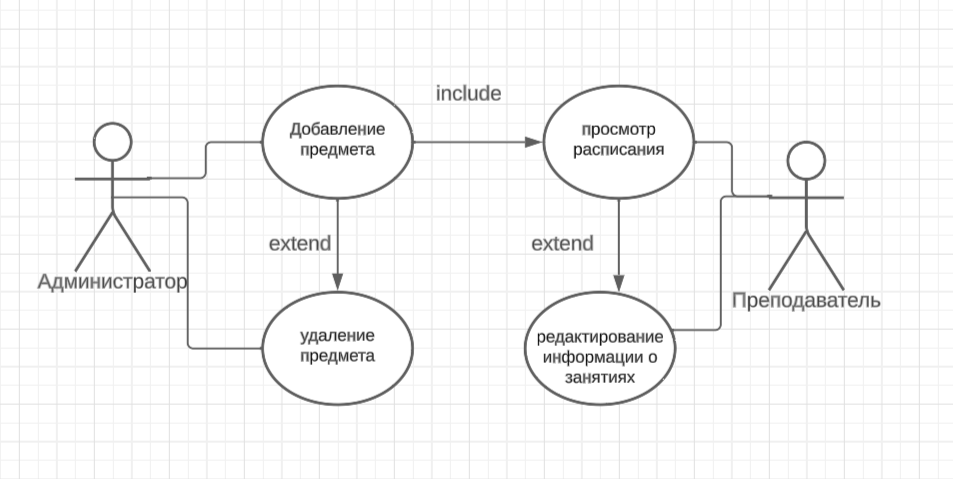


Рисунок 12 - Разработка диаграммы вариантов использования

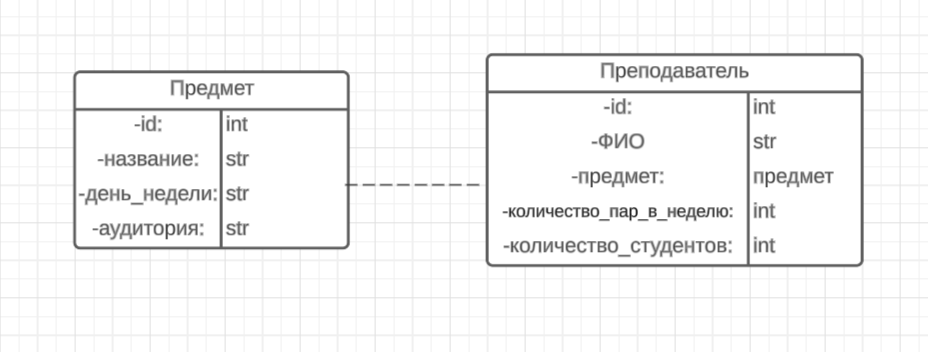


Рисунок 13 - Диаграмма классов

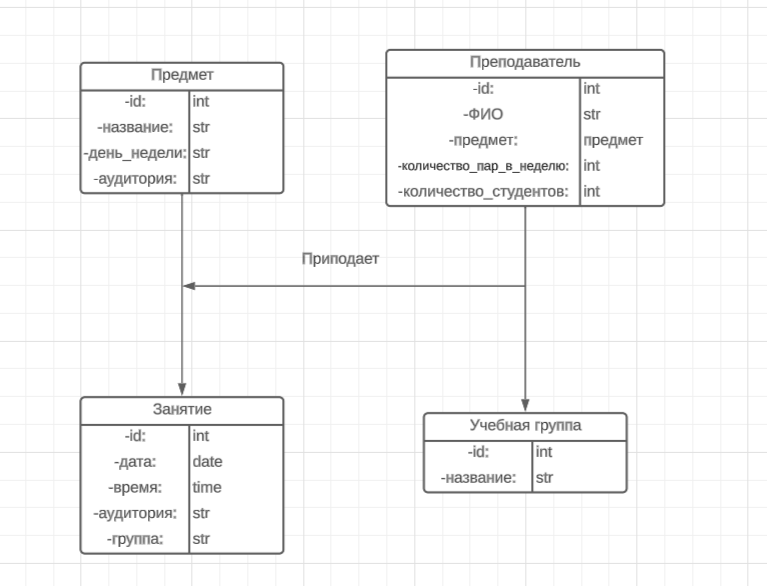


Рисунок 14 - Диаграмма коопераций

## 2.2 «Техническое задание»

1. Название проекта «Расписание занятий»

2. Цель проекта: Разработать программное приложение для учета расписания занятий, предметов и преподавателей.

3. Функциональные требования

3.1. Возможность добавления предметов:

- Название предмета

- День недели

- Аудитория

3.2. Возможность добавления преподавателей:

- ФИО преподавателя

- Название преподаваемого предмета

- Количество пар в неделю

- Количество студентов

3.3. Возможность просмотра добавленных предметов и преподавателей.

4. Нефункциональные требования

4.1. Интерфейс:

- Интуитивно понятный интерфейс для добавления и просмотра данных.

4.2. Безопасность:

- Данные о предметах и преподавателях должны храниться в безопасной базе данных.

- Аутентификация пользователя.

4.3. Производительность:

- Быстрый доступ к данным.

- Эффективное отображение расписания.

5. Требования к программной реализации

5.1. Язык программирования: Python.

5.2. GUI-фреймворк: Tkinter.

5.3. СУБД: SQLite.

6. Требования к документации

6.1. Руководство пользователя:

- Описание функциональности.

- Инструкции по использованию.

- Скриншоты интерфейса.

6.2. Техническое описание:

- Архитектура приложения.

- Описание базы данных.

7. План тестирования

7.1. Подготовка тестовых данных.

7.2. Тестирование добавления предметов и преподавателей.

7.3. Проверка отображения данных в Treeview.

7.4. Тестирование безопасности (аутентификация).

7.5. Тестирование производительности.

8. План развертывания

8.1. Инструкции по установке.

8.2. Конфигурация базы данных.

Краткая Спецификация Разрабатываемой Системы

Название Проекта: Расписание занятий

Описание: Разрабатываемая система представляет собой приложение для учета расписания занятий, предметов и преподавателей в образовательном учреждении.

Основные Функции:

* Добавление предметов с указанием названия, дня недели и аудитории.
* Добавление преподавателей с указанием ФИО, преподаваемого предмета, количества пар в неделю и количества студентов.
* Просмотр списка добавленных предметов и преподавателей.

Входные Данные:

Для добавления предмета:

* Название предмета (строка).
* День недели (строка).
* Аудитория (строка).

Для добавления преподавателя:

* ФИО преподавателя (строка).
* Название преподаваемого предмета (строка).
* Количество пар в неделю (целое число).
* Количество студентов (целое число).

Выходные Данные:

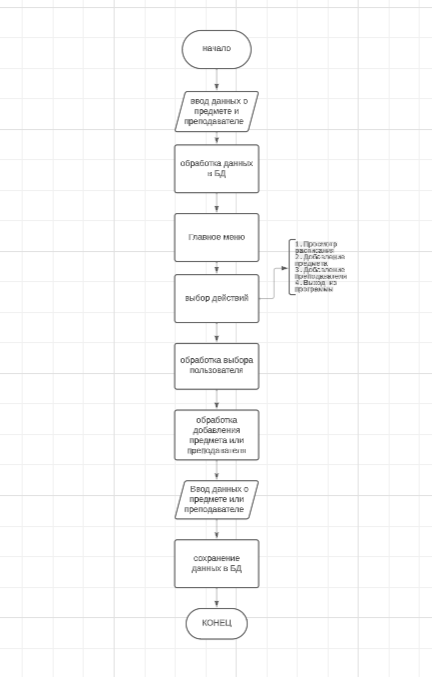
Список предметов:

* ID (уникальный идентификатор).
* Название предмета (строка).
* День недели (строка).
* Аудитория (строка).

Список преподавателей:

* ID (уникальный идентификатор).
* ФИО преподавателя (строка).
* Название предмета (строка).
* Количество пар в неделю (целое число).
* Количество студентов (целое число).

Основной Алгоритм Решения:



Шаг 1: Запуск приложения и открытие интерфейса.

Шаг 2: Выбор вкладки "Предметы" или "Преподаватели".

Шаг 3: Ввод необходимых данных для добавления предмета или преподавателя.

Шаг 4: Нажатие кнопки "Добавить".

Шаг 5: Обработка данных и добавление записи в базу данных.

Шаг 6: Обновление списка предметов или преподавателей в интерфейсе.

Пример Детализированного Алгоритма Добавления Предмета:

Шаг 1: Получение введенных данных из формы.

Шаг 2: Валидация данных (убедиться, что все поля заполнены корректно).

Шаг 3: Выполнение SQL-запроса для добавления предмета в базу данных.

Шаг 4: Обработка результата запроса (успешно или ошибка).

Шаг 5: Если успешно, обновление интерфейса списка предметов. Если ошибка, вывод сообщения об ошибке.

## 3.Руководство пользователя: Система Расписания

Введение

Добро пожаловать в систему расписания! Это приложение разработано для удобного управления расписанием предметов и преподавателей.

Содержание

* [Установка и запуск](https://chat.openai.com/c/1e16bfc5-59da-4a77-8769-5a66f8dbcd49#1-%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0-%D0%B8-%D0%B7%D0%B0%D0%BF%D1%83%D1%81%D0%BA)
* [Основные функции](https://chat.openai.com/c/1e16bfc5-59da-4a77-8769-5a66f8dbcd49#2-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8)
* [Добавление предмета](https://chat.openai.com/c/1e16bfc5-59da-4a77-8769-5a66f8dbcd49#21-%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BF%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%B0)
* [Добавление преподавателя](https://chat.openai.com/c/1e16bfc5-59da-4a77-8769-5a66f8dbcd49#22-%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BF%D1%80%D0%B5%D0%BF%D0%BE%D0%B4%D0%B0%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F)
* [Просмотр расписания](https://chat.openai.com/c/1e16bfc5-59da-4a77-8769-5a66f8dbcd49#23-%D0%BF%D1%80%D0%BE%D1%81%D0%BC%D0%BE%D1%82%D1%80-%D1%80%D0%B0%D1%81%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D1%8F)
* [Справочная система](https://chat.openai.com/c/1e16bfc5-59da-4a77-8769-5a66f8dbcd49#3-%D1%81%D0%BF%D1%80%D0%B0%D0%B2%D0%BE%D1%87%D0%BD%D0%B0%D1%8F-%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)

[Часто задаваемые вопросы (FAQ)](https://chat.openai.com/c/1e16bfc5-59da-4a77-8769-5a66f8dbcd49#4-%D1%87%D0%B0%D1%81%D1%82%D0%BE-%D0%B7%D0%B0%D0%B4%D0%B0%D0%B2%D0%B0%D0%B5%D0%BC%D1%8B%D0%B5-%D0%B2%D0%BE%D0%BF%D1%80%D0%BE%D1%81%D1%8B-faq)

1. Установка и запуск

* + Скачайте и установите Python, если у вас его нет: [Python официальный сайт](https://www.python.org/).
  + Скачайте исходный код системы расписания из репозитория.
  + Откройте терминал или командную строку и перейдите в каталог, содержащий файл main.py.
  + Выполните команду python main.py для запуска приложения.

2. Основные функции

2.1. Добавление предмета

Чтобы добавить новый предмет:

* Перейдите на вкладку "Предметы".
* Введите название предмета, выберите день недели и укажите аудиторию.
* Нажмите кнопку "Добавить предмет".

2.2. Добавление преподавателя

Чтобы добавить нового преподавателя:

* Перейдите на вкладку "Преподаватели".
* Введите ФИО преподавателя, выберите название предмета, количество пар в неделю и количество студентов.
* Нажмите кнопку "Добавить преподавателя".

2.3. Просмотр расписания

* Чтобы просмотреть расписание:
* Перейдите на соответствующую вкладку.
* Видите, расписание предметов или преподавателей в удобной таблице.

3. Справочная система

Для получения справки по функциям приложения:

Нажмите кнопку "Справка" в главном окне.

Выберите раздел, интересующий вас.

Тест-план для Расписания занятий:

Цель тестирования: Проверить правильность работы приложения для управления расписанием занятий.

Ресурсы для тестирования:

* Компьютер с установленным Python и библиотекой Tkinter.
* База данных SQLite с правильной структурой таблиц.

Сценарии тестирования:

Добавление предмета:

* Запустите приложение.
* В разделе "Предметы" введите данные нового предмета.
* Нажмите кнопку "Добавить предмет".
* Проверьте, что предмет добавлен в базу данных.

Добавление преподавателя:

* Запустите приложение.
* В разделе "Преподаватели" введите данные нового преподавателя.
* Нажмите кнопку "Добавить преподавателя".
* Проверьте, что преподаватель добавлен в базу данных.

Тестирование производительности:

Протестируйте приложение с большим количеством записей в базе данных, чтобы проверить производительность и стабильность.

## 4.Разработка сайта

На данном сайте присутствуют задания практики. На каждой странице представлено выполнение заданий, которые были на практике.

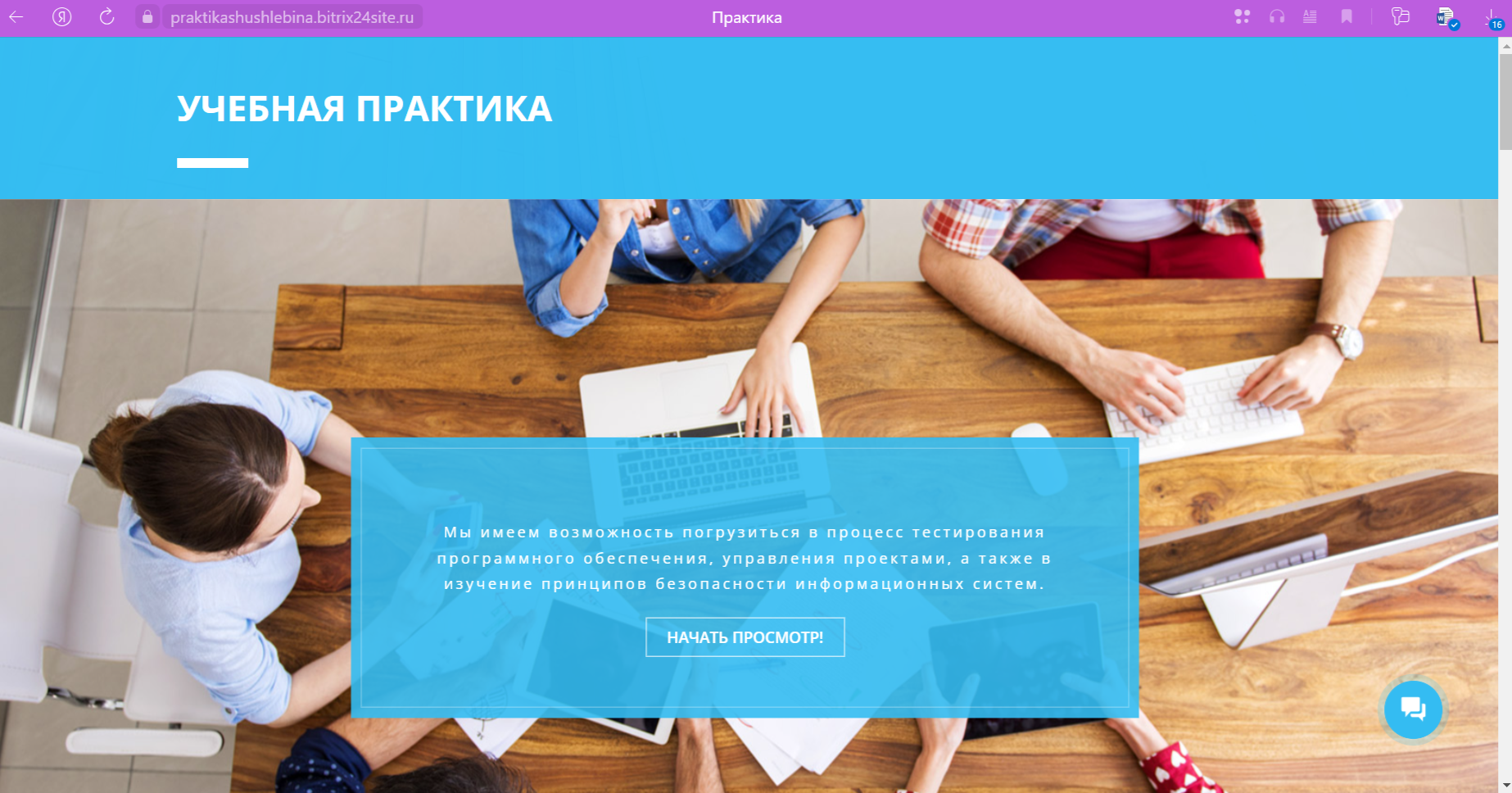


Рисунок 16 – 1 страница сайта

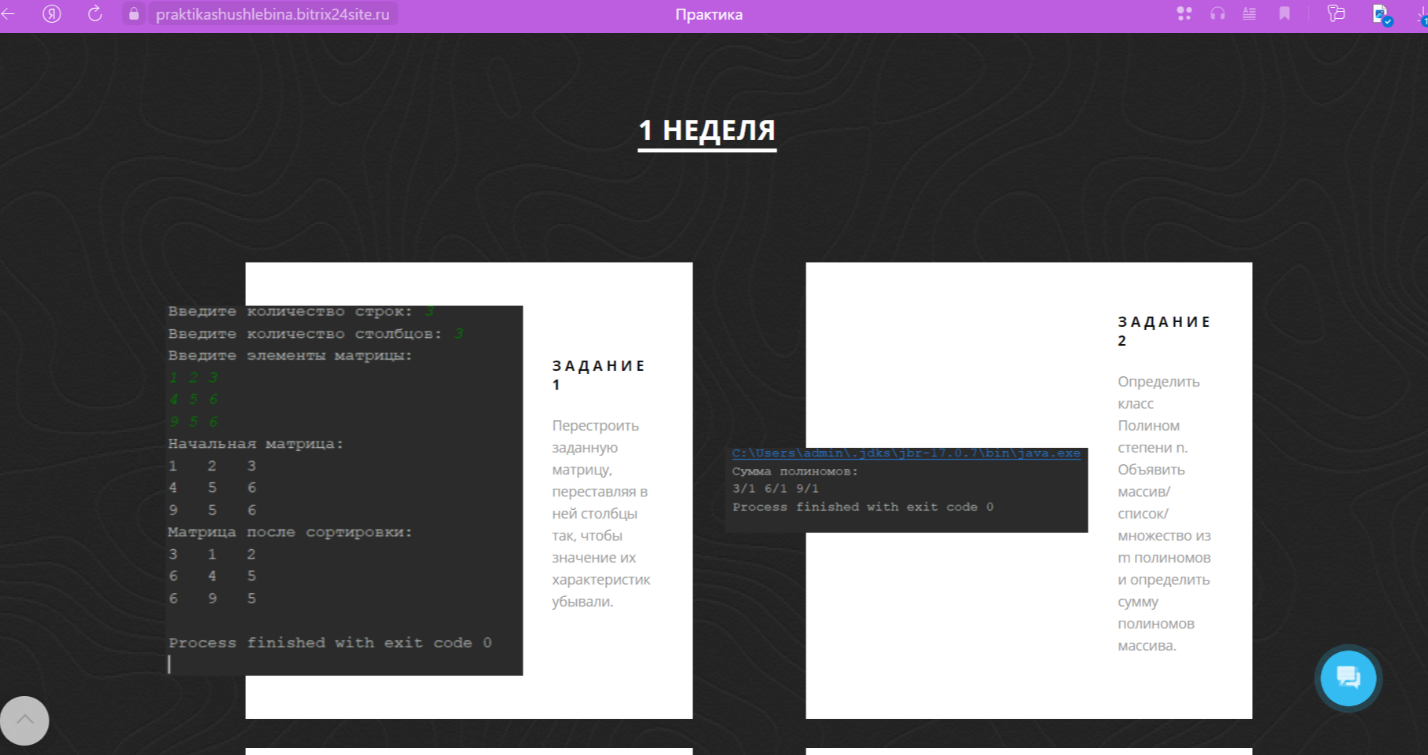


Рисунок 17 – 2 страница сайта

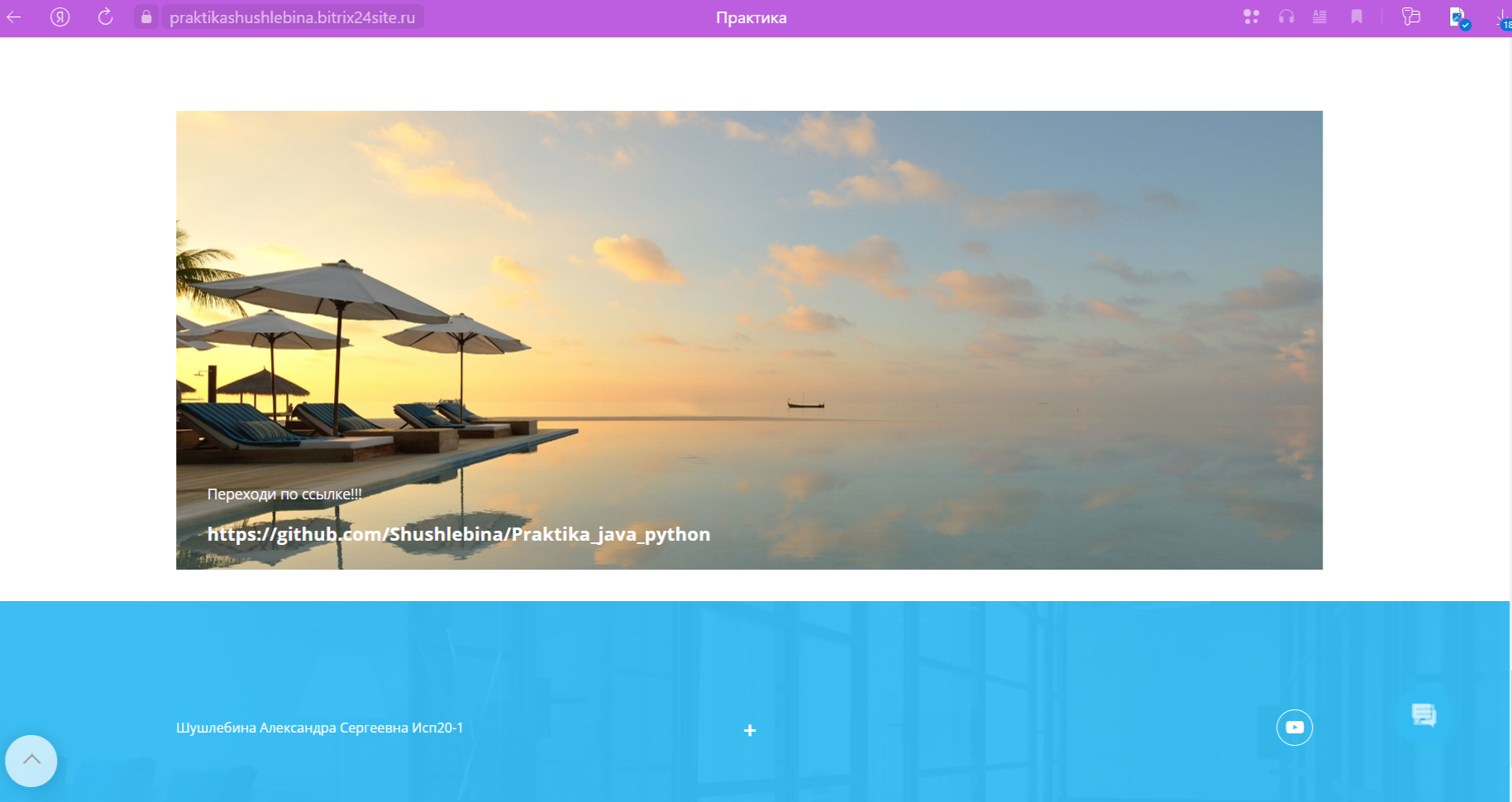


Рисунок 18 – Последняя страница сайта

Ссылка: [praktikashushlebina.bitrix24site.ru](https://praktikashushlebina.bitrix24site.ru/)

Ссылка на GitHub: https://github.com/Shushlebina/Praktika\_java\_python



Рисунок 19 – QR-код для мобильных устройств

## Заключение

Основные результаты практики включают:

Создание АИС Расписания занятий:

Разработана информационная система для эффективного управления расписанием учебных занятий.

Внедрены базы данных для хранения информации о предметах, преподавателях, аудиториях и других связанных данных.

Реализованы функции добавления, редактирования и удаления данных, обеспечивая удобство использования системы.

Рекомендации по дальнейшему совершенствованию:

Расширение Функциональности:

Рассмотреть возможность добавления дополнительных функций, таких как оповещения, автоматическое формирование расписания и интеграция с другими образовательными системами.

Обучение Пользователей:

Провести обучение персонала и конечных пользователей для максимального использования возможностей системы.

Тестирование безопасности:

Провести аудит безопасности для обеспечения защиты данных и предотвращения угроз.

Обратная связь Пользователей:

Организовать обратную связь с пользователями для получения их мнения и предложений по дальнейшему улучшению системы.

## Список использованных источников

1. Иванов П.С. Программирование модулей для информационных систем. – Санкт-Петербург : Техно Пресс, 2019. – 420 с. – ISBN 978-5-9775-1234-1.
2. Петрова Е.И. Разработка программного обеспечения: технологии и методы. – Москва : ТехноКнига, 2020. – 416 с. – ISBN 978-5-9963-6542-8.
3. Борисов А.М. Программирование на Python: от основ до продвинутого уровня. – Санкт-Петербург : БХВ-Петербург, 2021. – 832 с. – ISBN 978-5-97
4. Соколов Д.В. Моделирование систем с UML. – Москва : Техно Пресс, 2016. – 512 с. – ISBN 978-5-9775-2391-0.
5. DIRECTUM NX: Разработка и внедрение. Руководство разработчика. – Москва : DIRECTUM, 2018. – 380 с. – ISBN 978-5-93268-246-2.
6. Петров Н.А. Освоение Android: Практическое руководство. – Санкт-Петербург : Питер, 2016. – 240 с. – ISBN 978-5-496-00749-9.
7. Смирнова А.А. Дизайн веб-интерфейсов. – Москва : Издательство Логосфера, 2020. – 320 с. – ISBN 978-5-9909957-9-2.
8. Козлов А.С. Разработка интерфейсов: от идеи до реализации. – Санкт-Петербург : Питер, 2018. – 288 с. – ISBN 978-5-496-00749-7.
9. Джонсон М.А. ООП и разработка ПО: Принципы, паттерны, методики. – Москва : Техно Пресс, 2019. – 672 с. – ISBN 978-5-91474-174-9.
10. Кравец С. Г. Java: Современное программирование. – Москва : ДМК Пресс, 2020. – 752 с. – ISBN 978-5-97060-702-2.