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  Конопляник Дмитрий Андреевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2024

Содержание

[Министерство образования Калининградской области 1](#_Toc182467353)

[Введение 3](#_Toc182467354)

[1 Решение задач 4](#_Toc182467355)

[1.1 Задача 1 4](#_Toc182467356)

[1.2 Задача 2 7](#_Toc182467357)

[1.3 Задача 3 11](#_Toc182467358)

[1.4 Задача 4 16](#_Toc182467359)

[1.5 Задача 5 20](#_Toc182467360)

[1.6 Задача 6 22](#_Toc182467361)

[1.7 Задача 7 26](#_Toc182467362)

[1.8 Задача 8 26](#_Toc182467363)

[1.9 Задача 9 31](#_Toc182467364)

[1.10 Задача 1 35](#_Toc182467365)

[1.10 Задача 2 38](#_Toc182467366)

[2 Разработка АИС 40](#_Toc182467367)

[Техническое задание 41](#_Toc182467368)

[Краткая спецификация разрабатываемой системы 46](#_Toc182467369)

[4 Разработка сайта 55](#_Toc182467370)

[Заключение 56](#_Toc182467371)

[Список используемой литературы 57](#_Toc182467372)

Введение

Данный отчет по УП ПМ01, включающий в себя дисциплины: разработка программных модулей; разработка мобильных приложений; программирование 1С; поддержка и тестирование программных модулей; системное программирование.

Цели практики:

* Укрепить и углубить свои знания в изучаемых дисциплинах путем выполнения поставленного технического задания.
* Разработать автоматизированную информационную систему, предоставленную преподавателем практики.
* Провести тестирование разработанного программного модуля.
* Разработать мобильное приложение.
* Разработать сайт.
* Разработать и оформить руководство пользователя.

1. Решение задач
   1. Задача 1

11) Ввести с консоли n-размерность матрицы a [n] [n]. Задать значения элементов матрицы в интервале значений от -n до n с помощью датчика случайных чисел. Уплотнить матрицу, удаляя из нее строки и столбцы, заполненные нулями.

|  |  |  |
| --- | --- | --- |
| Название | Тип | Значение |
| n | int | Размерность матрицы |
| a | int[][] | Представление матрицы |
| random | Random | Генерация случайных чисел в матрице |
| min | int | Минимальное число из матрицы |
| max | int | Максимальное число из матрицы |
| i | int | Индекс строки |
| j | int | Индекс столбца |
| rows | int | Массив размером n, используемый для хранения индексов строк, не содержащих нулей |
| columns | int | Массив размером n, используемый для хранения индексов столбцов, не содержащих нулей |
| b | int[][] | уплотненная матрица, содержащая только строки и столбцы, не содержащие нулей |

import java.util.Random;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

int[][] a = new int[n][n];

Random random = new Random();

int min = -n;

int max = n;

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

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

a[i][j] = random.nextInt((max - min) + 1) + min;

}

}

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

printMatrix(a);

int[] rows = new int[n];

int[] columns = new int[n];

int rowCount = 0;

int columnCount = 0;

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

boolean rowHasZero = false;

boolean columnHasZero = false;

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

if (a[i][j] == 0) {

rowHasZero = true;

}

if (a[j][i] == 0) {

columnHasZero = true;

}

}

if (!rowHasZero) {

rows[rowCount] = i;

rowCount++;

}

if (!columnHasZero) {

columns[columnCount] = i;

columnCount++;

}

}

int[][] b = new int[rowCount][columnCount];

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

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

b[i][j] = a[rows[i]][columns[j]];

}

}

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

printMatrix(b);

}

public static void printMatrix(int[][] a) {

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

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

System.out.print(a[i][j] + "\t");

}

System.out.println();

}

}

}

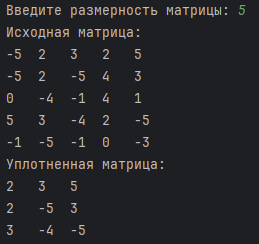


Рисунок 1 – Результат задачи 1

* 1. Задача 2

3) Реализовать методы сложения, вычитания, умножения и деления объектов (для

тех классов, объекты которых могут поддерживать арифметические действия). Определить класс Квадратное уравнение. Реализовать методы для поиска

корней, экстремумов, а также интервалов убывания/возрастания. Создать

массив/список/множество объектов и определить наибольшие и наименьшие по значению корни.

|  |  |  |
| --- | --- | --- |
| Название | Тип | Значение |
| q1 | QuadraticEquation | new QuadraticEquation(1, 2, 1) |
| q2 | QuadraticEquation | new QuadraticEquation(2, 3, 4) |
| q3 | QuadraticEquation | q1.add(q2) |
| q4 | QuadraticEquation | q1.subtract(q2) |
| q5 | QuadraticEquation | q1.multiply(q2) |
| q6 | QuadraticEquation | q1.divide(q2) |
| equations | List<QuadraticEquation> | new ArrayList<>(Arrays.asList(q1, q2, q3, q4, q5, q6)) |
| roots | List<Double> | new ArrayList<Double>() |
| equationRoots | double[] | результат вызова метода getRoots() для текущего объекта QuadraticEquation |
| root | Double | текущий корень уравнения, перебираемый в цикле |
| a | Double | коэффициент a уравнения q1 |
| b | Double | коэффициент b уравнения q1 |
| c | Double | коэффициент c уравнения q1 |
| d | Double | значение дискриминанта уравнения, вычисленное методом getDiscriminant() |
| x1 | Double | первый корень уравнения, вычисленный методом getRoots() |
| x2 | Double | второй корень уравнения, вычисленный методом getRoots() |

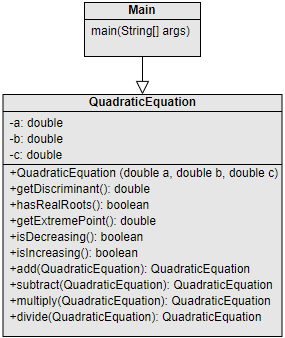


Рисунок 2 – Диаграмма классов задачи 2

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

public class Main {

public static void main(String[] args) {

QuadraticEquation q1 = new QuadraticEquation(1, 2, 1);

QuadraticEquation q2 = new QuadraticEquation(2, 3, 4);

QuadraticEquation q3 = q1.add(q2);

System.out.println("q1 + q2 = " + q3.a + "x^2 + " + q3.b + "x + " + q3.c);

QuadraticEquation q4 = q1.subtract(q2);

System.out.println("q1 - q2 = " + q4.a + "x^2 + " + q4.b + "x + " + q4.c);

QuadraticEquation q5 = q1.multiply(q2);

System.out.println("q1 \* q2 = " + q5.a + "x^2 + " + q5.b + "x + " + q5.c);

QuadraticEquation q6 = q1.divide(q2);

System.out.println("q1 / q2 = " + q6.a + "x^2 + " + q6.b + "x + " + q6.c);

List<QuadraticEquation> equations = new ArrayList<>();

equations.add(q1);

equations.add(q2);

equations.add(q3);

equations.add(q4);

equations.add(q5);

equations.add(q6);

List<Double> roots = new ArrayList<>();

for (QuadraticEquation equation : equations) {

double[] equationRoots = equation.getRoots();

for (double root : equationRoots) {

roots.add(root);

}

}

Collections.sort(roots);

System.out.println("Наименьший корень: " + roots.get(0));

System.out.println("Наибольший корень: " + roots.get(roots.size() - 1));

}

}

class QuadraticEquation {

double a;

double b;

double c;

public QuadraticEquation(double a, double b, double c) {

this.a = a;

this.b = b;

this.c = c;

}

public double getDiscriminant() {

return b \* b - 4 \* a \* c;

}

public boolean hasRealRoots() {

return getDiscriminant() >= 0;

}

public double[] getRoots() {

double d = getDiscriminant();

if (d < 0) {

return new double[0];

} else if (d == 0) {

double x = -b / (2 \* a);

return new double[]{x};

} else {

double x1 = (-b + Math.sqrt(d)) / (2 \* a);

double x2 = (-b - Math.sqrt(d)) / (2 \* a);

return new double[]{x1, x2};

}

}

public double getExtremePoint() {

return -b / (2 \* a);

}

public boolean isDecreasing() {

return a < 0;

}

public boolean isIncreasing() {

return a > 0;

}

public QuadraticEquation add(QuadraticEquation other) {

double newA = this.a + other.a;

double newB = this.b + other.b;

double newC = this.c + other.c;

return new QuadraticEquation(newA, newB, newC);

}

public QuadraticEquation subtract(QuadraticEquation other) {

double newA = this.a - other.a;

double newB = this.b - other.b;

double newC = this.c - other.c;

return new QuadraticEquation(newA, newB, newC);

}

public QuadraticEquation multiply(QuadraticEquation other) {

double newA = this.a \* other.a;

double newB = this.a \* other.b + this.b \* other.a;

double newC = this.a \* other.c + this.b \* other.b + this.c \* other.a;

return new QuadraticEquation(newA, newB, newC);

}

public QuadraticEquation divide(QuadraticEquation other) {

double newA = this.a / other.a;

double newB = (this.b \* other.a - this.a \* other.b) / (other.a \* other.a);

double newC = (this.c \* other.a - this.b \* other.b + this.a \* other.c) / (other.a \* other.a \* other.a);

return new QuadraticEquation(newA, newB, newC);

}

}

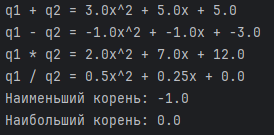


Рисунок 3 – Результат задачи 2

* 1. Задача 3

15) Создать приложение, удовлетворяющее требованиям, приведенным в задании. Наследование применять только в тех заданиях, в которых это логически обосновано. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString(). Создать объект класса Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей.

|  |  |  |
| --- | --- | --- |
| Название | Тип | Значение |
| house | House | новый экземпляр класса House, созданный в main методе |
| window1 | Window | новый экземпляр класса Window, добавленный в house через метод addWindow |
| window2 | Window | еще один новый экземпляр класса Window, добавленный в house через метод addWindow |
| door1 | Door | новый экземпляр класса Door, добавленный в house через метод addDoor |
| door2 | Door | еще один новый экземпляр класса Door, добавленный в house через метод addDoor |
| index | int | значение 0, переданное в метод lockDoor, чтобы заблокировать первую дверь в доме |
| windowCount | int | количество окон в доме, полученное вызовом метода getWindowCount |
| doorCount | int | количество дверей в доме, полученное вызовом метода getDoorCount |

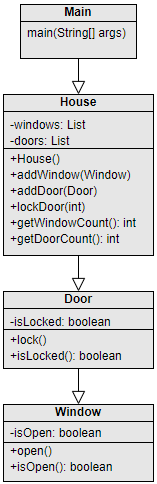


Рисунок 4 – Диаграмма классов задачи 3

import java.util.ArrayList;

import java.util.List;

class Door {

private boolean isLocked;

public void lock() {

isLocked = true;

}

public boolean isLocked() {

return isLocked;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Door door = (Door) obj;

return isLocked == door.isLocked;

}

@Override

public int hashCode() {

return isLocked ? 1 : 0;

}

@Override

public String toString() {

return "Door{" +

"isLocked=" + isLocked +

'}';

}

}

class Window {

private boolean isOpen;

public void open() {

isOpen = true;

}

public boolean isOpen() {

return isOpen;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Window window = (Window) obj;

return isOpen == window.isOpen;

}

@Override

public int hashCode() {

return isOpen ? 1 : 0;

}

@Override

public String toString() {

return "Window{" +

"isOpen=" + isOpen +

'}';

}

}

class House {

private List<Window> windows;

private List<Door> doors;

public House() {

windows = new ArrayList<>();

doors = new ArrayList<>();

}

public void addWindow(Window window) {

windows.add(window);

}

public void addDoor(Door door) {

doors.add(door);

}

public void lockDoor(int index) {

if (index >= 0 && index < doors.size()) {

doors.get(index).lock();

}

}

public int getWindowCount() {

return windows.size();

}

public int getDoorCount() {

return doors.size();

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

House house = (House) obj;

return windows.equals(house.windows) && doors.equals(house.doors);

}

@Override

public int hashCode() {

int result = windows.hashCode();

result = 31 \* result + doors.hashCode();

return result;

}

@Override

public String toString() {

return "House{" +

"windows=" + windows +

", doors=" + doors +

'}';

}

}

public class Main {

public static void main(String[] args) {

House house = new House();

house.addWindow(new Window());

house.addWindow(new Window());

house.addDoor(new Door());

house.addDoor(new Door());

house.lockDoor(0);

System.out.println("Количество окон: " + house.getWindowCount());

System.out.println("Количество дверей: " + house.getDoorCount());

System.out.println(house);

}

}

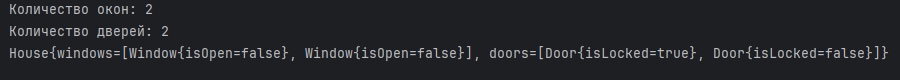


Рисунок 5 – Результат задачи 3

* 1. Задача 4

15) Создать класс Computer с внутренним классом, с помощью объектов которого можно хранить информацию об операционной системе, процессоре и оперативной памяти.

|  |  |  |
| --- | --- | --- |
| Название | Тип | Значение |
| winodws | String | новый экземпляр класса |
| intel | String | новый экземпляр класса |
| Ddr4 | String | новый экземпляр класса |
| pc | String | новый экземпляр класса |
| os | String | новый экземпляр класса |
| precessor | String | новый экземпляр класса |
| ram | String | новый экземпляр класса |

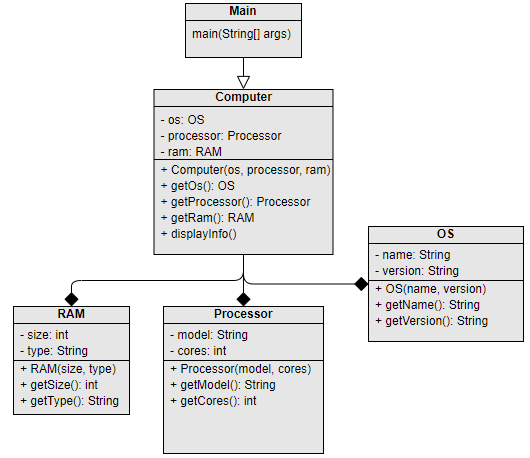


Рисунок 6 – Диаграмма классов задачи 4

// Внутренний класс для хранения информации об операционной системе

class OS {

private String name;

private String version;

public OS(String name, String version) {

this.name = name;

this.version = version;

}

public String getName() {

return name;

}

public String getVersion() {

return version;

}

}

// Внутренний класс для хранения информации о процессоре

class Processor {

private String model;

private int cores;

public Processor(String model, int cores) {

this.model = model;

this.cores = cores;

}

public String getModel() {

return model;

}

public int getCores() {

return cores;

}

}

// Внутренний класс для хранения информации об оперативной памяти

class RAM {

private int size;

private String type;

public RAM(int size, String type) {

this.size = size;

this.type = type;

}

public int getSize() {

return size;

}

public String getType() {

return type;

}

}

// Основной класс для хранения информации о компьютере

class Computer {

private OS os;

private Processor processor;

private RAM ram;

public Computer(OS os, Processor processor, RAM ram) {

this.os = os;

this.processor = processor;

this.ram = ram;

}

public OS getOs() {

return os;

}

public Processor getProcessor() {

return processor;

}

public RAM getRam() {

return ram;

}

public void displayInfo() {

System.out.println("Операционная система: " + os.getName() + " " + os.getVersion());

System.out.println("Процессор: " + processor.getModel() + ", ядер: " + processor.getCores());

System.out.println("Оперативная память: " + ram.getSize() + " " + ram.getType());

}

}

// Пример использования класса Computer

public class Main {

public static void main(String[] args) {

// Создаем объекты внутренних классов

OS windows = new OS("Windows", "10");

Processor intel = new Processor("Intel Core i7", 4);

RAM ddr4 = new RAM(16, "DDR4");

// Создаем объект основного класса Computer

Computer pc = new Computer(windows, intel, ddr4);

// Выводим информацию о компьютере

pc.displayInfo();

}

}

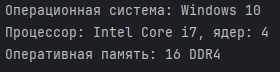


Рисунок 7 – Результат задачи 4

* 1. Задача 5

15) Напечатать квитанцию об оплате телеграммы, если стоимость одного слова задана.

|  |  |  |
| --- | --- | --- |
| Название | Тип | Значение |
| scanner | Scanner | новый экземпляр класса Scanner для считывания данных из консоли |
| wordCost | Double | стоимость одного слова, введенная пользователем |
| message | String | текст телеграммы, введенный пользователем |
| wordCount | Int | количество слов в тексте телеграммы, вычисленное методом countWords |
| totalCost | Double | итоговая стоимость телеграммы, вычисленная как произведение wordCount и wordCost |

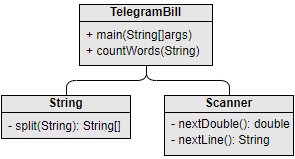


Рисунок 8 – Диаграмма классов задачи 5

import java.util.Scanner;

public class TelegramBill {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Введите стоимость одного слова: ");

double wordCost = scanner.nextDouble();

System.out.print("Введите текст телеграммы: ");

scanner.nextLine(); // consume newline left-over

String message = scanner.nextLine();

int wordCount = countWords(message);

double totalCost = wordCount \* wordCost;

System.out.printf("Квитанция об оплате телеграммы:%n%n");

System.out.printf("Текст телеграммы:%n%s%n", message);

System.out.printf("Стоимость одного слова: %.2f%n", wordCost);

System.out.printf("Количество слов: %d%n", wordCount);

System.out.printf("Итого к оплате: %.2f%n", totalCost);

}

private static int countWords(String message) {

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

return words.length;

}

}

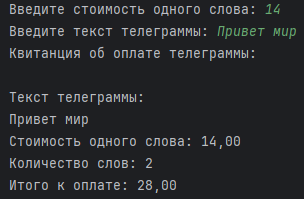


Рисунок 9 – Результат задачи 5

* 1. Задача 6

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.

|  |  |  |
| --- | --- | --- |
| Название | Тип | Значение |
| house | House | новый экземпляр класса House, созданный в main методе |
| window1 | Window | новый экземпляр класса Window, добавленный в house через метод addWindow |
| window2 | Window | еще один новый экземпляр класса Window, добавленный в house через метод addWindow |
| door1 | Door | новый экземпляр класса Door, добавленный в house через метод addDoor |
| door2 | Door | еще один новый экземпляр класса Door, добавленный в house через метод addDoor |
| index | int | значение 0, переданное в метод lockDoor, чтобы заблокировать первую дверь в доме |
| windowCount | int | количество окон в доме, полученное вызовом метода getWindowCount |
| doorCount | int | количество дверей в доме, полученное вызовом метода getDoorCount |

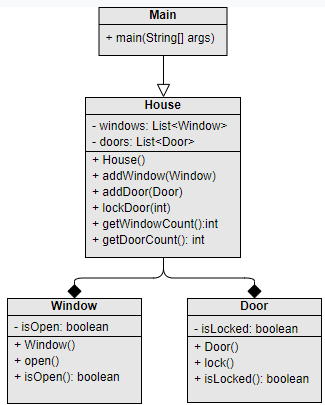


Рисунок 10 – Диаграмма классов задачи 6

import java.util.ArrayList;

import java.util.List;

class Door {

private boolean isLocked;

public void lock() {

isLocked = true;

}

public boolean isLocked() {

return isLocked;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Door door = (Door) obj;

return isLocked == door.isLocked;

}

@Override

public int hashCode() {

return isLocked ? 1 : 0;

}

@Override

public String toString() {

return "Door{" +

"isLocked=" + isLocked +

'}';

}

}

class Window {

private boolean isOpen;

public void open() {

isOpen = true;

}

public boolean isOpen() {

return isOpen;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Window window = (Window) obj;

return isOpen == window.isOpen;

}

@Override

public int hashCode() {

return isOpen ? 1 : 0;

}

@Override

public String toString() {

return "Window{" +

"isOpen=" + isOpen +

'}';

}

}

class House {

private List<Window> windows;

private List<Door> doors;

public House() {

windows = new ArrayList<>();

doors = new ArrayList<>();

}

public void addWindow(Window window) {

windows.add(window);

}

public void addDoor(Door door) {

doors.add(door);

}

public void lockDoor(int index) {

if (index >= 0 && index < doors.size()) {

doors.get(index).lock();

}

}

public int getWindowCount() {

return windows.size();

}

public int getDoorCount() {

return doors.size();

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

House house = (House) obj;

return windows.equals(house.windows) && doors.equals(house.doors);

}

@Override

public int hashCode() {

int result = windows.hashCode();

result = 31 \* result + doors.hashCode();

return result;

}

@Override

public String toString() {

return "House{" +

"windows=" + windows +

", doors=" + doors +

'}';

}

}

public class Main {

public static void main(String[] args) {

House house = new House();

house.addWindow(new Window());

house.addWindow(new Window());

house.addDoor(new Door());

house.addDoor(new Door());

house.lockDoor(0);

System.out.println("Количество окон: " + house.getWindowCount());

System.out.println("Количество дверей: " + house.getDoorCount());

System.out.println(house);

}

}

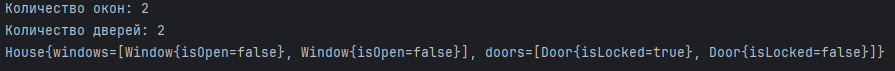
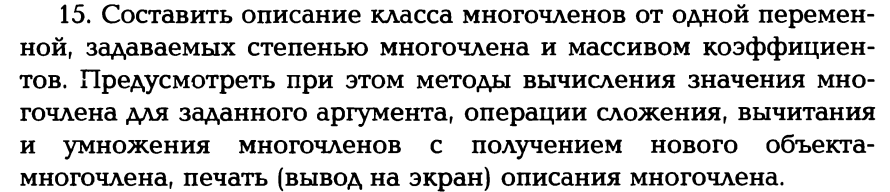


Рисунок 11 – Результат задачи 6

* 1. Задача 7
  2. Задача 8



|  |  |  |
| --- | --- | --- |
| Название | Тип | Значение |
| coefficients1 | int[] | {1, 2, 3} |
| coefficients2 | int[] | {4, 5, 6} |
| polynomial1 | Polynomial | new Polynomial(coefficients1) |
| polynomial2 | Polynomial | new Polynomial(coefficients2) |
| x | double | 2 |

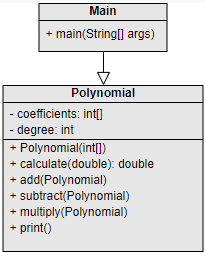


Рисунок 12 – Диаграмма классов задачи 8

class Polynomial {

private int[] coefficients;

private int degree;

public Polynomial(int[] coefficients) {

this.coefficients = coefficients;

this.degree = coefficients.length - 1;

}

public double calculate(double x) {

double result = 0;

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

result += coefficients[i] \* Math.pow(x, i);

}

return result;

}

public Polynomial add(Polynomial other) {

int maxDegree = Math.max(this.degree, other.degree);

int[] newCoefficients = new int[maxDegree + 1];

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

newCoefficients[i] += this.coefficients[i];

}

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

newCoefficients[i] += other.coefficients[i];

}

return new Polynomial(newCoefficients);

}

public Polynomial subtract(Polynomial other) {

int maxDegree = Math.max(this.degree, other.degree);

int[] newCoefficients = new int[maxDegree + 1];

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

newCoefficients[i] += this.coefficients[i];

}

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

newCoefficients[i] -= other.coefficients[i];

}

return new Polynomial(newCoefficients);

}

public Polynomial multiply(Polynomial other) {

int newDegree = this.degree + other.degree;

int[] newCoefficients = new int[newDegree + 1];

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

for (int j = 0; j <= other.degree; j++) {

newCoefficients[i + j] += this.coefficients[i] \* other.coefficients[j];

}

}

return new Polynomial(newCoefficients);

}

public void print() {

for (int i = degree; i >= 0; i--) {

if (coefficients[i] != 0) {

if (i != degree) {

if (coefficients[i] > 0) {

System.out.print(" + ");

} else {

System.out.print(" - ");

}

}

if (Math.abs(coefficients[i]) != 1 || i == 0) {

System.out.print(Math.abs(coefficients[i]));

}

if (i > 0) {

System.out.print("x");

if (i > 1) {

System.out.print("^" + i);

}

}

}

}

System.out.println();

}

}

public class Main {

public static void main(String[] args) {

int[] coefficients1 = {1, 2, 3};

Polynomial polynomial1 = new Polynomial(coefficients1);

System.out.print("Polynomial 1: ");

polynomial1.print();

int[] coefficients2 = {4, 5, 6};

Polynomial polynomial2 = new Polynomial(coefficients2);

System.out.print("Polynomial 2: ");

polynomial2.print();

System.out.print("Polynomial 1 + Polynomial 2: ");

polynomial1.add(polynomial2).print();

System.out.print("Polynomial 1 - Polynomial 2: ");

polynomial1.subtract(polynomial2).print();

System.out.print("Polynomial 1 \* Polynomial 2: ");

polynomial1.multiply(polynomial2).print();

System.out.println("Value of Polynomial 1 at x = 2: " + polynomial1.calculate(2));

}

}

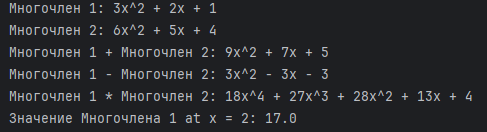


Рисунок 13 – Результат задачи 8

* 1. Задача 9



|  |  |  |
| --- | --- | --- |
| Название | Тип | Значение |
| SIZE | int | 5 |
| VALUES | int[][] | {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13} |
| SCORES | int[][] | Таблица оценок для каждой клетки в зависимости от числа, размещенного в ней |
| board | int[][] | Двумерный массив 5x5, представляющий игровое поле |
| numbers | List<Integer> | Список чисел от 1 до 13 в случайном порядке |
| playerScore | int | Количество очков, набранных игроком |
| computerScore | Int | Количество очков, набранных компьютером |
| i | Int | Индекс для перебора всех клеток в поле |
| number | Int | Выпавшее число в текущем ходу |
| x, y | Int | Координаты клетки, выбранной игроком или компьютером |
| computerX, computerY | Int | Координаты клетки, выбранной компьютером |
| maxScore | Int | Максимальное количество очков, которое можно получить за один ход |
| j, k | int | Вспомогательные индексы для перебора всех клеток в поле |
| scanner | Scanner | Объект Scanner для считывания ввода пользователя с клавиатуры |

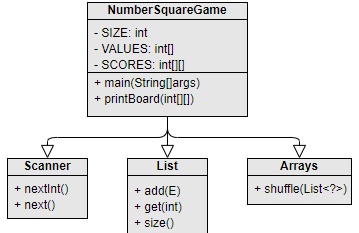


Рисунок 14 – Диаграмма классов задачи 8

import java.util.\*;

public class NumberSquareGame {

private static final int SIZE = 5;

private static final int[] VALUES = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};

private static final int[][] SCORES = {

{0, 0, 0, 0, 0},

{1, 1, 1, 1, 1},

{2, 2, 2, 2, 2},

{3, 3, 3, 3, 3},

{4, 4, 4, 4, 4},

{5, 5, 5, 5, 5},

{6, 6, 6, 6, 6},

{7, 7, 7, 7, 7},

{8, 8, 8, 8, 8},

{9, 9, 9, 9, 9},

{10, 10, 10, 10, 10},

{11, 11, 11, 11, 11},

{12, 12, 12, 12, 12},

{13, 13, 13, 13, 13}

};

public static void main(String[] args) {

int[][] board = new int[SIZE][SIZE];

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

for (int value : VALUES) {

numbers.add(value);

}

Collections.shuffle(numbers);

int playerScore = 0;

int computerScore = 0;

for (int i = 0; i < SIZE \* SIZE; i++) {

int number = numbers.get(i);

System.out.println("Выпало число: " + number);

// Игрок делает ход

System.out.print("Ваш ход (введите координаты x и y через пробел): ");

Scanner scanner = new Scanner(System.in);

int x = scanner.nextInt();

int y = scanner.nextInt();

board[x][y] = number;

playerScore += SCORES[number - 1][x] + SCORES[number - 1][y];

// Компьютер делает ход

int computerX = -1;

int computerY = -1;

int maxScore = Integer.MIN\_VALUE;

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

for (int k = 0; k < SIZE; k++) {

if (board[j][k] == 0) {

int score = SCORES[number - 1][j] + SCORES[number - 1][k];

if (score > maxScore) {

maxScore = score;

computerX = j;

computerY = k;

}

}

}

}

board[computerX][computerY] = number;

computerScore += maxScore;

// Выводим текущее состояние доски

printBoard(board);

System.out.println("Ваши очки: " + playerScore);

System.out.println("Очки компьютера: " + computerScore);

}

if (playerScore > computerScore) {

System.out.println("Вы выиграли!");

} else if (computerScore > playerScore) {

System.out.println("Компьютер выиграл!");

} else {

System.out.println("Ничья!");

}

}

private static void printBoard(int[][] board) {

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

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

System.out.print(board[i][j] + " ");

}

System.out.println();

}

}

}

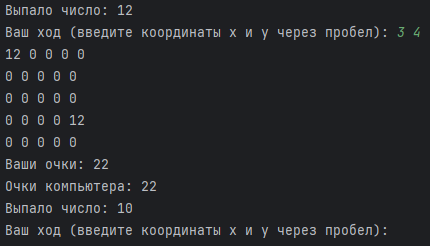


Рисунок 15 – Результат задачи 9

* 1. Задача 1

section .data

N dd 5

result1 db 'Result (N \* 2): ', 0

result2 db 'Result (N / 2): ', 0

newline db 10, 0

section .bss

buf resb 10

section .text

global \_start

\_start:

mov eax, [N]

shl eax, 1

mov [N], eax

mov edi, buf

call int\_to\_string

mov eax, 1

mov edi, 1

mov rsi, result1

mov edx, 17

syscall

mov eax, 1

mov edi, 1

mov rsi, buf

mov edx, 10

syscall

mov eax, 1

mov edi, 1

mov rsi, newline

mov edx, 1

syscall

mov eax, [N]

shr eax, 1

mov [N], eax

mov edi, buf

call int\_to\_string

mov eax, 1

mov edi, 1

mov rsi, result2

mov edx, 17

syscall

mov eax, 1

mov edi, 1

mov rsi, buf

mov edx, 10

syscall

mov eax, 1

mov edi, 1

mov rsi, newline

mov edx, 1

syscall

mov eax, 60

xor edi, edi

syscall

int\_to\_string:

mov ecx, 10

xor ebx, ebx

convert\_loop:

xor edx, edx

div ecx

add dl, '0'

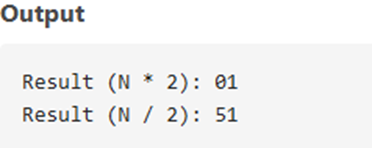
mov [edi], dl

inc edi

test eax, eax

jnz convert\_loop

ret



1.10 Задача 2

section .data

msg db "Result: ", 0

res db "00", 0

section .text

global \_start

\_start:

mov rdi, 3

mov rsi, 4

push rdi

push rsi

call Padd

pop rdi

add rsp, 8

mov rax, rdi

mov rdi, res

call int\_to\_str

mov rax, 1

mov rdi, 1

mov rsi, msg

mov rdx, 8

syscall

mov rax, 1

mov rdi, 1

mov rsi, res

mov rdx, 2

syscall

mov rax, 60

xor rdi, rdi

syscall

Padd:

push rbp

mov rbp, rsp

mov rax, [rbp+16]

add rax, [rbp+24]

mov [rbp+16], rax

pop rbp

ret

int\_to\_str:

mov rcx, 10

mov rbx, rdi

add rbx, 1

convert\_loop:

xor rdx, rdx

div rcx

add dl, '0'

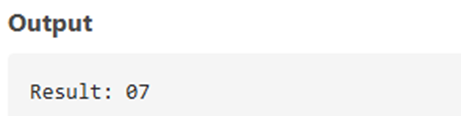
mov [rbx], dl

dec rbx

test rax, rax

jnz convert\_loop

ret



1. Разработка АИС

Вариант 15 - Ресторан

1. Анализ предметной области. Разработка ER-диаграммы «сущность-связь» (ErWin). Обязательна 3 нормальная форма с обеспечением ссылочной целостности. При разработке диаграммы обратите внимание на согласованную осмысленную схему именования, создайте необходимые первичные и внешние ключи, определите ограничения внешних ключей, отражающие характер предметной области.
2. ER - диаграмма должна быть представлена в формате удобном для просмотра и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь) проведение анализа поставленной задачи и проектирования базы данных (ERD модели) с применением case-средств;

Создайте все необходимые сущности, определите отношения, создайте ограничения на связи между сущностями (при наличии всех связей), приведите базу данных к 3НФ (при наличии всех сущностей и связей).

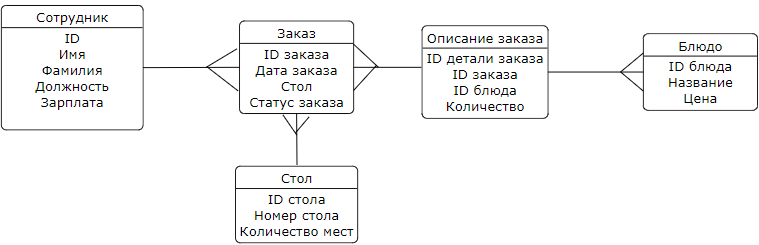


Рисунок 16 – ER-диаграмма

Задача 5

Техническое задание

Введение

Целью настоящего технического задания является определение требований к разработке автоматизированной информационной системы (АИС) для ресторана. АИС будет обеспечивать комплексное решение для управления операциями ресторана, включая управление меню, прием заказов, управление клиентами и управление бронированием.

Функциональные требования

1. Управление меню

* АИС должна позволять администраторам ресторана создавать, редактировать и удалять меню.
* АИС должна позволять администраторам ресторана добавлять, редактировать и удалять блюда из меню.
* АИС должна отображать пункты меню с ценами и описаниями.

1. Прием заказов

* АИС должна позволять клиентам делать заказы онлайн или через мобильное приложение.
* АИС должна позволять персоналу ресторана принимать заказы вручную.
* АИС должна генерировать номера заказов и отслеживать статус заказов.
* АИС должна рассчитывать общую сумму заказов и отображать резюме заказов.

1. Управление клиентами

* АИС должна позволять клиентам создавать аккаунты и авторизоваться для просмотра истории заказов и бронирования.
* АИС должна позволять администраторам ресторана просматривать информацию о клиентах и историю заказов.

1. Управление бронированием

* АИС должна позволять клиентам бронировать столы онлайн или через мобильное приложение.
* АИС должна позволять персоналу ресторана управлять бронированием и назначать столы.
* АИС должна отображать расписание бронирования и доступность столов.

1. Отчетность и аналитика

* АИС должна генерировать отчеты о продажах, поведении клиентов и производительности ресторана.
* АИС должна обеспечивать аналитику и инсайты для помощи администраторам ресторана в принятии обоснованных решений.

Нефункциональные требования

1. Безопасность

* АИС должна обеспечивать конфиденциальность, целостность и доступность данных клиентов и операций ресторана.
* АИС должна соответствовать требованиям законодательства о защите данных.

1. Простота использования

* АИС должна обеспечивать интуитивно понятный и пользовательский интерфейс для клиентов и персонала ресторана.
* АИС должна быть доступна на различных устройствах, включая компьютеры, ноутбуки, планшеты и мобильные телефоны.

1. Производительность

* АИС должна отвечать на ввод пользователя в течение2 секунд.
* АИС должна обрабатывать не менее100 одновременных пользователей.

1. Масштабируемость

* АИС должна быть спроектирована для масштабирования для обеспечения растущего трафика и объема данных.

1. Интеграция

* АИС должна интегрироваться с существующими системами ресторана, включая системы управления точками продаж и платежные шлюзы.

Архитектура системы

АИС будет разработана с использованием модульной архитектуры, состоящей из следующих компонентов:

1. Презентационный слой

* Веб-приложение для клиентов и персонала ресторана.
* Мобильное приложение для клиентов.

1. Слой приложения

* Бизнес-логика и правила для управления меню, приемом заказов, управлением клиентами и управлением бронированием.

1. Слой данных

* Система управления базами данных для хранения данных ресторана, включая меню, заказы, клиентов и бронирование.

1. Слой интеграции

* API для интеграции с существующими системами ресторана, включая системы управления точками продаж и платежные шлюзы.

Технические требования

1. Языки программирования

* Java или Python для слоя приложения.
* JavaScript для презентационного слоя.

1. Система управления базами данных

* Реляционная система управления базами данных, такая как MySQL или PostgreSQL.

1. Операционная система

* Windows

1. Веб-сервер

* Apache или Nginx.

1. Фреймворк для мобильных приложений

* React Native или Flutter.

Тестирование и обеспечение качества

АИС будет подвергнута тщательному тестированию и обеспечению качества для обеспечения соответствия функциональным и нефункциональным требованиям, изложенным в настоящем техническом задании.

Сроки реализации

Реализация АИС будет завершена в течение6 месяцев с даты начала проекта.

Бюджет

Бюджет на разработку АИС составит [$X], который включает в себя все затраты, связанные с разработкой, тестированием и развертыванием.

Заключение

АИС для ресторана будет обеспечивать комплексное решение для управления операциями ресторана, включая управление меню, прием заказов, управление клиентами и управление бронированием.

Задача 6

Краткая спецификация разрабатываемой системы

Система: Автоматизированная информационная система (АИС) для ресторанаЦель: Обеспечить комплексное решение для управления операциями ресторана, включая управление меню, прием заказов, управление клиентами и управление бронированием.

Входные данные:

Информация о меню (названия блюд, цены, описания)

Заказы клиентов (онлайн и офлайн)

Информация о клиентах (логин, пароль, история заказов)

Бронирование столов (дата, время, количество гостей)

Информация о персонале ресторана (логин, пароль, права доступа)

Выходные данные:

Меню с ценами и описаниями\* Номера заказов и статус заказов\* Резюме заказов (общая сумма, список блюд)

Информация о клиентах (история заказов, бронирование)

Расписание бронирования и доступность столов\* Отчеты о продажах, поведении клиентов и производительности ресторана Основной алгоритм решения учета в виде укрупненной блок-схемы:

Код АИС:

package org.example;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class Main extends JFrame {

private JTable employeeTable;

private JTable orderTable;

private JTable orderDetailsTable;

private JTable dishTable;

private JTable tableTable;

private DefaultTableModel employeeTableModel;

private DefaultTableModel orderTableModel;

private DefaultTableModel orderDetailsTableModel;

private DefaultTableModel dishTableModel;

private DefaultTableModel tableTableModel;

private Connection connection;

public Main() {

super("База данных ресторана");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(800, 600);

setLocationRelativeTo(null);

// Создаем таблицы

employeeTable = new JTable();

orderTable = new JTable();

orderDetailsTable = new JTable();

dishTable = new JTable();

tableTable = new JTable();

// Создаем модели таблиц

employeeTableModel = new DefaultTableModel();

orderTableModel = new DefaultTableModel();

orderDetailsTableModel = new DefaultTableModel();

dishTableModel = new DefaultTableModel();

tableTableModel = new DefaultTableModel();

// Устанавливаем модели для таблиц

employeeTable.setModel(employeeTableModel);

orderTable.setModel(orderTableModel);

orderDetailsTable.setModel(orderDetailsTableModel);

dishTable.setModel(dishTableModel);

tableTable.setModel(tableTableModel);

// Создаем вкладки

JTabbedPane tabbedPane = new JTabbedPane();

tabbedPane.addTab("Сотрудники", new JScrollPane(employeeTable));

tabbedPane.addTab("Заказы", new JScrollPane(orderTable));

tabbedPane.addTab("Детали заказа", new JScrollPane(orderDetailsTable));

tabbedPane.addTab("Блюда", new JScrollPane(dishTable));

tabbedPane.addTab("Столы", new JScrollPane(tableTable));

// Создаем кнопку "Обновить"

JButton updateButton = new JButton("Обновить");

updateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

updateTables();

}

});

// Добавляем кнопку в панель инструментов

JToolBar toolBar = new JToolBar();

toolBar.add(updateButton);

// Добавляем вкладки и кнопку в фрейм

getContentPane().add(toolBar, BorderLayout.NORTH);

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

// Устанавливаем соединение с базой данных

connectToDatabase();

// Заполняем таблицы

updateTables();

setVisible(true);

}

private void connectToDatabase() {

try {

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/restaurant", "root", "password");

System.out.println("Соединение с базой данных установлено.");

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Ошибка при подключении к базе данных: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateTables() {

try {

// Обновляем данные в таблице "Сотрудники"

updateEmployeeTable();

// Обновляем данные в таблице "Заказы"

updateOrderTable();

// Обновляем данные в таблице "Детали заказа"

updateOrderDetailsTable();

// Обновляем данные в таблице "Блюда"

updateDishTable();

// Обновляем данные в таблице "Столы"

updateTableTable();

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Ошибка при обновлении данных: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

// Методы для обновления таблиц

private void updateEmployeeTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM Employee");

employeeTableModel.setDataVector(getResultSetData(rs), new String[]{"ID", "Имя", "Фамилия", "Должность"});

rs.close();

stmt.close();

}

private void updateOrderTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM Order");

orderTableModel.setDataVector(getResultSetData(rs), new String[]{"ID заказа", "Дата заказа", "Стол", "Статус заказа"});

rs.close();

stmt.close();

}

private void updateOrderDetailsTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM OrderDetails");

orderDetailsTableModel.setDataVector(getResultSetData(rs), new String[]{"ID детали заказа", "ID заказа", "ID блюда", "Количество"});

rs.close();

stmt.close();

}

private void updateDishTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM Dish");

dishTableModel.setDataVector(getResultSetData(rs), new String[]{"ID блюда", "Название", "Цена"});

rs.close();

stmt.close();

}

private void updateTableTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM Table");

tableTableModel.setDataVector(getResultSetData(rs), new String[]{"ID стола", "Номер стола", "Количество мест"});

rs.close();

stmt.close();

}

private Object[][] getResultSetData(ResultSet rs) throws SQLException {

// Получаем количество столбцов

ResultSetMetaData metaData = rs.getMetaData();

int columnCount = metaData.getColumnCount();

// Получаем количество строк

rs.last();

int rowCount = rs.getRow();

rs.beforeFirst();

// Создаем двумерный массив для хранения данных

Object[][] data = new Object[rowCount][columnCount];

// Заполняем массив данными

int row = 0;

while (rs.next()) {

for (int col = 0; col < columnCount; col++) {

data[row][col] = rs.getObject(col + 1);

}

row++;

}

return data;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new Main();

}

});

}

}

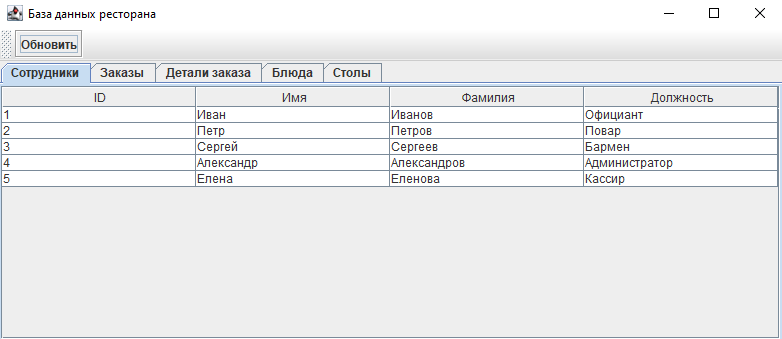


Рисунок 17 – Реализация АИС на Java

4 Разработка сайта

<https://yz3944.craftum.io>

Заключение

Полное выполнение программы — Практикант успешно завершил все поставленные задачи, охватывающие темы от системного программирования до разработки мобильных приложений и сайтов. Каждое задание имеет ясное описание, кодовые реализации, схемы и результаты выполнения программ.

Разработка Автоматизированной информационной системы (АИС) — Проект по созданию системы управления рестораном был выполнен в полном объеме. В отчёте подробно описан процесс анализа предметной области, разработка ER-диаграммы и структуры базы данных с соблюдением 3НФ, а также разработка документации.

Качественная реализация кода и использование OOP — В отчёте представлены классы и методы, отражающие основное понимание ООП. Сложные программы, такие как "Математико" и система управления счётом с историей операций, выполнены с обработкой исключений и соответствующими проверками, что улучшает надёжность кода.

Проектирование и тестирование — В процессе создания программы уделено внимание тестированию, проверке ошибок и обработке исключений, что указывает на тщательный подход к разработке качественного программного обеспечения.

Профессиональные навыки — отчёт демонстрирует владение практическими навыками, которые будут полезны в дальнейшей профессиональной деятельности.

Список используемой литературы

1. Белугина, С. В. Разработка программных модулей программного обеспечения для компьютерных систем. Прикладное программирование / С. В. Белугина. – 4-е изд., стер. – Санкт-Петербург: Лань, 2023. – 312 с.

2. Крыжановская, Ю. А. Основы JAVA: учебное пособие / Ю. А. Крыжановская, В. Г. Ляликова, М. М. Безрядин. – Воронеж: ВГУ, 2020. – 96 с.

3. Курбатова, И. В. Основы программирования на языке Java: учебное пособие для спо / И. В. Курбатова, А. В. Печкуров. – Санкт-Петербург: Лань, 2024. – 348 с.

4. Курбатова, И. В. Основы программирования на языке Java: учебное пособие для вузов / И. В. Курбатова, А. В. Печкуров. – Санкт-Петербург: Лань, 2024. – 348 с.

5. Куляс, О. Л. Курс программирования на ASSEMBLER: учебное пособие / О. Л. Куляс, К. А. Никитин. – Москва: СОЛОН-Пресс, 2017. – 220 с.

6. Фомичева, С. Г. Основы программирования на языке низкого уровня Assembler: учебное пособие / С. Г. Фомичева, О. С. Варига, А. А. Попкова. – Норильск: ЗГУ им. Н.М. Федоровского, 2022. – 198 с.

7. Новикова, Т. В. Системное проектирование АИС учреждения здравоохранения: учебное пособие / Т. В. Новикова. – Томск: СибГМУ, 2017. – 190 с.

8. Викторова, Н. В. 1С Бухгалтерия: учебное пособие / Н. В. Викторова. – Тюмень: ТюмГУ, 2016. – 40 с.

9. Гантц, И. С. 1С: Предприятие. Программирование для начинающих: Практикум: учебное пособие / И. С. Гантц. – Москва: РТУ МИРЭА, 2023. – 71 с.

10. Рысин, М. Л. Введение в современную Android-разработку на языке Java: учебное пособие / М. Л. Рысин. – Москва: РТУ МИРЭА, 2023 – Часть 1 – 2023. – 132 с.