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  Сорокин – Кузнецов Иван Андреевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2024

Содержание

|  |
| --- |
| [Введение 3](#_Toc182473437)  [1.Решение задач 4](#_Toc182473438)  [1.1 Задача 1 4](#_Toc182473439)  [1.2 Задача 2: 5](#_Toc182473440)  [1.3 Задание 3: 10](#_Toc182473441)  [1.4 Задача 4: 13](#_Toc182473442)  [1.5 Задача 5: 17](#_Toc182473443)  [1.6 Задача 6: 18](#_Toc182473444)  [1.7 Задача 7: 23](#_Toc182473445)  [1.8 Задача 8: 25](#_Toc182473446)  [1.9 Задача 9: 27](#_Toc182473447)  [1.10 Задача 10 31](#_Toc182473448)  [Неделя 2 35](#_Toc182473449)  [Разработка АИС по теме "Кадровое агентство" 35](#_Toc182473450)  [Анализ предметной области для базы данных "Кадровое агентство" 35](#_Toc182473451)  [Техническое задание для создания АИС "Кадровое агентство" 36](#_Toc182473452)  [Разработка 1С 46](#_Toc182473453)  [Мобильное приложение: 47](#_Toc182473454)  [Руководство пользователя для АИС "Кадровое агентство" на Android 47](#_Toc182473455)  [Сайт с АИС 48](#_Toc182473456)  [Заключение 49](#_Toc182473457)  [Список литературы 50](#_Toc182473458) |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

Введение

Этот отчет по УП ПМ01 охватывает дисциплины: разработка программных модулей, мобильных приложений, 1С-программирование, поддержка и тестирование программных модулей, системное программирование.

Цели практики:

- Углубление теоретических знаний: Применение полученных теоретических знаний для лучшего освоения изучаемых дисциплин.

- Развитие практических навыков: Создание реальных программных продуктов, таких как АИС, мобильные приложения и сайты.

- Проверка и улучшение решений: Тестирование и доработка созданных программных модулей.

- Создание комплексных решений: Разработка и документирование проектов, включающих программный код и руководство пользователя.

Результаты практики:

- Укрепление и расширение знаний в соответствующих областях.

- Получение практического опыта в разработке программного обеспечения.

- Развитие навыков анализа, проектирования и реализации программных решений.

- Овладение созданием качественных и удобных для пользователей продуктов.

1. Решение задач

1.1 Задача 1

Найти число локальных минимумов. (Соседями элемента матрицы назовем элементы, имеющие с ним общую сторону или угол. Элемент матрицы назы вается локальным минимумом, если он строго меньше всех своих соседей.)

Таблица 1 – Переменные кода задачи 1

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| matrix | int[][] | Двумерный массив целых чисел, представляющий матрицу. |
| rows | int | Количество строк в матрице. |
| cols | int | Количество столбцов в матрице. |
| localMinimaCount | int | Счетчик найденных локальных минимумов. |
| i, j | int | Индексы для итерации по строкам и столбцам матрицы. |
| current | int | Текущий рассматриваемый элемент матрицы. |
| x, y | int | Смещения для поиска соседей текущего элемента. |
| neighborRow, neighborCol | int | Индексы строки и столбца соседа. |

Код:  
public class LocalMinima {

public static void main(String[] args) {

int[][] matrix = {

{3, 4, 5},

{6, 1, 8},

{7, 2, 9}

};

int count = findLocalMinima(matrix);

System.out.println("Число локальных минимумов: " + count);

}

public static int findLocalMinima(int[][] matrix) {

int rows = matrix.length;

int cols = matrix[0].length;

int localMinimaCount = 0;

// Проход по каждому элементу матрицы

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

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

if (isLocalMinimum(matrix, i, j, rows, cols)) {

localMinimaCount++;

}

}

}

return localMinimaCount;

}

// Проверка, является ли элемент локальным минимумом

private static boolean isLocalMinimum(int[][] matrix, int i, int j, int rows, int cols) {

int current = matrix[i][j];

// Проверка всех 8 соседей

for (int x = -1; x <= 1; x++) {

for (int y = -1; y <= 1; y++) {

// Пропустить сам элемент

if (x == 0 && y == 0) {

continue;

}

int neighborRow = i + x;

int neighborCol = j + y;

if (neighborRow >= 0 && neighborRow < rows && neighborCol >= 0 && neighborCol < cols) {

if (matrix[neighborRow][neighborCol] <= current) {

return false;

}

}

}

}

return true;

}

}

Результат:

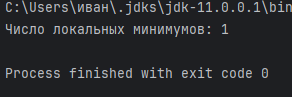


Рисунок 1 – Результат задания 1

1.2 Задача 2:

Определить класс Окружность на плоскости. Определить площадь и периметр. Создать массив/список/множество объектов и определить группы окружностей, центры которых лежат на одной прямой. Определить наи больший и наименьший по площади (периметру) объект

Таблица 2 – Переменные кода задачи 2

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| x, y, radius (в классе Circle) | double | Координаты центра окружности по осям X и Y, радиус окружности. |
| circles (в классе CircleMain) | List<Circle> | Список объектов типа Circle, представляющий множество окружностей. |
| largestAreaCircle, smallestAreaCircle, largestPerimeterCircle, smallestPerimeterCircle (в классе CircleMain) | Circle | Ссылки на окружности с соответствующими характеристиками. |
| group (в методе findCirclesOnSameLine) | List<Circle> | Временный список для группировки окружностей с одинаковой координатой. |
| i, j (в методе findCirclesOnSameLine) | int | Индексы для итерации по списку окружностей. |

Код:

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

class Circle {

private double x; // Координата центра окружности по оси X

private double y; // Координата центра окружности по оси Y

private double radius; // Радиус окружности

public Circle(double x, double y, double radius) {

this.x = x;

this.y = y;

this.radius = radius;

}

public double getX() {

return x;

}

public double getY() {

return y;

}

public double getRadius() {

return radius;

}

// Метод для вычисления площади окружности

public double getArea() {

return Math.PI \* radius \* radius;

}

// Метод для вычисления периметра окружности

public double getPerimeter() {

return 2 \* Math.PI \* radius;

}

@Override

public String toString() {

return String.format("Окружность[центр=(%.2f, %.2f), радиус=%.2f]", x, y, radius);

}

}

public class CircleMain {

public static void main(String[] args) {

// Создаем массив окружностей

List<Circle> circles = new ArrayList<>();

circles.add(new Circle(0, 0, 5));

circles.add(new Circle(0, 1, 3));

circles.add(new Circle(1, 1, 4));

circles.add(new Circle(2, 2, 2));

circles.add(new Circle(0, -1, 6));

// Находим окружности с наибольшей и наименьшей площадью

Circle largestAreaCircle = findCircleWithLargestArea(circles);

Circle smallestAreaCircle = findCircleWithSmallestArea(circles);

System.out.println("Окружность с наибольшей площадью: " + largestAreaCircle);

System.out.println("Окружность с наименьшей площадью: " + smallestAreaCircle);

// Находим окружности с наибольшим и наименьшим периметром

Circle largestPerimeterCircle = findCircleWithLargestPerimeter(circles);

Circle smallestPerimeterCircle = findCircleWithSmallestPerimeter(circles);

System.out.println("Окружность с наибольшим периметром: " + largestPerimeterCircle);

System.out.println("Окружность с наименьшим периметром: " + smallestPerimeterCircle);

// Находим группы окружностей, центры которых лежат на одной прямой

findCirclesOnSameLine(circles);

}

// Метод для нахождения окружности с наибольшей площадью

public static Circle findCircleWithLargestArea(List<Circle> circles) {

return circles.stream()

.max(Comparator.comparingDouble(Circle::getArea))

.orElse(null);

}

// Метод для нахождения окружности с наименьшей площадью

public static Circle findCircleWithSmallestArea(List<Circle> circles) {

return circles.stream()

.min(Comparator.comparingDouble(Circle::getArea))

.orElse(null);

}

// Метод для нахождения окружности с наибольшим периметром

public static Circle findCircleWithLargestPerimeter(List<Circle> circles) {

return circles.stream()

.max(Comparator.comparingDouble(Circle::getPerimeter))

.orElse(null);

}

// Метод для нахождения окружности с наименьшим периметром

public static Circle findCircleWithSmallestPerimeter(List<Circle> circles) {

return circles.stream()

.min(Comparator.comparingDouble(Circle::getPerimeter))

.orElse(null);

}

// Метод для поиска окружностей, центры которых лежат на одной прямой

public static void findCirclesOnSameLine(List<Circle> circles) {

System.out.println("\nГруппы окружностей, центры которых лежат на одной прямой:");

// Проверяем по оси X

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

List<Circle> group = new ArrayList<>();

group.add(circles.get(i));

for (int j = i + 1; j < circles.size(); j++) {

if (circles.get(i).getX() == circles.get(j).getX()) {

group.add(circles.get(j));

}

}

if (group.size() > 1) {

System.out.println("Окружности с одинаковой X-координатой: " + group);

}

}

// Проверяем по оси Y

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

List<Circle> group = new ArrayList<>();

group.add(circles.get(i));

for (int j = i + 1; j < circles.size(); j++) {

if (circles.get(i).getY() == circles.get(j).getY()) {

group.add(circles.get(j));

}

}

if (group.size() > 1) {

System.out.println("Окружности с одинаковой Y-координатой: " + group);

}

}

}

}

Результат:

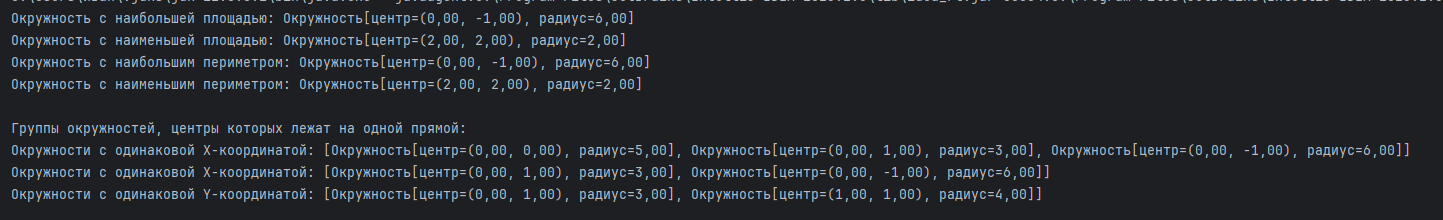


Рисунок 2 – Результат задания 2

1.3 Задание 3:

Создать объект класса Сутки, используя классы Час, Минута. Методы: вывести на консоль текущее время, рассчитать время суток (утро, день, вечер, ночь).

Таблица 3 – Переменные кода задачи 3

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| hour (в классах Hour, Day) | int | Часы (от 0 до 23). |
| minute (в классах Minute, Day) | int | Минуты (от 0 до 59). |
| day (в классе DayTime) | Day | Объект класса Day, представляющий текущее время. |
| scanner (в классе DayTime) | Scanner | Объект для считывания ввода с клавиатуры. |
| response (в классе DayTime) | String | Строка, содержащая ответ пользователя на вопрос о смене времени. |

Код:

import java.util.Scanner;

// Класс Час

class Hour {

private int hour;

public Hour(int hour) {

if (hour < 0 || hour > 23) {

throw new IllegalArgumentException("Час должен быть в диапазоне от 0 до 23.");

}

this.hour = hour;

}

public int getHour() {

return hour;

}

public void setHour(int hour) {

if (hour < 0 || hour > 23) {

throw new IllegalArgumentException("Час должен быть в диапазоне от 0 до 23.");

}

this.hour = hour;

}

}

// Класс Минута

class Minute {

private int minute;

public Minute(int minute) {

if (minute < 0 || minute > 59) {

throw new IllegalArgumentException("Минута должна быть в диапазоне от 0 до 59.");

}

this.minute = minute;

}

public int getMinute() {

return minute;

}

public void setMinute(int minute) {

if (minute < 0 || minute > 59) {

throw new IllegalArgumentException("Минута должна быть в диапазоне от 0 до 59.");

}

this.minute = minute;

}

}

// Класс Сутки

class Day {

private Hour hour;

private Minute minute;

public Day(int hour, int minute) {

this.hour = new Hour(hour);

this.minute = new Minute(minute);

}

// Метод для вывода текущего времени

public void showCurrentTime() {

System.out.printf("Текущее время: %02d:%02d%n", hour.getHour(), minute.getMinute());

}

// Метод для определения времени суток (утро, день, вечер, ночь)

public void determineTimeOfDay() {

int currentHour = hour.getHour();

if (currentHour >= 6 && currentHour < 12) {

System.out.println("Сейчас утро.");

} else if (currentHour >= 12 && currentHour < 18) {

System.out.println("Сейчас день.");

} else if (currentHour >= 18 && currentHour < 22) {

System.out.println("Сейчас вечер.");

} else {

System.out.println("Сейчас ночь.");

}

}

// Метод для изменения времени

public void setTime(int hour, int minute) {

this.hour.setHour(hour);

this.minute.setMinute(minute);

}

}

// Основной класс для демонстрации

public class DayTime {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Ввод времени с клавиатуры

System.out.print("Введите часы (0-23): ");

int hour = scanner.nextInt();

System.out.print("Введите минуты (0-59): ");

int minute = scanner.nextInt();

// Создаем объект Сутки с введенным временем

Day day = new Day(hour, minute);

// Выводим текущее время

day.showCurrentTime();

// Определяем текущее время суток

day.determineTimeOfDay();

// Возможность изменения времени и повторного вывода

System.out.print("\nХотите изменить время? (да/нет): ");

String response = scanner.next();

if (response.equalsIgnoreCase("да")) {

System.out.print("Введите новые часы (0-23): ");

hour = scanner.nextInt();

System.out.print("Введите новые минуты (0-59): ");

minute = scanner.nextInt();

day.setTime(hour, minute);

day.showCurrentTime();

day.determineTimeOfDay();

}

scanner.close();

}

}

Результат:

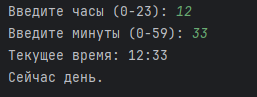


Рисунок 3 – Результат задания 3

1.4 Задача 4:

Создать класс Payment с внутренним классом, с помощью объектов кото рого можно сформировать покупку из нескольких товаров.

Таблица 4 – Переменные кода задачи 4

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| name | String | Название товара |
| price | double | Цена товара |
| quantity | int | Количество товара |
| items | List<Item> | Список товаров в покупке |
| total | double | Общая стоимость покупки |
| scanner | Scanner | Объект для считывания ввода с клавиатуры |
| response | String | Ответ пользователя на вопрос о добавлении нового товара |

Код:

package test;  
  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
// Внешний класс Payment  
public class Payment {  
 // Внутренний класс Item (Товар)  
 public class Item {  
 private String name; // Название товара  
 private double price; // Цена товара  
 private int quantity; // Количество товара  
  
 public Item(String name, double price, int quantity) {  
 this.name = name;  
 this.price = price;  
 this.quantity = quantity;  
 }  
  
 // Метод для получения полной стоимости товара (цена \* количество)  
 public double getTotalPrice() {  
 return price \* quantity;  
 }  
  
 @Override  
 public String toString() {  
 return String.*format*("%s: %d шт., %.2f руб./шт., Итого: %.2f руб.", name, quantity, price, getTotalPrice());  
 }  
 }  
  
 // Список товаров в покупке  
 private List<Item> items = new ArrayList<>();  
  
 // Метод для добавления товара в покупку  
 public void addItem(String name, double price, int quantity) {  
 Item item = new Item(name, price, quantity);  
 items.add(item);  
 }  
  
 // Метод для вывода всех товаров  
 public void showItems() {  
 if (items.isEmpty()) {  
 System.*out*.println("Покупка пуста.");  
 } else {  
 System.*out*.println("Товары в вашей покупке:");  
 for (Item item : items) {  
 System.*out*.println(item);  
 }  
 }  
 }  
  
 // Метод для расчета общей стоимости покупки  
 public double getTotalPayment() {  
 double total = 0;  
 for (Item item : items) {  
 total += item.getTotalPrice();  
 }  
 return total;  
 }  
  
 // Основной метод для демонстрации работы  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 Payment payment = new Payment();  
 String response;  
  
 // Ввод товаров  
 do {  
 System.*out*.print("Введите название товара: ");  
 String name = scanner.nextLine();  
  
 System.*out*.print("Введите цену товара: ");  
 double price = scanner.nextDouble();  
  
 System.*out*.print("Введите количество товара: ");  
 int quantity = scanner.nextInt();  
  
 // Добавляем товар в список покупок  
 payment.addItem(name, price, quantity);  
  
 System.*out*.print("Хотите добавить еще один товар? (да/нет): ");  
 scanner.nextLine(); // Для того, чтобы корректно обработать переход на новую строку  
 response = scanner.nextLine();  
  
 } while (response.equalsIgnoreCase("да"));  
  
 // Выводим список товаров  
 payment.showItems();  
  
 // Выводим общую стоимость покупки  
 System.*out*.printf("Общая стоимость покупки: %.2f руб.%n", payment.getTotalPayment());  
  
 scanner.close();  
 }  
}

Результат:

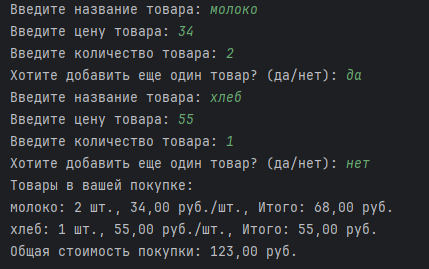


Рисунок 4 – Результат задания 4

1.5 Задача 5:

В заданном тексте найти сумму всех встречающихся цифр.

Таблица 5 – Переменные кода задачи 5

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| scanner | Scanner | Объект для считывания ввода с клавиатуры. |
| text | String | Строка, введенная пользователем. |
| sum | int | Сумма всех цифр, найденных в тексте. |
| i | int | Индекс для итерации по символам строки. |
| currentChar | char | Текущий символ, который проверяется. |

Код:

package test;  
  
import java.util.Scanner;  
  
public class SumOfDigitsInText {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 // Ввод текста пользователем  
 System.*out*.print("Введите текст: ");  
 String text = scanner.nextLine();  
  
 int sum = 0;  
  
 // Проходим по каждому символу строки  
 for (int i = 0; i < text.length(); i++) {  
 char currentChar = text.charAt(i);  
  
 // Проверяем, является ли символ цифрой  
 if (Character.*isDigit*(currentChar)) {  
 // Преобразуем символ в числовое значение и добавляем к сумме  
 sum += Character.*getNumericValue*(currentChar);  
 }  
 }  
  
 // Вывод суммы цифр  
 System.*out*.println("Сумма всех цифр в тексте: " + sum);  
  
 scanner.close();  
 }  
}

Результат:



Рисунок 5 – Результат задания 5

1.6 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.

Таблица 6 – Переменные кода задачи 6

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| hour | int | Часы в формате 24-часовой системы (0-23) |
| minute | int | Минуты (0-59) |
| day | Day | Объект класса Day, представляющий текущие сутки |
| scanner | Scanner | Объект для считывания ввода с клавиатуры |
| response | String | Строка для хранения ответа пользователя (да/нет) |
| writer | FileWriter | Объект для записи в файл |

Код:  
import java.io.\*;

import java.util.InputMismatchException;

import java.util.Scanner;

// Класс Час

class Hour {

private int hour;

public Hour(int hour) {

if (hour < 0 || hour > 23) {

throw new IllegalArgumentException("Час должен быть в диапазоне от 0 до 23.");

}

this.hour = hour;

}

public int getHour() {

return hour;

}

public void setHour(int hour) {

if (hour < 0 || hour > 23) {

throw new IllegalArgumentException("Час должен быть в диапазоне от 0 до 23.");

}

this.hour = hour;

}

}

// Класс Минута

class Minute {

private int minute;

public Minute(int minute) {

if (minute < 0 || minute > 59) {

throw new IllegalArgumentException("Минута должна быть в диапазоне от 0 до 59.");

}

this.minute = minute;

}

public int getMinute() {

return minute;

}

public void setMinute(int minute) {

if (minute < 0 || minute > 59) {

throw new IllegalArgumentException("Минута должна быть в диапазоне от 0 до 59.");

}

this.minute = minute;

}

}

// Класс Сутки

class Day {

private Hour hour;

private Minute minute;

public Day(int hour, int minute) {

this.hour = new Hour(hour);

this.minute = new Minute(minute);

}

// Метод для вывода текущего времени

public void showCurrentTime() {

System.out.printf("Текущее время: %02d:%02d%n", hour.getHour(), minute.getMinute());

}

// Метод для определения времени суток (утро, день, вечер, ночь)

public void determineTimeOfDay() {

int currentHour = hour.getHour();

if (currentHour >= 6 && currentHour < 12) {

System.out.println("Сейчас утро.");

} else if (currentHour >= 12 && currentHour < 18) {

System.out.println("Сейчас день.");

} else if (currentHour >= 18 && currentHour < 22) {

System.out.println("Сейчас вечер.");

} else {

System.out.println("Сейчас ночь.");

}

}

// Метод для изменения времени

public void setTime(int hour, int minute) {

this.hour.setHour(hour);

this.minute.setMinute(minute);

}

}

public class DayTime {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

// Ввод времени с клавиатуры с контролем исключений

System.out.print("Введите часы (0-23): ");

int hour = getIntInput(scanner);

System.out.print("Введите минуты (0-59): ");

int minute = getIntInput(scanner);

// Создаем объект Сутки с введенным временем

Day day = new Day(hour, minute);

// Выводим текущее время

day.showCurrentTime();

// Определяем текущее время суток

day.determineTimeOfDay();

// Возможность изменения времени и повторного вывода

System.out.print("\nХотите изменить время? (да/нет): ");

String response = scanner.next();

if (response.equalsIgnoreCase("да")) {

System.out.print("Введите новые часы (0-23): ");

hour = getIntInput(scanner);

System.out.print("Введите новые минуты (0-59): ");

minute = getIntInput(scanner);

day.setTime(hour, minute);

day.showCurrentTime();

day.determineTimeOfDay();

}

// Пример работы с файлом (чтение/запись)

System.out.print("\nХотите сохранить текущее время в файл? (да/нет): ");

response = scanner.next();

if (response.equalsIgnoreCase("да")) {

saveTimeToFile(day);

}

} catch (InputMismatchException e) {

System.out.println("Ошибка: некорректный ввод данных. Введите целое число.");

} catch (IllegalArgumentException e) {

System.out.println("Ошибка: " + e.getMessage());

} catch (OutOfMemoryError e) {

System.out.println("Ошибка: недостаточно памяти для выполнения операции.");

} catch (IOException e) {

System.out.println("Ошибка: проблема с файлом. " + e.getMessage());

} finally {

scanner.close();

}

}

// Метод для безопасного ввода целого числа

private static int getIntInput(Scanner scanner) {

while (true) {

try {

return scanner.nextInt();

} catch (InputMismatchException e) {

System.out.println("Ошибка: пожалуйста, введите корректное целое число.");

scanner.next(); // Очищаем некорректный ввод

}

}

}

// Метод для сохранения времени в файл

private static void saveTimeToFile(Day day) throws IOException {

try (FileWriter writer = new FileWriter("current\_time.txt")) {

writer.write(String.format("Текущее время: %02d:%02d%n", day.hour.getHour(), day.minute.getMinute()));

System.out.println("Время успешно сохранено в файл.");

} catch (IOException e) {

throw new IOException("Ошибка при записи в файл.", e);

}

}

}

Результат:

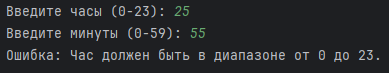


Рисунок 6 – Результат задачи 6 (1)

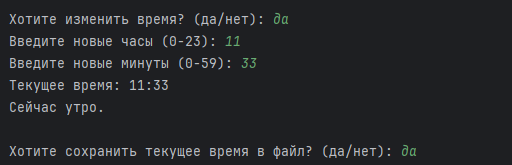


Рисунок 7 – Результат задачи 6 (2)



Рисунок 8 – Результат задачи 6 (3)

1.7 Задача 7:

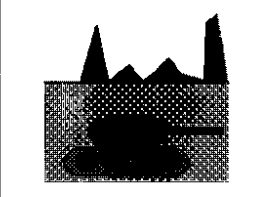


Рисунок 9 – Задание задачи 7

Таблица 7 – Переменные кода задачи 7

|  |  |  |
| --- | --- | --- |
| Переменная | Тип данных | Описание |
| g | Graphics | Объект, представляющий графический контекст, на котором происходит рисование. |
| xPoints1, yPoints1, xPoints2, yPoints2 | int[] | Массивы целых чисел, хранящие координаты вершин треугольников (гор). |
| frame | JFrame | Окно приложения. |
| panel | CustomDrawing | Панель, на которой происходит рисование. |

Код:

import javax.swing.\*;  
import java.awt.\*;  
  
public class CustomDrawing extends JPanel {  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
  
 // Прямоугольник - земля/основание (черный и больше по высоте)  
 g.setColor(Color.*BLACK*);  
 g.fillRect(100, 250, 400, 150); // Увеличена высота  
  
 // Горы (треугольники)  
 g.setColor(Color.*GRAY*);  
 int[] xPoints1 = {120, 200, 280};  
 int[] yPoints1 = {250, 150, 250};  
 g.fillPolygon(xPoints1, yPoints1, 3);  
  
 int[] xPoints2 = {320, 380, 440};  
 int[] yPoints2 = {250, 170, 250};  
 g.fillPolygon(xPoints2, yPoints2, 3);  
  
 // Корпус танка (нижняя часть)  
 g.setColor(Color.*DARK\_GRAY*);  
 g.fillRect(180, 320, 240, 60); // Длинный прямоугольник  
  
 // Башня танка (верхняя часть)  
 g.fillRect(240, 280, 120, 60); // Верхняя часть танка  
  
 // Дуло танка (длинный узкий прямоугольник)  
 g.fillRect(360, 300, 100, 20); // Дуло  
  
 // Колеса танка (круги)  
 g.setColor(Color.*LIGHT\_GRAY*);  
 g.fillOval(200, 380, 60, 60); // Первое колесо  
 g.fillOval(260, 380, 60, 60); // Второе колесо  
 g.fillOval(320, 380, 60, 60); // Третье колесо  
 }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Tank Drawing with Black Base");  
 CustomDrawing panel = new CustomDrawing();  
  
 frame.setSize(600, 600); // Размер окна  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.add(panel);  
 frame.setVisible(true);  
 }  
}

Результат:

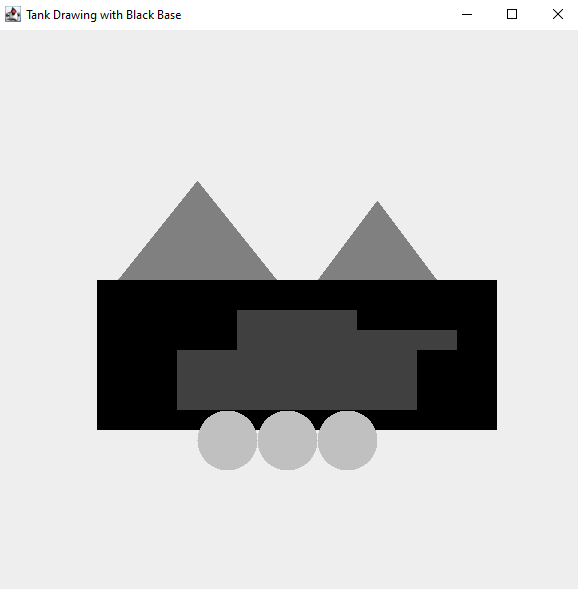


Рисунок 10 – Результат задачи 7

1.8 Задача 8:

Определить объекты TPike и ТКагр, которые являются «на следниками» объекта TFish (см. задачу 19) и отличаются от него только тем, что по-разному изображают себя на экране: TPike — в виде зеленой стрелки, а ТКагр — в виде красного треугольника. Ипользовать виртуальные методы, т.е. вернувшись к определе нию объекта TFish, откорректировать его, сделав Draw пустым и виртуальным.

Таблица 8 – Переменные кода задачи 8

|  |  |  |
| --- | --- | --- |
| Переменная | Тип | Описание |
| name | String | Хранит имя рыбы. Инициализируется в конструкторе базового класса TFish. |
| g | Graphics | Объект класса Graphics, предоставляемый методу paint() для рисования на компоненте. |
| xPoints, yPoints | int[] | Массивы целых чисел, содержащие координаты вершин многоугольника для рисования рыбы. |
| pike, kagr | TFish | Ссылки на объекты классов TPike и TKagr соответственно. Представляют собой экземпляры рыб. |
| frame | FishFrame | Экземпляр класса FishFrame, представляющий главное окно приложения. |
| args | String[] | Массив строковых аргументов командной строки (не используется в данном коде). |

Код:

import javax.swing.\*;

import java.awt.\*;

// Абстрактный класс TFish с виртуальным методом Draw

abstract class TFish {

protected String name;

public TFish(String name) {

this.name = name;

}

public abstract void Draw(Graphics g);

}

// Класс TPike, наследник TFish, отображающийся как зелёная стрелка

class TPike extends TFish {

public TPike(String name) {

super(name);

}

@Override

public void Draw(Graphics g) {

g.setColor(Color.GREEN);

int[] xPoints = {50, 100, 50, 70, 50};

int[] yPoints = {100, 120, 140, 120, 100};

g.fillPolygon(xPoints, yPoints, 5);

}

}

// Класс TKagr, наследник TFish, отображающийся как красный треугольник

class TKagr extends TFish {

public TKagr(String name) {

super(name);

}

@Override

public void Draw(Graphics g) {

g.setColor(Color.RED);

int[] xPoints = {200, 250, 225};

int[] yPoints = {100, 100, 150};

g.fillPolygon(xPoints, yPoints, 3);

}

}

// Класс для отображения графического окна

class FishFrame extends JFrame {

private TFish pike;

private TFish kagr;

public FishFrame() {

setTitle("Fish Display");

setSize(400, 300);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

pike = new TPike("Щука");

kagr = new TKagr("Карп");

}

@Override

public void paint(Graphics g) {

super.paint(g);

pike.Draw(g);

kagr.Draw(g);

}

}

// Главный класс для запуска программы

public class Main {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

FishFrame frame = new FishFrame();

frame.setVisible(true);

});

}

}

Результат:

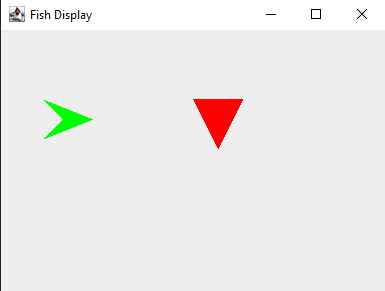


Рисунок 11 – Результат задачи 8

1.9 Задача 9:

Итальянская игра «Математико»\*. Имеется квадратное поле из 25 клеток и набор из 52 карточек, на которых записаны числа от 1 до 13, причем карточки с каждым из этих чисел встречаются по четыре раза.

Разработать программу, которая позволит имитировать игру человека с компьютером: случайным образом извлекается какая-либо из имеющихся карточек и выдается записанное на ней число. Каждый игрок заносит это число в одну из клеток квадрата, и так продолжается до тех пор, пока не будут заполнены все клетки квадрата.

Таблица 9 – Переменные кода задачи 9

|  |  |  |
| --- | --- | --- |
| Переменная | Тип | Описание |
| GRID\_SIZE | int (static final) | Размер игрового поля (константа). |
| NUM\_CARDS | int (static final) | Количество карт в колоде (константа). |
| MAX\_CARD\_VALUE | int (static final) | Максимальное значение карты (константа). |
| deck | ArrayList<Integer> | Колода карт. |
| cells | JButton[][] | Двумерный массив кнопок, представляющий игровое поле. |
| turn | int | Ход игры (0 - игрок, 1 - компьютер). |
| currentCard | int | Текущая вытянутая карта. |

Код:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.Collections;

public class MatematikoGameGUI extends JFrame {

private static final int GRID\_SIZE = 5;

private static final int NUM\_CARDS = 52;

private static final int MAX\_CARD\_VALUE = 13;

private ArrayList<Integer> deck = new ArrayList<>();

private JButton[][] cells = new JButton[GRID\_SIZE][GRID\_SIZE];

private int turn = 0;

private int currentCard = 0;

public MatematikoGameGUI() {

setTitle("Игра Математико");

setSize(600, 600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout());

initializeDeck();

setupGrid();

JButton drawButton = new JButton("Вытянуть карту");

drawButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (!deck.isEmpty()) {

currentCard = drawCard();

JOptionPane.showMessageDialog(null, "Вытянутая карта: " + currentCard);

if (turn % 2 == 0) {

enablePlayerMove();

} else {

computerMove();

}

turn++;

} else {

int score = calculateScore();

JOptionPane.showMessageDialog(null, "Игра завершена! Ваши очки: " + score);

}

}

});

add(drawButton, BorderLayout.SOUTH);

}

private void initializeDeck() {

for (int i = 1; i <= MAX\_CARD\_VALUE; i++) {

for (int j = 0; j < 4; j++) {

deck.add(i);

}

}

Collections.shuffle(deck);

}

private void setupGrid() {

JPanel gridPanel = new JPanel();

gridPanel.setLayout(new GridLayout(GRID\_SIZE, GRID\_SIZE));

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++) {

cells[i][j] = new JButton("");

int row = i;

int col = j;

cells[i][j].addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (cells[row][col].getText().isEmpty() && currentCard != 0) {

cells[row][col].setText(String.valueOf(currentCard));

currentCard = 0;

disableAllButtons();

}

}

});

cells[i][j].setEnabled(false);

gridPanel.add(cells[i][j]);

}

}

add(gridPanel, BorderLayout.CENTER);

}

private int drawCard() {

return deck.remove(deck.size() - 1);

}

private void enablePlayerMove() {

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++) {

if (cells[i][j].getText().isEmpty()) {

cells[i][j].setEnabled(true);

}

}

}

}

private void computerMove() {

JOptionPane.showMessageDialog(null, "Ход компьютера!");

int bestRow = -1, bestCol = -1;

int maxScore = Integer.MIN\_VALUE;

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++) {

if (cells[i][j].getText().isEmpty()) {

cells[i][j].setText(String.valueOf(currentCard));

int score = calculateScore();

if (score > maxScore) {

maxScore = score;

bestRow = i;

bestCol = j;

}

cells[i][j].setText("");

}

}

}

if (bestRow != -1 && bestCol != -1) {

cells[bestRow][bestCol].setText(String.valueOf(currentCard));

}

currentCard = 0;

}

private void disableAllButtons() {

for (int i = 0; i < GRID\_SIZE; i++) {

for (int j = 0; j < GRID\_SIZE; j++) {

cells[i][j].setEnabled(false);

}

}

}

private int calculateScore() {

int score = 0;

for (int i = 0; i < GRID\_SIZE; i++) {

int rowSum = 0;

int colSum = 0;

for (int j = 0; j < GRID\_SIZE; j++) {

if (!cells[i][j].getText().isEmpty()) {

rowSum += Integer.parseInt(cells[i][j].getText());

}

if (!cells[j][i].getText().isEmpty()) {

colSum += Integer.parseInt(cells[j][i].getText());

}

}

score += rowSum + colSum;

}

return score;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

MatematikoGameGUI game = new MatematikoGameGUI();

game.setVisible(true);

});

}

}

Результат:

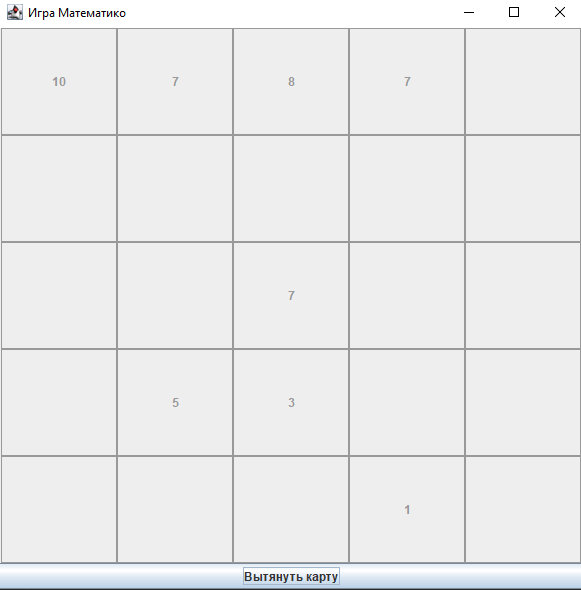


Рисунок 12 – Результат задачи 9

* 1. Задача 10

Задание 1

N DQ ? ; число без знака

Не используя арифметические команды, реализовать оператор

(1 вариант) N := N \* 2

(2 вариант) N := N div 2

Код:

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

Результат:

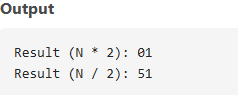


Рисунок 13 – Результат задачи 10 (1)

Задание 2

1 вариант: Реализовать процедуру Padd(x, а), которая реализует оператор х:=x+а. Передача параметров в стеке, стандартные соглашения о связях.

Код:

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

Результат:

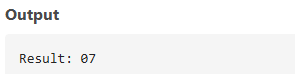


Рисунок 14 – Результат задачи 10 (2)

Неделя 2

Разработка АИС по теме "Кадровое агентство"

Анализ предметной области для базы данных "Кадровое агентство"

**Цель разработки**: Создание базы данных для автоматизации работы кадрового агентства, включая процессы найма, размещения вакансий, подачи заявок кандидатами и отслеживания результатов этих процессов.

База данных кадрового агентства предназначена для автоматизации процессов найма, подачи заявок и отслеживания вакансий. Основные сущности:

1. **Работодатель**: Компания, размещающая вакансии. Атрибуты: название компании, контактный email.
2. **Вакансия**: Предложение работы от работодателя. Атрибуты: должность, описание, местоположение, связь с работодателем.
3. **Кандидат**: Человек, ищущий работу. Атрибуты: имя, email.
4. **Заявка**: Кандидат подает заявку на вакансию. Атрибуты: дата подачи, статус, связи с кандидатом и вакансией.
5. **Навыки**: Специальности, которыми обладает кандидат. Атрибуты: название навыка.
6. **Навыки кандидатов**: Связь между кандидатами и их навыками.

**Связи**:

* Работодатель размещает несколько вакансий (1:М).
* Кандидат подает несколько заявок на вакансии (1:М).
* Вакансия может иметь несколько заявок (1:М).
* Кандидат может обладать несколькими навыками, и каждый навык может быть у нескольких кандидатов (М:М).

Техническое задание для создания АИС "Кадровое агентство"

#### 1. **Введение**

Автоматизированная информационная система (АИС) "Кадровое агентство" предназначена для автоматизации процессов рекрутинга, включая поиск кандидатов на вакансии, управление вакансиями, отслеживание заявок, а также управление данными работодателей и кандидатов.

#### 2. **Цели и задачи системы**

Целью разработки системы является повышение эффективности работы кадрового агентства, улучшение взаимодействия между работодателями и кандидатами, а также автоматизация учета всех операций.

**Задачи:**

* Создание базы данных для хранения информации о кандидатах, работодателях, вакансиях, заявках и навыках.
* Разработка интерфейса для управления данными в базе: добавление, редактирование и удаление информации.
* Разработка функционала для отображения вакансий и кандидатов.
* Разработка отчетных форм и фильтров для поиска кандидатов по навыкам, опыту и другим критериям.
* Обеспечение безопасности данных (шифрование, доступ через логины и пароли).

#### 3. **Функциональные требования**

1. **Модули системы:**
   * **Работодатели**:
     + Добавление, редактирование, удаление информации о работодателях (название компании, контактный email).
     + Просмотр списка работодателей.
   * **Вакансии**:
     + Добавление, редактирование, удаление вакансий (название вакансии, описание, местоположение, ID работодателя).
     + Просмотр списка вакансий.
     + Просмотр вакансий по фильтрам (местоположение, должность, работодатель).
   * **Кандидаты**:
     + Добавление, редактирование, удаление данных кандидатов (имя, email).
     + Просмотр списка кандидатов.
     + Просмотр профилей кандидатов (включая навыки).
   * **Заявки**:
     + Создание заявки кандидата на вакансию.
     + Просмотр всех заявок (связь между кандидатами и вакансиями).
   * **Навыки**:
     + Добавление, редактирование, удаление навыков.
     + Связывание навыков с кандидатами.
   * **Поиск и фильтрация**:
     + Поиск кандидатов по имени, email, навыкам.
     + Фильтрация вакансий по местоположению, работодателю, должности.
2. **Пользователи системы:**
   * **Администратор**:
     + Управление данными всех сущностей (работодатели, вакансии, кандидаты, заявки).
     + Управление правами доступа.
   * **Менеджер кадров**:
     + Просмотр, добавление и редактирование вакансий и кандидатов.
     + Просмотр заявок на вакансии.
   * **Работодатель**:
     + Добавление, редактирование вакансий.
     + Просмотр кандидатов, откликнувшихся на вакансии.
   * **Кандидат**:
     + Просмотр доступных вакансий.
     + Отправка заявок на вакансии.
     + Обновление собственного профиля и навыков.

#### 4. **Нефункциональные требования**

1. **Технические требования**:
   * Программное обеспечение: Java (с использованием JDBC для работы с базой данных), Swing для создания графического интерфейса.
   * Система должна поддерживать многопользовательский доступ с возможностью разграничения прав.
   * Работать на операционных системах Windows, Linux, macOS.
   * Подключение к базе данных MySQL.
2. **Требования к производительности**:
   * Время отклика на запросы должно быть не более 3 секунд для большинства операций.
   * Система должна поддерживать работу с базой данных объемом до 100,000 записей.
3. **Требования безопасности**:
   * Доступ к системе должен быть защищен с помощью логина и пароля.
   * Шифрование данных для защиты личной информации.
   * Возможность восстановить пароль для пользователя.
4. **Интерфейс пользователя**:
   * Графический интерфейс на Java Swing с вкладками для разных сущностей (работодатели, вакансии, кандидаты, заявки).
   * Использование таблиц для отображения данных.
   * Валидация ввода данных (например, проверка формата email).

#### 5. **Структура базы данных**

База данных должна включать следующие таблицы:

1. **Employers**:
   * id (INT, PK)
   * company\_name (VARCHAR)
   * contact\_email (VARCHAR)
2. **Jobs**:
   * id (INT, PK)
   * title (VARCHAR)
   * description (TEXT)
   * location (VARCHAR)
   * employer\_id (INT, FK на таблицу Employers)
3. **Candidates**:
   * id (INT, PK)
   * name (VARCHAR)
   * email (VARCHAR)
4. **Applications**:
   * id (INT, PK)
   * candidate\_id (INT, FK на таблицу Candidates)
   * job\_id (INT, FK на таблицу Jobs)
   * application\_date (DATE)
   * status (VARCHAR)
5. **Skills**:
   * id (INT, PK)
   * skill\_name (VARCHAR)
6. **Candidate\_skills**:
   * candidate\_id (INT, FK на таблицу Candidates)
   * skill\_id (INT, FK на таблицу Skills)

#### 6. **Возможности отчетности**

Система должна поддерживать создание отчетов по следующим направлениям:

* Список вакансий с данными о работодателе.
* Список кандидатов по навыкам.
* Статистика по количеству заявок на вакансии.

#### 7. **Требования к тестированию**

* Проведение юнит-тестирования для всех критичных функций (например, добавление, редактирование данных).
* Проведение нагрузочного тестирования для проверки производительности системы с большим количеством записей.

#### 8. **Сроки разработки и внедрения**

* Этап 1: Анализ и проектирование — 2 недели.
* Этап 2: Разработка и тестирование — 8 недель.
* Этап 3: Внедрение и обучение пользователей — 2 недели.

#### 9. **Заключение**

АИС "Кадровое агентство" позволит автоматизировать процессы рекрутинга, улучшит взаимодействие с кандидатами и работодателями и повысит производительность кадрового агентства.

ER-Диаграмма:

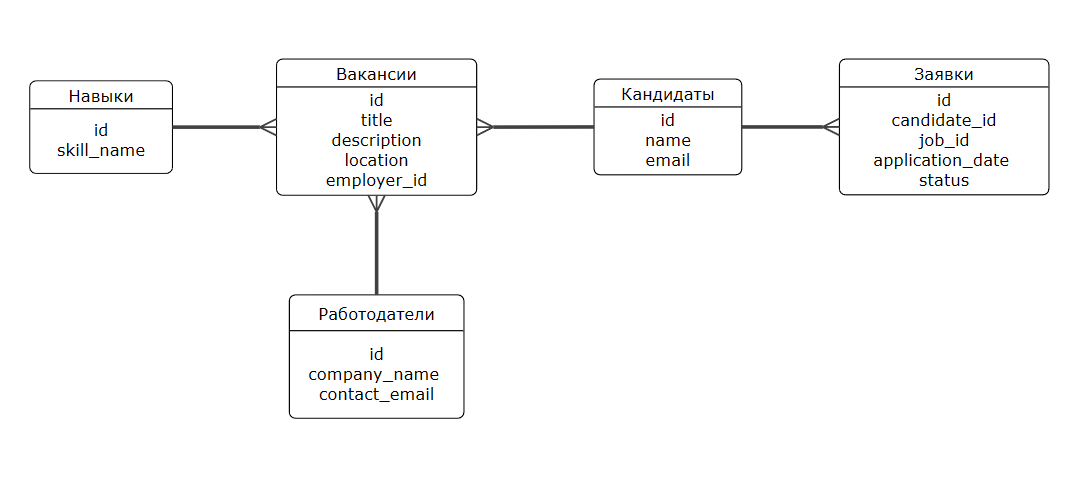


Рисунок 15 – ER Диаграмма

Код АИС:

package org.example;

import java.sql.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

public class RecruitmentAgencyApp extends JFrame {

private JTable employerTable;

private JTable jobTable;

private JTable candidateTable;

private JTable applicationTable;

private JTable skillTable;

private JTable candidateSkillTable;

private DefaultTableModel employerTableModel;

private DefaultTableModel jobTableModel;

private DefaultTableModel candidateTableModel;

private DefaultTableModel applicationTableModel;

private DefaultTableModel skillTableModel;

private DefaultTableModel candidateSkillTableModel;

private Connection connection;

public RecruitmentAgencyApp() {

super("База данных рекрутингового агентства");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(800, 600);

setLocationRelativeTo(null);

employerTable = new JTable();

jobTable = new JTable();

candidateTable = new JTable();

applicationTable = new JTable();

skillTable = new JTable();

candidateSkillTable = new JTable();

employerTableModel = new DefaultTableModel();

jobTableModel = new DefaultTableModel();

candidateTableModel = new DefaultTableModel();

applicationTableModel = new DefaultTableModel();

skillTableModel = new DefaultTableModel();

candidateSkillTableModel = new DefaultTableModel();

employerTable.setModel(employerTableModel);

jobTable.setModel(jobTableModel);

candidateTable.setModel(candidateTableModel);

applicationTable.setModel(applicationTableModel);

skillTable.setModel(skillTableModel);

candidateSkillTable.setModel(candidateSkillTableModel);

JTabbedPane tabbedPane = new JTabbedPane();

tabbedPane.addTab("Работодатели", new JScrollPane(employerTable));

tabbedPane.addTab("Вакансии", new JScrollPane(jobTable));

tabbedPane.addTab("Кандидаты", new JScrollPane(candidateTable));

tabbedPane.addTab("Заявки", new JScrollPane(applicationTable));

tabbedPane.addTab("Навыки", new JScrollPane(skillTable));

tabbedPane.addTab("Навыки кандидатов", new JScrollPane(candidateSkillTable));

JButton updateButton = new JButton("Обновить");

updateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

updateTables();

}

});

JToolBar toolBar = new JToolBar();

toolBar.add(updateButton);

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

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

connectToDatabase();

updateTables();

setVisible(true);

}

private void connectToDatabase() {

try {

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/recruitment\_agency?useUnicode=true&characterEncoding=UTF-8", "root", "Bublikya123");

} catch (SQLException e) {

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

}

}

private void updateTables() {

try {

updateEmployerTable();

updateJobTable();

updateCandidateTable();

updateApplicationTable();

updateSkillTable();

updateCandidateSkillTable();

} catch (SQLException e) {

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

}

}

private void updateEmployerTable() throws SQLException {

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

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

employerTableModel.setDataVector(getResultSetData(rs), new String[]{"ID", "Компания", "Email"});

rs.close();

stmt.close();

}

private void updateJobTable() throws SQLException {

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

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

jobTableModel.setDataVector(getResultSetData(rs), new String[]{"ID вакансии", "Название", "Описание", "Местоположение", "Работодатель ID"});

rs.close();

stmt.close();

}

private void updateCandidateTable() throws SQLException {

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

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

candidateTableModel.setDataVector(getResultSetData(rs), new String[]{"ID кандидата", "Имя", "Email"});

rs.close();

stmt.close();

}

private void updateApplicationTable() throws SQLException {

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

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

applicationTableModel.setDataVector(getResultSetData(rs), new String[]{"ID заявки", "Кандидат ID", "Вакансия ID", "Дата", "Статус"});

rs.close();

stmt.close();

}

private void updateSkillTable() throws SQLException {

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

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

skillTableModel.setDataVector(getResultSetData(rs), new String[]{"ID навыка", "Название"});

rs.close();

stmt.close();

}

private void updateCandidateSkillTable() throws SQLException {

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

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

candidateSkillTableModel.setDataVector(getResultSetData(rs), new String[]{"Кандидат ID", "Навык ID"});

rs.close();

stmt.close();

}

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

ResultSetMetaData metaData = rs.getMetaData();

int columnCount = metaData.getColumnCount();

rs.last();

int rowCount = rs.getRow();

rs.beforeFirst();

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

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 RecruitmentAgencyApp();

}

});

}

}

Скриншоты готовой БД:

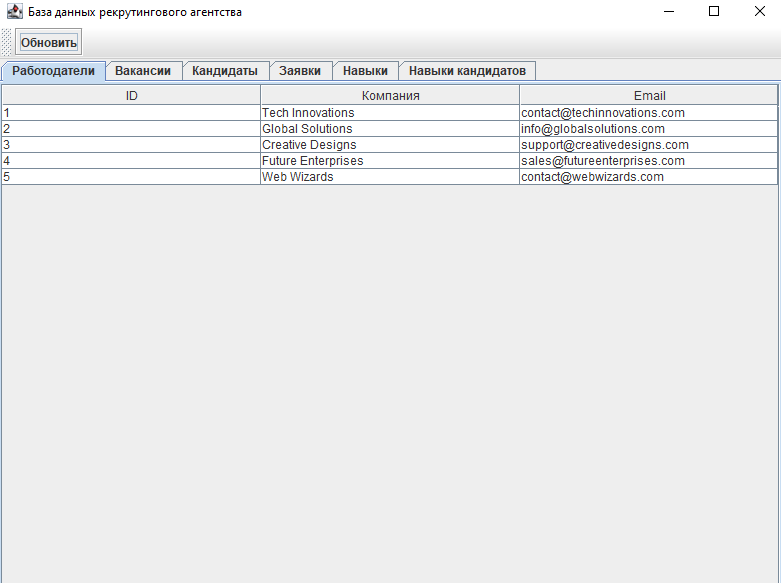


Рисунок 15 – Демонстрация АИС

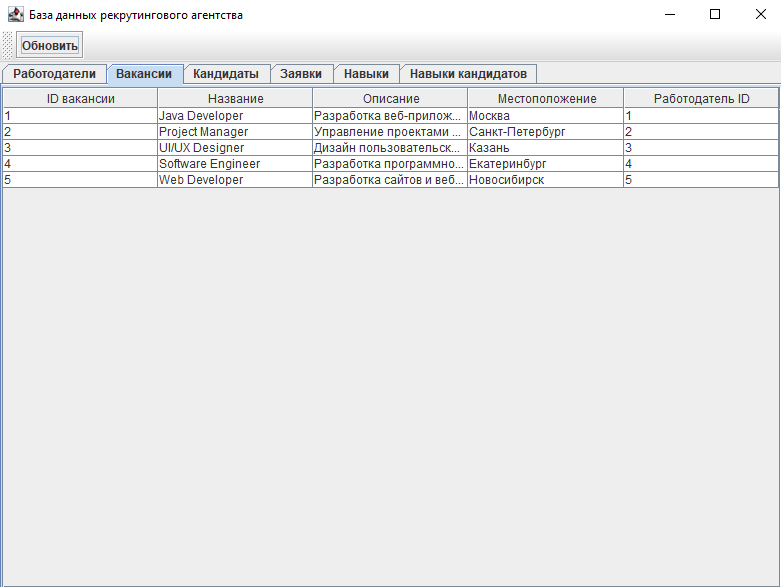


Рисунок 16 – Демонстрация АИС

Разработка 1С

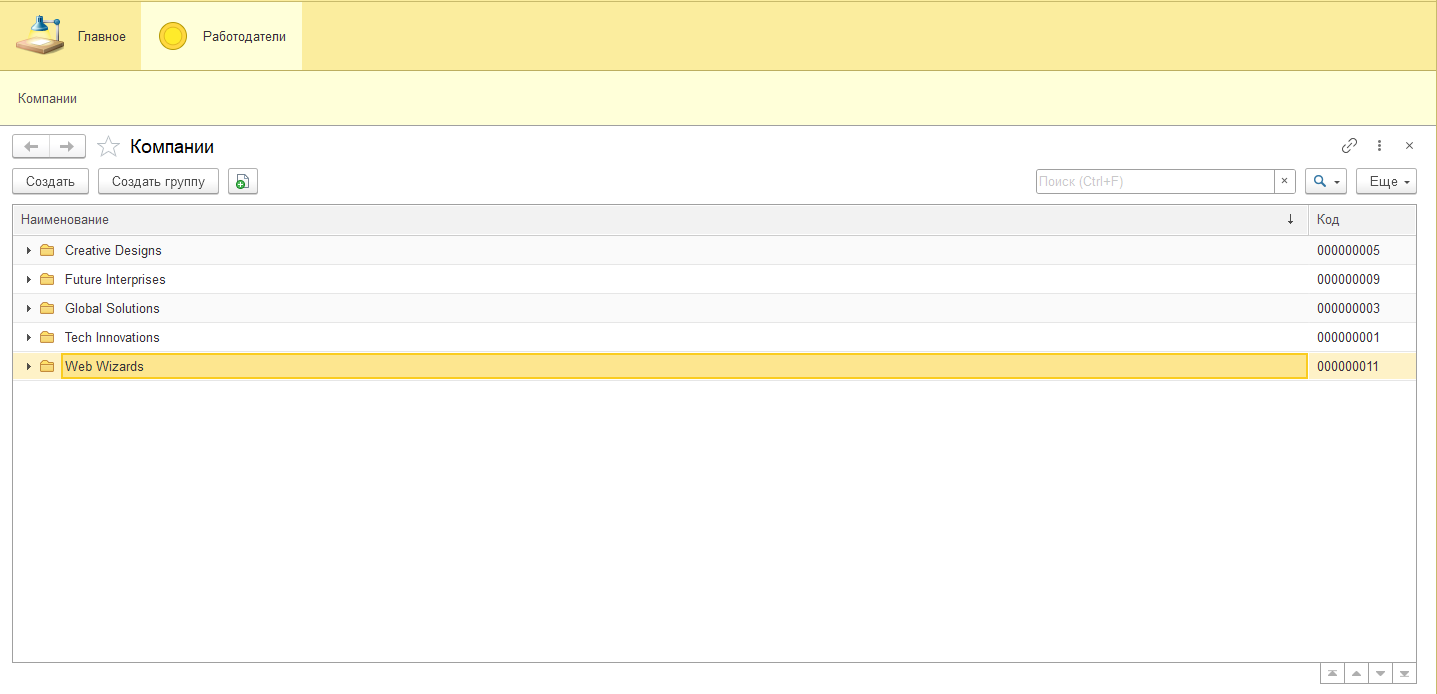


Рисунок 17 – Демонстрация БД в 1С

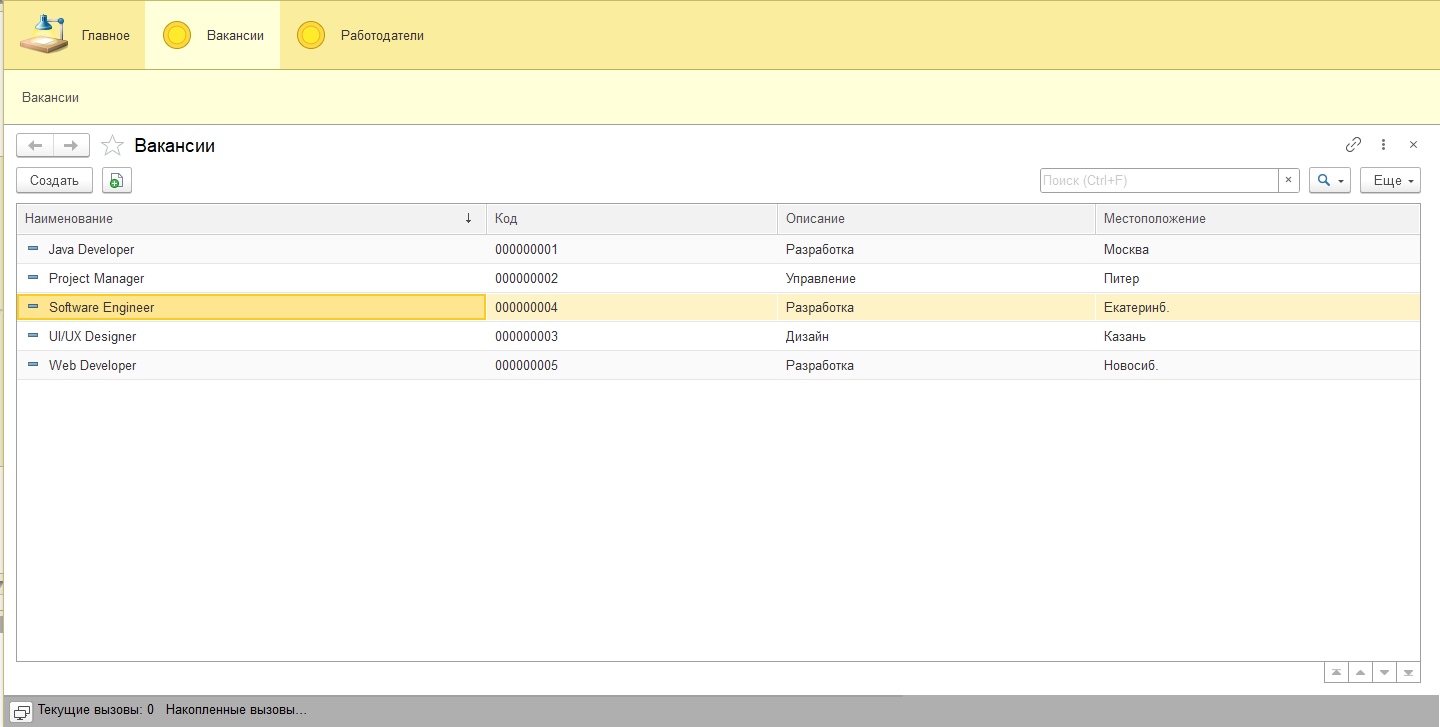


Рисунок 18 – Демонстрация БД в 1С

Мобильное приложение:

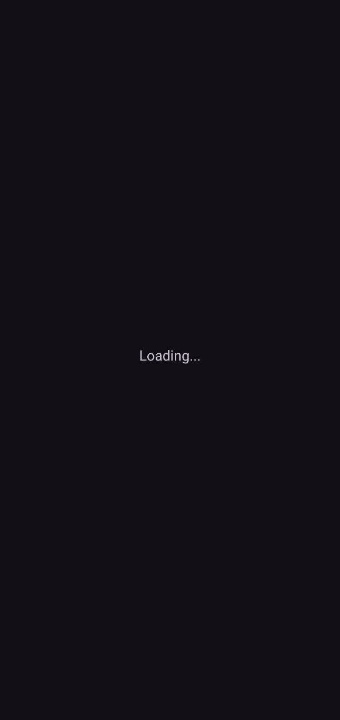


Рисунок 19 – Демонстрация мобильного приложения

Руководство пользователя для АИС "Кадровое агентство" на Android

1. Начало работы

* Установка: Скачайте приложение из Google Play и откройте его.
* Регистрация: Создайте учетную запись с указанием имени, email и пароля.

2. Основные функции

* Вакансии: Добавляйте, редактируйте и просматривайте вакансии.
* Кандидаты: Добавляйте кандидатов, просматривайте резюме и отслеживайте их статус.
* Заявки: Управляйте заявками от кандидатов на вакансии.
* Чат: Общайтесь с кандидатами и клиентами в чате.

3. Дополнительные возможности

* Фильтры: Используйте фильтры для поиска вакансий и кандидатов.
* Напоминания: Устанавливайте напоминания о собеседованиях и встречах.
* Календарь: Интегрируйте приложение с вашим календарем.

4. Настройки

* Персонализируйте информацию о себе, настройте уведомления и внешний вид приложения.

5. Поддержка

* Для помощи обращайтесь в службу поддержки через раздел "Контакты".

Эти функции помогают эффективно управлять кадровыми процессами в агентстве.

Сайт с АИС

Ссылка на сайт с описанием разработки АИС “Кадровое Агентство”:

[st2323.craftum.io](https://st2323.craftum.io/)

Заключение

Разработка автоматизированной информационной системы (АИС) для кадрового агентства является важным шагом на пути к оптимизации бизнес-процессов, улучшению качества обслуживания клиентов и сокращению времени на подбор персонала. Такой подход значительно упрощает управление вакансиями, кандидатами и заявками, а также позволяет агентству эффективно организовать коммуникацию между всеми участниками процесса.

АИС для кадрового агентства позволяет:

1. Автоматизировать процессы подбора персонала, обеспечивая быстрый поиск и фильтрацию кандидатов по ключевым критериям, таким как опыт работы, квалификация и другие параметры.

2. Управлять вакансиями и заявками с возможностью отслеживания статуса каждой заявки, что способствует улучшению процессов взаимодействия с клиентами и кандидатами.

3. Обеспечить удобство общения через встроенные чаты и уведомления, что позволяет своевременно информировать кандидатов и клиентов о ходе работы.

4. Гибко настраивать систему под специфические нужды агентства, добавлять новые функции и адаптировать приложение под изменения в законодательстве или бизнес-операциях.

Кроме того, применение АИС позволяет сократить ошибки, связанные с человеческим фактором, повысить прозрачность процессов и улучшить отчетность. Это, в свою очередь, способствует росту доверия со стороны клиентов и улучшению репутации кадрового агентства на рынке.

В заключение, создание и внедрение АИС для кадрового агентства не только улучшает внутренние процессы, но и способствует повышению конкурентоспособности, сокращая временные затраты на подбор персонала и увеличивая качество обслуживания.

Список литературы

1. Микросервисы с Spring Boot и Netflix / Джош Лонг. – O'Reilly Media, 2019. – 400 с.
2. Java 9 для начинающих / Патрик Нимейер. – O'Reilly Media, 2018. – 336 с.
3. Архитектура корпоративных приложений в Java / Мартин Фаулер. – Addison-Wesley Professional, 2020. – 576 с.
4. Современные подходы к разработке на Java / Энди Хантер, Кэмерон МакКул, Билл Бурке. – Manning Publications, 2021. – 512 с.
5. Spring в действии / Крэйг Уоллс.– Manning Publications, 2023. – 896 с.
6. Эффективные Java-приложения: разработка, тестирование и производительность / Кей С. Хорстманн. – Pearson Education, 2021. – 704 с.
7. Масштабируемая Java для крупных предприятий / Кевин Донохью, Армандо Файя, Алекс Сильва. – O'Reilly Media, 2019. – 464 с.
8. Реактивное программирование на Java / Скотт Майерс. – O'Reilly Media, 2020. – 352 с.
9. Разработка высокопроизводительных приложений на Java / Майкл Моррисон, Патрик Нимейер, Рафаэль Сандерс, Даг Лиди, Джеймс Суэйл. – Manning Publications, 2022. – 448 с.
10. Java Persistence with Hibernate / Крис Вермюлен, Гэвин Кинг, Лиам Хьюитт, Максим Коломыцев, Джонатан Ганц, Питер Дентон. – Packt Publishing, 2022. – 528 с.

Ссылка на GitHub: https://github.com/roxasttt/UP01