Министерство образования Калининградской области

государственное бюджетное учреждение Калининградской области

профессиональная образовательная организация

«Колледж информационных технологий и строительства»

(ГБУ КО ПОО «КИТиС»)

**Отчет по учебной практике**

УП.01 Разработка модулей программного обеспечения для компьютерных систем

по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Специальность 09.02.07 «Информационные системы и программирование»

Сроки прохождения практики:

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

Место практики ГБУ КО ПОО «КИТиС»

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

Калининград, 2024

Содержание

[Введение 3](#_Toc177500889)

[1 Решение задач 4](#_Toc177500890)

[1.1 Задача 1 4](#_Toc177500891)

[1.2 Задача 2 5](#_Toc177500892)

[1.3 Задача 3 8](#_Toc177500893)

[1.4 Задача 4 10](#_Toc177500894)

[1.5 Задачи 5 12](#_Toc177500895)

[1.6 Задача 6 12](#_Toc177500896)

Введение

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

Цели практики:

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

1. Решение задач
   1. Задача 1

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

import java.util.Random;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

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

Random random = new Random();

int min = -n;

int max = n;

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

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

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

}

}

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

printMatrix(a);

int[] rows = new int[n];

int[] columns = new int[n];

int rowCount = 0;

int columnCount = 0;

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

boolean rowHasZero = false;

boolean columnHasZero = false;

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

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

rowHasZero = true;

}

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

columnHasZero = true;

}

}

if (!rowHasZero) {

rows[rowCount] = i;

rowCount++;

}

if (!columnHasZero) {

columns[columnCount] = i;

columnCount++;

}

}

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

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

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

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

}

}

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

printMatrix(b);

}

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

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

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

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

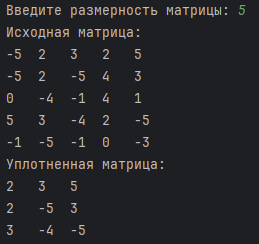
}

System.out.println();

}

}

}



* 1. Задача 2

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

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

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

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

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

public class Main {

public static void main(String[] args) {

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

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

QuadraticEquation q3 = q1.add(q2);

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

QuadraticEquation q4 = q1.subtract(q2);

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

QuadraticEquation q5 = q1.multiply(q2);

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

QuadraticEquation q6 = q1.divide(q2);

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

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

equations.add(q1);

equations.add(q2);

equations.add(q3);

equations.add(q4);

equations.add(q5);

equations.add(q6);

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

for (QuadraticEquation equation : equations) {

double[] equationRoots = equation.getRoots();

for (double root : equationRoots) {

roots.add(root);

}

}

Collections.sort(roots);

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

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

}

}

class QuadraticEquation {

double a;

double b;

double c;

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

this.a = a;

this.b = b;

this.c = c;

}

public double getDiscriminant() {

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

}

public boolean hasRealRoots() {

return getDiscriminant() >= 0;

}

public double[] getRoots() {

double d = getDiscriminant();

if (d < 0) {

return new double[0];

} else if (d == 0) {

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

return new double[]{x};

} else {

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

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

return new double[]{x1, x2};

}

}

public double getExtremePoint() {

return -b / (2 \* a);

}

public boolean isDecreasing() {

return a < 0;

}

public boolean isIncreasing() {

return a > 0;

}

public QuadraticEquation add(QuadraticEquation other) {

double newA = this.a + other.a;

double newB = this.b + other.b;

double newC = this.c + other.c;

return new QuadraticEquation(newA, newB, newC);

}

public QuadraticEquation subtract(QuadraticEquation other) {

double newA = this.a - other.a;

double newB = this.b - other.b;

double newC = this.c - other.c;

return new QuadraticEquation(newA, newB, newC);

}

public QuadraticEquation multiply(QuadraticEquation other) {

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

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

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

return new QuadraticEquation(newA, newB, newC);

}

public QuadraticEquation divide(QuadraticEquation other) {

double newA = this.a / other.a;

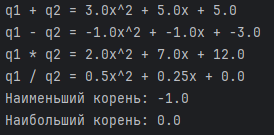
double newB = (this.b \* other.a - this.a \* other.b) / (other.a \* other.a);

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

return new QuadraticEquation(newA, newB, newC);

}

}



* 1. Задача 3

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

import java.util.ArrayList;

import java.util.List;

class Door {

private boolean isLocked;

public void lock() {

isLocked = true;

}

public boolean isLocked() {

return isLocked;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

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

Door door = (Door) obj;

return isLocked == door.isLocked;

}

@Override

public int hashCode() {

return isLocked ? 1 : 0;

}

@Override

public String toString() {

return "Door{" +

"isLocked=" + isLocked +

'}';

}

}

class Window {

private boolean isOpen;

public void open() {

isOpen = true;

}

public boolean isOpen() {

return isOpen;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

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

Window window = (Window) obj;

return isOpen == window.isOpen;

}

@Override

public int hashCode() {

return isOpen ? 1 : 0;

}

@Override

public String toString() {

return "Window{" +

"isOpen=" + isOpen +

'}';

}

}

class House {

private List<Window> windows;

private List<Door> doors;

public House() {

windows = new ArrayList<>();

doors = new ArrayList<>();

}

public void addWindow(Window window) {

windows.add(window);

}

public void addDoor(Door door) {

doors.add(door);

}

public void lockDoor(int index) {

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

doors.get(index).lock();

}

}

public int getWindowCount() {

return windows.size();

}

public int getDoorCount() {

return doors.size();

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

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

House house = (House) obj;

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

}

@Override

public int hashCode() {

int result = windows.hashCode();

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

return result;

}

@Override

public String toString() {

return "House{" +

"windows=" + windows +

", doors=" + doors +

'}';

}

}

public class Main {

public static void main(String[] args) {

House house = new House();

house.addWindow(new Window());

house.addWindow(new Window());

house.addDoor(new Door());

house.addDoor(new Door());

house.lockDoor(0);

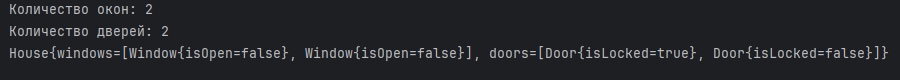
System.out.println("Количество окон: " + house.getWindowCount());

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

System.out.println(house);

}

}



* 1. Задача 4

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

// Внутренний класс для хранения информации об операционной системе

class OS {

private String name;

private String version;

public OS(String name, String version) {

this.name = name;

this.version = version;

}

public String getName() {

return name;

}

public String getVersion() {

return version;

}

}

// Внутренний класс для хранения информации о процессоре

class Processor {

private String model;

private int cores;

public Processor(String model, int cores) {

this.model = model;

this.cores = cores;

}

public String getModel() {

return model;

}

public int getCores() {

return cores;

}

}

// Внутренний класс для хранения информации об оперативной памяти

class RAM {

private int size;

private String type;

public RAM(int size, String type) {

this.size = size;

this.type = type;

}

public int getSize() {

return size;

}

public String getType() {

return type;

}

}

// Основной класс для хранения информации о компьютере

class Computer {

private OS os;

private Processor processor;

private RAM ram;

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

this.os = os;

this.processor = processor;

this.ram = ram;

}

public OS getOs() {

return os;

}

public Processor getProcessor() {

return processor;

}

public RAM getRam() {

return ram;

}

public void displayInfo() {

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

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

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

}

}

// Пример использования класса Computer

public class Main {

public static void main(String[] args) {

// Создаем объекты внутренних классов

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

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

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

// Создаем объект основного класса Computer

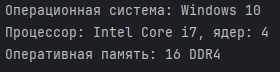
Computer pc = new Computer(windows, intel, ddr4);

// Выводим информацию о компьютере

pc.displayInfo();

}

}



* 1. Задачи 5

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

import java.util.Scanner;

public class TelegramBill {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

double wordCost = scanner.nextDouble();

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

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

String message = scanner.nextLine();

int wordCount = countWords(message);

double totalCost = wordCount \* wordCost;

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

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

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

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

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

}

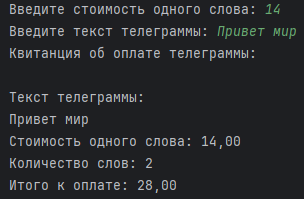
private static int countWords(String message) {

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

return words.length;

}

}



* 1. Задача 6

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

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

import java.time.DayOfWeek;  
import java.time.LocalDate;  
import java.time.temporal.ChronoUnit;  
import java.util.InputMismatchException;  
import java.util.Objects;  
import java.util.Scanner;  
  
class Month {  
 private int month;  
  
 public Month(int month) {  
 this.month = month;  
 }  
  
 public int getMonth() {  
 return month;  
 }  
}  
  
class Day {  
 private int day;  
  
 public Day(int day) {  
 this.day = day;  
 }  
  
 public int getDay() {  
 return day;  
 }  
}  
  
class Year {  
 private int year;  
  
 public Year(int year) {  
 this.year = year;  
 }  
  
 public int getYear() {  
 return year;  
 }  
}  
  
class Date {  
 private Year year;  
 private Month month;  
 private Day day;  
  
 public Date(Year year, Month month, Day day) {  
 this.year = year;  
 this.month = month;  
 this.day = day;  
 }  
  
 public void setDate(int year, int month, int day) {  
 this.year = new Year(year);  
 this.month = new Month(month);  
 this.day = new Day(day);  
 }  
  
 public void printDayOfWeek() {  
 LocalDate localDate = LocalDate.*of*(year.getYear(), month.getMonth(), day.getDay());  
 DayOfWeek dayOfWeek = localDate.getDayOfWeek();  
 System.*out*.println("День недели: " + dayOfWeek);  
 }  
  
 public long calculateDaysBetweenDates(Date startDate, Date endDate) {  
 LocalDate startLocalDate = LocalDate.*of*(startDate.year.getYear(), startDate.month.getMonth(), startDate.day.getDay());  
 LocalDate endLocalDate = LocalDate.*of*(endDate.year.getYear(), endDate.month.getMonth(), endDate.day.getDay());  
 return ChronoUnit.*DAYS*.between(startLocalDate, endLocalDate);  
 }  
  
 public long calculateMonthsBetweenDates(Date startDate, Date endDate) {  
 LocalDate startLocalDate = LocalDate.*of*(startDate.year.getYear(), startDate.month.getMonth(), startDate.day.getDay());  
 LocalDate endLocalDate = LocalDate.*of*(endDate.year.getYear(), endDate.month.getMonth(), endDate.day.getDay());  
 return ChronoUnit.*MONTHS*.between(startLocalDate, endLocalDate);  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Date date = (Date) o;  
 return year.equals(date.year) && month.equals(date.month) && day.equals(date.day);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(year, month, day);  
 }  
  
 @Override  
 public String toString() {  
 return "Дата{" +  
 "Год=" + year +  
 ", Месяц=" + month +  
 ", День=" + day +  
 '}';  
 }  
}  
  
public class DateApp {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 while (true) {  
 try {  
 System.*out*.print("Введите год: ");  
 int year = scanner.nextInt();  
 System.*out*.print("Введите месяц: ");  
 int month = scanner.nextInt();  
 System.*out*.print("Введите день: ");  
 int day = scanner.nextInt();  
  
 if (month < 1 || month > 12) {  
 throw new InvalidMonthException("Недопустимый месяц");  
 }  
  
 if (day < 1 || day > 31) {  
 throw new InvalidDayException("Недопустимый день");  
 }  
  
 Date date = new Date(new Year(year), new Month(month), new Day(day));  
 date.printDayOfWeek();  
  
 System.*out*.print("Введите год: ");  
 int startYear = scanner.nextInt();  
 System.*out*.print("Введите месяц: ");  
 int startMonth = scanner.nextInt();  
 System.*out*.print("Введите день: ");  
 int startDay = scanner.nextInt();  
  
 System.*out*.print("Введите год: ");  
 int endYear = scanner.nextInt();  
 System.*out*.print("Введите месяц: ");  
 int endMonth = scanner.nextInt();  
 System.*out*.print("Введите день: ");  
 int endDay = scanner.nextInt();  
  
 Date startDate = new Date(new Year(startYear), new Month(startMonth), new Day(startDay));  
 Date endDate = new Date(new Year(endYear), new Month(endMonth), new Day(endDay));  
  
 System.*out*.println("Дней между датами: " + date.calculateDaysBetweenDates(startDate, endDate));  
 System.*out*.println("Месяцей между датами: " + date.calculateMonthsBetweenDates(startDate, endDate));  
  
 break;  
 } catch (InputMismatchException e) {  
 System.*out*.println("Недопустимый ввод. Пожалуйста, введите целое число.");  
 scanner.next();  
 } catch (InvalidMonthException e) {  
 System.*out*.println(e.getMessage());  
 } catch (InvalidDayException e) {  
 System.*out*.println(e.getMessage());  
 }  
 }  
 }  
}  
  
class InvalidMonthException extends Exception {  
 public InvalidMonthException(String message) {  
 super(message);  
 }  
}  
  
class InvalidDayException extends Exception {  
 public InvalidDayException(String message) {  
 super(message);  
 }  
} import java.util.ArrayList;

import java.util.List;

class Door {

private boolean isLocked;

public void lock() {

isLocked = true;

}

public boolean isLocked() {

return isLocked;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

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

Door door = (Door) obj;

return isLocked == door.isLocked;

}

@Override

public int hashCode() {

return isLocked ? 1 : 0;

}

@Override

public String toString() {

return "Door{" +

"isLocked=" + isLocked +

'}';

}

}

class Window {

private boolean isOpen;

public void open() {

isOpen = true;

}

public boolean isOpen() {

return isOpen;

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

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

Window window = (Window) obj;

return isOpen == window.isOpen;

}

@Override

public int hashCode() {

return isOpen ? 1 : 0;

}

@Override

public String toString() {

return "Window{" +

"isOpen=" + isOpen +

'}';

}

}

class House {

private List<Window> windows;

private List<Door> doors;

public House() {

windows = new ArrayList<>();

doors = new ArrayList<>();

}

public void addWindow(Window window) {

windows.add(window);

}

public void addDoor(Door door) {

doors.add(door);

}

public void lockDoor(int index) {

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

doors.get(index).lock();

}

}

public int getWindowCount() {

return windows.size();

}

public int getDoorCount() {

return doors.size();

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

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

House house = (House) obj;

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

}

@Override

public int hashCode() {

int result = windows.hashCode();

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

return result;

}

@Override

public String toString() {

return "House{" +

"windows=" + windows +

", doors=" + doors +

'}';

}

}

public class Main {

public static void main(String[] args) {

House house = new House();

house.addWindow(new Window());

house.addWindow(new Window());

house.addDoor(new Door());

house.addDoor(new Door());

house.lockDoor(0);

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

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

System.out.println(house);

}

}

