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  Благодаров Вячеслав Игоревич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_             (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_             (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_       (подпись, дата) |

Калининград, 2024

Содержание

|  |  |
| --- | --- |
| Введение |  |
| 1 Решение задач |  |
| 1.1 Задача 1 |  |
| 1.2 Задача 2 |  |
|  |  |
| 2 Разработка АИС « » |  |
| 2.1 ER-диаграмма, диаграмма классов, диаграмма… |  |
| 2.2 Техническое задание |  |
|  |  |
| 3 Разработка мобильного приложения |  |
|  |  |
| 4 Разработка сайта |  |
|  |  |
| 5 Руководство пользователя |  |
|  |  |
| Заключение |  |
|  |  |
| Список использованных источников |  |
|  |  |
| Приложение |  |

Введение

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

Цели:

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

Разработка программных модулей

Задача 1

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

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

Код:

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import java.util.Scanner;

public class MatrixMaxElementRemoval {

   public static void main(String[] args) {

       Scanner scanner = new Scanner(System.in);

       // Ввод размерности матрицы

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

       int n = scanner.nextInt();

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

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

       Random random = new Random();

       for (int i = 0; i < n; i++) {

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

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

           }

       }

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

       printMatrix(matrix);

       // Найдем максимальный элемент и его позиции

       int max = Integer.MIN\_VALUE;

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

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

       for (int i = 0; i < n; i++) {

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

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

                   max = matrix[i][j];

                   rowsToRemove.clear();

                   colsToRemove.clear();

                   rowsToRemove.add(i);

                   colsToRemove.add(j);

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

                   if (!rowsToRemove.contains(i)) {

                       rowsToRemove.add(i);

                   }

                   if (!colsToRemove.contains(j)) {

                       colsToRemove.add(j);

                   }

               }

           }

       }

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

       // Удаляем строки и столбцы, содержащие максимальный элемент

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

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

       printMatrix(updatedMatrix);

       scanner.close();

   }

   // Метод для удаления строк и столбцов

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

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

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

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

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

       }

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

       int newRow = 0;

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

           if (rowsToRemove.contains(i)) continue;

           int newCol = 0;

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

               if (colsToRemove.contains(j)) continue;

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

               newCol++;

           }

           newRow++;

       }

       return newMatrix;

   }

   // Метод для печати матрицы

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

       if (matrix.length == 0) {

           System.out.println("Матрица пуста");

           return;

       }

       for (int[] row : matrix) {

           for (int elem : row) {

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

           }

           System.out.println();

       }

   }

}

Рисунок 1 – результат задания 1

​

Задание 2

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

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

Рисунок 2 - Диаграмма классов задания 2

Код:

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Complex {

   private double real;  // Реальная часть

   private double imag;  // Мнимая часть

   // Конструктор для создания комплексного числа

   public Complex(double real, double imag) {

       this.real = real;

       this.imag = imag;

   }

   // Метод для сложения двух комплексных чисел

   public Complex add(Complex other) {

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

   }

   // Метод для умножения двух комплексных чисел

   public Complex multiply(Complex other) {

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

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

       return new Complex(newReal, newImag);

   }

   // Метод для вывода комплексного числа в строковом формате

   @Override

   public String toString() {

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

   }

}

public class ComplexCalculator {

   public static void main(String[] args) {

       Scanner scanner = new Scanner(System.in);

       // Ввод размерности массива

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

       int n = scanner.nextInt();

       // Создание и заполнение массива комплексных чисел

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

       for (int i = 0; i < n; i++) {

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

           double real = scanner.nextDouble();

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

           double imag = scanner.nextDouble();

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

       }

       // Выполнение сложения всех комплексных чисел

       Complex sumResult = addAllComplexNumbers(complexNumbers);

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

       // Выполнение умножения всех комплексных чисел

       Complex productResult = multiplyAllComplexNumbers(complexNumbers);

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

       scanner.close();

   }

   // Метод для сложения всех элементов списка комплексных чисел

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

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

       for (Complex complex : numbers) {

           sum = sum.add(complex);

       }

       return sum;

   }

   // Метод для умножения всех элементов списка комплексных чисел

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

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

       for (Complex complex : numbers) {

           product = product.multiply(complex);

       }

       return product;

   }

}

Рисунок 3 – решение задания 2

Задание 3

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

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

Рисунок 4 – Диаграмма задания 3

Код:

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

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

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

Рисунок 4 – результат задания 3

Задание 4

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

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

Рисунок 5 – диаграмма задания 4

Код:

java.util.ArrayList;

import java.util.List;

// Класс Справочная Служба Общественного Транспорта

public class TransportInfoService {

   // Внутренний класс Route

   public class Route {

       private String routeNumber;

       private String workingHours;

       private String lines;

       private double fare;

       // Конструктор

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

           this.routeNumber = routeNumber;

           this.workingHours = workingHours;

           this.lines = lines;

           this.fare = fare;

       }

       // Геттеры

       public String getRouteNumber() {

           return routeNumber;

       }

       public String getWorkingHours() {

           return workingHours;

       }

       public String getLines() {

           return lines;

       }

       public double getFare() {

           return fare;

       }

       @Override

       public String toString() {

           return "Route{" +

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

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

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

                   ", fare=" + fare +

                   '}';

       }

   }

   // Хранилище маршрутов

   private List<Route> routes;

   // Конструктор

   public TransportInfoService() {

       routes = new ArrayList<>();

   }

   // Метод для добавления маршрута

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

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

   }

   // Метод для поиска маршрутов по номеру

   public Route findRouteByNumber(String routeNumber) {

       for (Route route : routes) {

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

               return route;

           }

       }

       return null;

   }

   // Метод для вывода всех маршрутов

   public void printAllRoutes() {

       for (Route route : routes) {

           System.out.println(route);

       }

   }

   // Главный метод для тестирования

   public static void main(String[] args) {

       TransportInfoService service = new TransportInfoService();

       // Добавляем несколько маршрутов

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

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

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

       // Ищем маршрут по номеру

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

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

       if (route != null) {

           System.out.println(route);

       } else {

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

       }

       // Выводим все маршруты

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

       service.printAllRoutes();

   }

}

Рисунок 6 – результат задания 4

Задание 5

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

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

Код:

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class WordLengthFinder {

   // Метод для нахождения слов максимальной и минимальной длины

   public static void findAndPrintWordsByLength(String text) {

       // Разделяем текст на слова

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

       if (words.length == 0) {

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

           return;

       }

       // Инициализируем переменные для минимальной и максимальной длины

       int minLength = Integer.MAX\_VALUE;

       int maxLength = Integer.MIN\_VALUE;

       // Списки для хранения слов минимальной и максимальной длины

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

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

       // Определяем минимальную и максимальную длину

       for (String word : words) {

           int length = word.length();

           if (length < minLength) {

               minLength = length;

               minLengthWords.clear();

               minLengthWords.add(word);

           } else if (length == minLength) {

               minLengthWords.add(word);

           }

           if (length > maxLength) {

               maxLength = length;

               maxLengthWords.clear();

               maxLengthWords.add(word);

           } else if (length == maxLength) {

               maxLengthWords.add(word);

           }

       }

       // Выводим слова минимальной длины

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

       for (String word : minLengthWords) {

           System.out.println(word);

       }

       // Выводим слова максимальной длины

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

       for (String word : maxLengthWords) {

           System.out.println(word);

       }

   }

   // Главный метод для тестирования

   public static void main(String[] args) {

       Scanner scanner = new Scanner(System.in);

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

       String text = scanner.nextLine();

       scanner.close();

       findAndPrintWordsByLength(text);

   }

}

Задание 8

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

Код:

package com.example.demo1;

import java.util.HashSet;

import java.util.Set;

public class FixedSizeStringArray {

   private String[] array;

   private int size;

   // Конструктор, принимающий размер массива

   public FixedSizeStringArray(int size) {

       this.size = size;

       this.array = new String[size];

   }

   // Метод для получения строки по индексу

   public String get(int index) {

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

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

       }

       return array[index];

   }

   // Метод для установки строки по индексу

   public void set(int index, String value) {

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

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

       }

       array[index] = value;

   }

   // Метод для печати элементов массива

   public void printArray() {

       for (String s : array) {

           System.out.println(s);

       }

   }

   // Метод для слияния двух массивов с исключением повторяющихся элементов

   public FixedSizeStringArray merge(FixedSizeStringArray other) {

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

       for (String s : this.array) {

           if (s != null) {

               set.add(s);

           }

       }

       for (String s : other.array) {

           if (s != null) {

               set.add(s);

           }

       }

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

       int index = 0;

       for (String s : set) {

           mergedArray.set(index++, s);

       }

       return mergedArray;

   }

   // Метод для поэлементного сцепления двух массивов

   public FixedSizeStringArray concatenate(FixedSizeStringArray other) {

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

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

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

       }

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

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

       }

       return concatenatedArray;

   }

   // Основной метод для тестирования класса

   public static void main(String[] args) {

       FixedSizeStringArray array1 = new FixedSizeStringArray(3);

       array1.set(0, "Привет");

       array1.set(1, "Мир");

       array1.set(2, "Java");

       FixedSizeStringArray array2 = new FixedSizeStringArray(3);

       array2.set(0, "Java");

       array2.set(1, "Программирование");

       array2.set(2, "Язык");

       System.out.println("Массив 1:");

       array1.printArray();

       System.out.println("\nМассив 2:");

       array2.printArray();

       FixedSizeStringArray mergedArray = array1.merge(array2);

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

       mergedArray.printArray();

       FixedSizeStringArray concatenatedArray = array1.concatenate(array2);

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

       concatenatedArray.printArray();

   }

}

Результат:

Рисунок 9 – результат задания 8

Рисунок 10 – результат задания 8

Задание 9

Рисунок 11 – задание 9

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

Код:

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

Результат:

Рисунок 12 – решение задания 9

​​​​Разработка АИС

2. Разработка АИС «Поликлиника»

Анализ предметной области

Автоматизированная информационная система для поликлиники включает базу данных с информацией о пациентах, врачах, расписании приёма и услугах.

ER-диаграмма

ER-диаграмма включает сущности:

Пациенты

Врачи

Расписание

Услуги

Реализация базы данных на MySQL

CREATE TABLE Patients (

   PatientID INT PRIMARY KEY,

   Name VARCHAR(100),

   BirthDate DATE,

   Gender CHAR(1),

   ContactInfo VARCHAR(100)

);

CREATE TABLE Doctors (

   DoctorID INT PRIMARY KEY,

   Name VARCHAR(100),

   Specialization VARCHAR(100)

);

CREATE TABLE Schedule (

   ScheduleID INT PRIMARY KEY,

   DoctorID INT,

   Date DATE,

   Time TIME,

   FOREIGN KEY (DoctorID) REFERENCES Doctors(DoctorID)

);

3. Клиент-серверное приложение на Java

Клиентская часть разработана на Java и взаимодействует с базой данных MySQL.

Основные методы клиента

import java.sql.\*;

public class DatabaseClient {

   private Connection connection;

   public DatabaseClient() throws SQLException {

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

   }

   public void fetchPatients() {

       String query = "SELECT \* FROM Patients";

       try (Statement stmt = connection.createStatement(); ResultSet rs = stmt.executeQuery(query)) {

           while (rs.next()) {

               System.out.println("Пациент: " + rs.getString("Name"));

           }

       } catch (SQLException e) {

           e.printStackTrace();

       }

   }

}

​​Разработка мобтльного приложения

4. Разработка мобильного приложения (Android)

Описание

Мобильное приложение позволяет пользователям записываться на прием, просматривать расписание врачей и получать основные сведения о поликлинике.

Код для Android-приложения

Главный экран содержит две кнопки: "Запись на приём" и "Просмотр расписания".

MainActivity.java

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

   @Override

   protected void onCreate(Bundle savedInstanceState) {

       super.onCreate(savedInstanceState);

       setContentView(R.layout.activity\_main);

       Button appointmentButton = findViewById(R.id.btnAppointment);

       Button scheduleButton = findViewById(R.id.btnSchedule);

       appointmentButton.setOnClickListener(new View.OnClickListener() {

           @Override

           public void onClick(View v) {

               Intent intent = new Intent(MainActivity.this, AppointmentActivity.class);

               startActivity(intent);

           }

       });

       scheduleButton.setOnClickListener(new View.OnClickListener() {

           @Override

           public void onClick(View v) {

               Intent intent = new Intent(MainActivity.this, ScheduleActivity.class);

               startActivity(intent);

           }

       });

   }

}

AppointmentActivity.java

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class AppointmentActivity extends AppCompatActivity {

   @Override

   protected void onCreate(Bundle savedInstanceState) {

       super.onCreate(savedInstanceState);

       setContentView(R.layout.activity\_appointment);

       // Код для обработки записи на приём

   }

}

ScheduleActivity.java

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

public class ScheduleActivity extends AppCompatActivity {

   @Override

   protected void onCreate(Bundle savedInstanceState) {

       super.onCreate(savedInstanceState);

       setContentView(R.layout.activity\_schedule);

       // Код для отображения расписания врачей

   }

}

activity\_main.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout\_width="match\_parent"

   android:layout\_height="match\_parent"

   android:orientation="vertical"

   android:padding="16dp">

   <Button

       android:id="@+id/btnAppointment"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:text="Запись на приём" />

   <Button

       android:id="@+id/btnSchedule"

       android:layout\_width="match\_parent"

       android:layout\_height="wrap\_content"

       android:text="Просмотр расписания" />

</LinearLayout>

5. Создание веб-сайта с использованием CMS

Описание сайта

Сайт создан на CMS, например, craftum. Содержит следующие страницы:

<https://craftum.com/app/site/622792/page/737084>

Заключение

Отчёт охватывает выполнение всех этапов практики, включая задачи на Java, разработку АИС «Поликлиника», создание клиент-серверного приложения, мобильного приложения и веб-сайта.