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 21-1  Хорунов Иван Павлович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2024

|  |  |
| --- | --- |
|  |  |
| Содержание  [Введение 4](#_Toc182488777)  [1 Решение задач 6](#_Toc182488778)  [1.1 Задача 1 6](#_Toc182488779)  [1.2 Задача 2 8](#_Toc182488780)  [1.3 Задача 3 12](#_Toc182488781)  [1.4 Задача 4 19](#_Toc182488782)  [1.5 Задача 5 23](#_Toc182488783)  [1.6 Задача 6 30](#_Toc182488784)  [1.7 Задача 7 35](#_Toc182488785)  [1.8 Задача 8 41](#_Toc182488786)  [1.9 Задача 9 44](#_Toc182488787)  [1.10 Задание 10 50](#_Toc182488788)  [2 Разработка АИС 51](#_Toc182488789)  [2.1 Задание 1,2,3 ER – диаграмма 51](#_Toc182488790)  [2.2 Задание 4 51](#_Toc182488791)  [2.3 Задание 5 54](#_Toc182488792)  [2.4 Спецификация системы и алгоритмы 57](#_Toc182488793)  [2.4.1 Спецификация системы 57](#_Toc182488794)  [2.4.2 Основной алгоритм учета (укрупненная блок-схема) 58](#_Toc182488795)  [2.4.3 Детализация функции системы: Покупка билета 59](#_Toc182488796)  [2.5 Разработка структурной и функциональной схемы АИС 60](#_Toc182488797)  [2.6 Проектирование и реализация интерфейса 60](#_Toc182488798)  [2.7 Разработка встроенной системы справочного руководства и «Руководства пользователя» 66](#_Toc182488799)  [2.8 Проведение тестирования 66](#_Toc182488800)  [2.9 Функциональное тестирование 67](#_Toc182488801)  [2.10 Формирование отчета по практике 67](#_Toc182488802)  [2.11 Особые условия: обеспечение механизма авторизации и разграничения прав 68](#_Toc182488803)  [3 Разработка сайта, мобильного приложения и тестирование 68](#_Toc182488804)  [3.1 Разработка мобильного приложения для АИС "Киноконцертный Комплекс" 68](#_Toc182488805)  [3.2 Создание сайта с использованием CMS 73](#_Toc182488806)  [3.3 Оценка программного продукта с точки зрения эффективности использования ресурсов и применение паттерна проектирования 73](#_Toc182488807)  [3.4 Предложения по дальнейшей оптимизации 75](#_Toc182488808)  [3.5 Проведение оптимизационных процедур 75](#_Toc182488809)  [Заключение 76](#_Toc182488810)  [Список использованных источников 77](#_Toc182488811) |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Введение

Эта практика направлена на закрепление теоретических знаний, полученных в процессе изучения дисциплин, а также их практическое применение в условиях, приближенных к реальной профессиональной деятельности. Основная цель учебной практики — обеспечить студентов возможностью применения теоретических знаний на практике, развить профессиональные навыки и подготовить их к будущей профессиональной деятельности. Во время практики мне предстоит решить широкий спектр задач, включая разработку модулей программного обеспечения для компьютерных систем, тестирование и проектирование информационных систем.

Практика включает несколько этапов, начиная с решения задач по учебнику по программированию на языке Java и заканчивая разработкой и внедрением автоматизированных информационных систем для различных предметных областей. В ходе учебной практики я изучаю основы анализа и проектирования информационных систем, учатся разрабатывать ER-диаграммы, проектировать пользовательские интерфейсы и выполнять тестирование программного обеспечения. На втором этапе практики особое внимание уделяется проектированию и реализации автоматизированной информационной системы для заданной предметной области, где я разрабатываю ER-диаграммы, проектирую базы данных, а также создаю и тестирую программное обеспечение, соответствующее требованиям учебной программы.

Значимым аспектом является оформление всех выполненных работ в соответствии с государственными стандартами, а также соблюдение требований к структуре и содержанию отчетов. Согласно методическим рекомендациям, отчет по практике должен включать титульный лист, содержание, введение, основную часть с подробным описанием выполненных заданий и результатов, заключение и список использованных источников. Особое внимание следует уделить правильному оформлению ссылок, скриншотов, блок-схем и таблиц, что позволит не только продемонстрировать свои знания и навыки, но и представить их в наглядной и профессионально оформленной форме. Таким образом, успешное прохождение учебной практики позволит мне не только укрепить свои знания и развить навыки, но и получить ценный опыт работы с реальными проектами и технологиями. Эта учебная практика играет важную роль в подготовке к дальнейшей профессиональной деятельности в условиях нынешнего реального времени, где от специалистов требуется глубокое понимание процессов разработки программного обеспечения, тестирования и оптимизации.

1 Решение задач

* 1. Задача 1

Определить принадлежность некоторого значения k интервалам (n, m], [n, m), (n, m), [n, m].

Таблица 1 – «Таблица идентификаторов 1»

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение переменной |
| path | Path object | Хранение текущего рисуемого пути |
| n | Начало интервала | 2.0 |
| m | Конец интервала | 5.0 |
| k | Проверяемое значение | 3.0 |
| inOpenClosed | Проверка принадлежит интервалу (n, m] | True |
| inOpenClosed | Проверка принадлежит интервалу [n, m) | True |
| inOpenClosed | Проверка принадлежит интервалу (n, m] | True |
| inClosedClosed | Проверка принадлежит интервалу [n, m) | True |

Таблица 2 – «Таблица компонентов 1»

|  |  |  |  |
| --- | --- | --- | --- |
| Назв. Компон. | Класс | Св-во | Метод |
| Button1 | AppCompatButton | Color = #000000 | OnClick Pencil |

Блок-схема

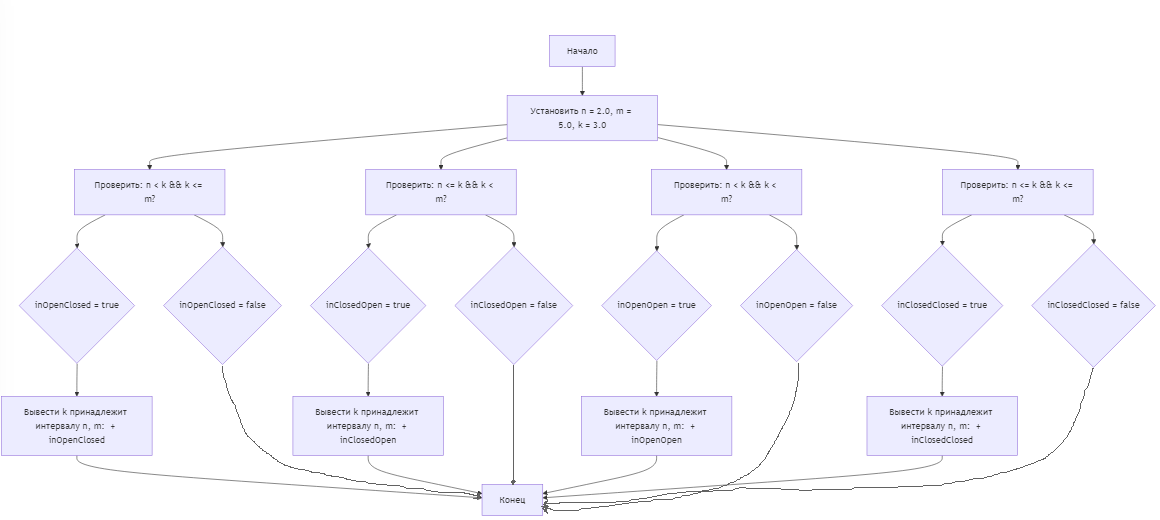


Рисунок 1 «Блок схема 1»

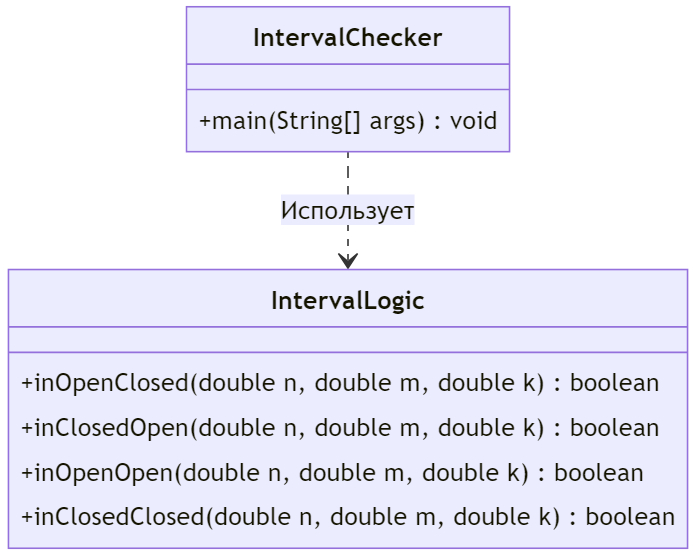
Диаграмма-классов  


Рисунок 2 – «Схема классов 1»

Код

public class IntervalChecker {

public static void main(String[] args) {

double n = 2.0;

double m = 5.0;

double k = 3.0;

boolean inOpenClosed = (n < k && k <= m);

boolean inClosedOpen = (n <= k && k < m);

boolean inOpenOpen = (n < k && k < m);

boolean inClosedClosed = (n <= k && k <= m);

System.out.println("Value k belong to interval (n, m]: " + inOpenClosed);

System.out.println("Value k belong to interval [n, m): " + inClosedOpen);

System.out.println("Value k belong to interval (n, m): " + inOpenOpen);

System.out.println("Value k belong to interval [n, m]: " + inClosedClosed);

}

}

Скриншот

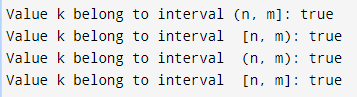


Рисунок 3 – «Результат 1»

* 1. Задача 2

Построить класс Булев вектор (BoolVector). Реализовать методы для выполнения поразрядных конъюнкции, дизъюнкции и отрицания векторов, а также подсчета числа единиц и нулей в векторе.

Таблица 3 – «Таблица идентификаторов 2»

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Описание |
| path | Path object | Хранение текущего рисуемого пути |
| vector | boolean[] | Массив для хранения логических значений вектора |
| size | int | Размер инициализируемого вектора |
| intialValues | boolean[] | Начальные значения для вектора |
| index | int | Индекс для установки или получения значения в векторе |
| value | boolean | Логическое значение для установки в векторе с заданным индексом |
| other | BoolVector | Другой объект BoolVector для операций сравнения |
| result | BoolVector | Результирующий логический вектор после выполнения логических операций |
| count | int | Счетчик для подсчёта единиц или нулей в векторе |
| bit | boolean[] | Каждый бит вектора в цикле |
| sb | StringBuilder | Объект StringBuilder для построения строкового представления вектора |

Блок схема

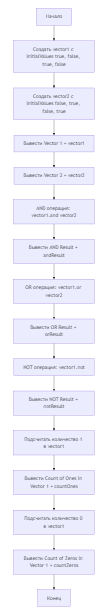


Рисунок 4 – «Блок схема 2»

Диаграмма-классов

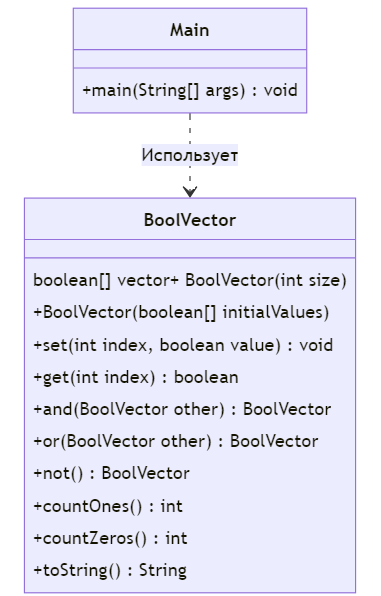


Рисунок 5 – «Схема классов 2»

Код  
public class BoolVector {

private boolean[] vector;

public BoolVector(int size) {

vector = new boolean[size];

}

public BoolVector(boolean[] initialValues) {

vector = new boolean[initialValues.length];

System.arraycopy(initialValues, 0, vector, 0, initialValues.length);

}

public void set(int index, boolean value) {

if (index >= 0 && index < vector.length) {

vector[index] = value;

} else {

throw new IndexOutOfBoundsException("Index beyond vector.");

}

}

public boolean get(int index) {

if (index >= 0 && index < vector.length) {

return vector[index];

} else {

throw new IndexOutOfBoundsException("Index beyond vector.");

}

}

public BoolVector and(BoolVector other) {

if (vector.length != other.vector.length) {

throw new IllegalArgumentException("Vectors should be same size.");

}

BoolVector result = new BoolVector(vector.length);

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

result.set(i, this.vector[i] && other.vector[i]);

}

return result;

}

public BoolVector or(BoolVector other) {

if (vector.length != other.vector.length) {

throw new IllegalArgumentException("Vectors should be same size.");

}

BoolVector result = new BoolVector(vector.length);

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

result.set(i, this.vector[i] || other.vector[i]);

}

return result;

}

public BoolVector not() {

BoolVector result = new BoolVector(vector.length);

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

result.set(i, !this.vector[i]);

}

return result;

}

public int countOnes() {

int count = 0;

for (boolean bit : vector) {

if (bit) {

count++;

}

}

return count;

}

public int countZeros() {

int count = 0;

for (boolean bit : vector) {

if (!bit) {

count++;

}

}

return count;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

sb.append("[");

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

sb.append(vector[i] ? "1" : "0");

if (i < vector.length - 1) {

sb.append(", ");

}

}

sb.append("]");

return sb.toString();

}

public static void main(String[] args) {

boolean[] values1 = {true, false, true, false};

boolean[] values2 = {false, true, false, true};

BoolVector vector1 = new BoolVector(values1);

BoolVector vector2 = new BoolVector(values2);

System.out.println("Vector 1: " + vector1);

System.out.println("Vector 2: " + vector2);

BoolVector andResult = vector1.and(vector2);

System.out.println("AND Result: " + andResult);

BoolVector orResult = vector1.or(vector2);

System.out.println("OR Result: " + orResult);

BoolVector notResult = vector1.not();

System.out.println("NOT Result: " + notResult);

System.out.println("Count of Ones in Vector 1: " + vector1.countOnes());

System.out.println("Count of Zeros in Vector 1: " + vector1.countZeros());

}

}

Результат

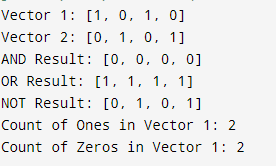


Рисунок 6 – «Результат 2»

* 1. Задача 3

Мотоциклист. Определить иерархию амуниции. Экипировать мотоциклиста. Подсчитать стоимость. Провести сортировку амуниции на основе веса. Найти элементы амуниции, соответствующие заданному диапазону параметров цены.

Таблица 4 – «Таблица идентификаторов 3»

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Описание |
| String | name | Название амуниции (например, "Шлем", "Куртка"). |
| double | price | Цена амуниции. |
| double | weight | Вес амуниции. |
| List<Ammunition> | equipment | Список объектов Ammunition, представляющий экипировку мотоциклиста. |
| double | totalCost | Общая стоимость всей экипировки мотоциклиста. |
| List<Ammunition> | result | Список амуниции, найденной в заданном диапазоне цен. |
| boolean | exit | Логическая переменная, управляющая циклом работы консольного меню. |
| int | choice | Переменная для хранения выбора пользователя в консольном меню. |
| double | minPrice | Минимальная цена, введенная пользователем для поиска амуниции по диапазону цен. |
| double | maxPrice | Максимальная цена, введенная пользователем для поиска амуниции по диапазону цен. |
| List<Ammunition> | foundItems | Список амуниции, найденной в заданном диапазоне цен, возвращаемый методом findEquipmentByPriceRange. |
| Scanner | scanner | Объект для считывания пользовательского ввода из консоли. |

Блок схема

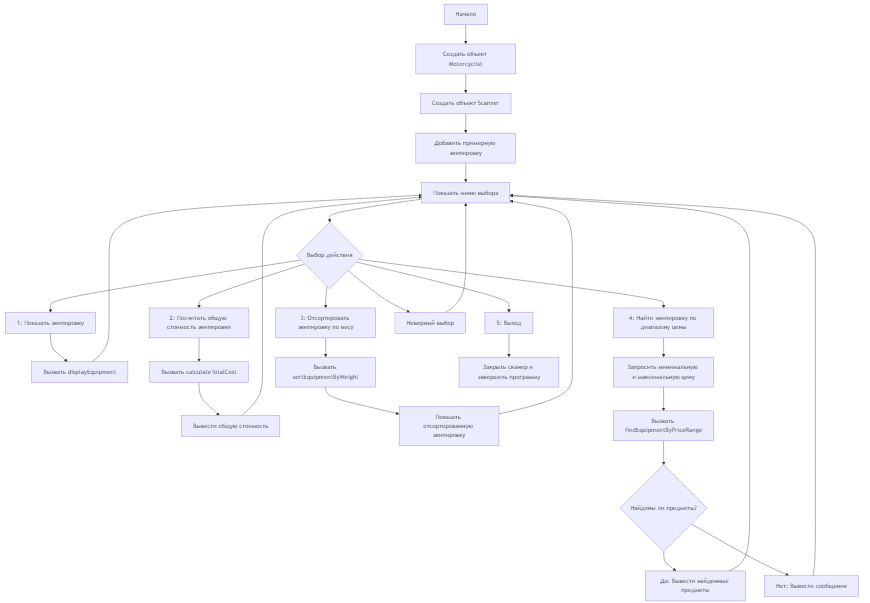


Рисунок 7 – «Блок схема 3»

Схема классов

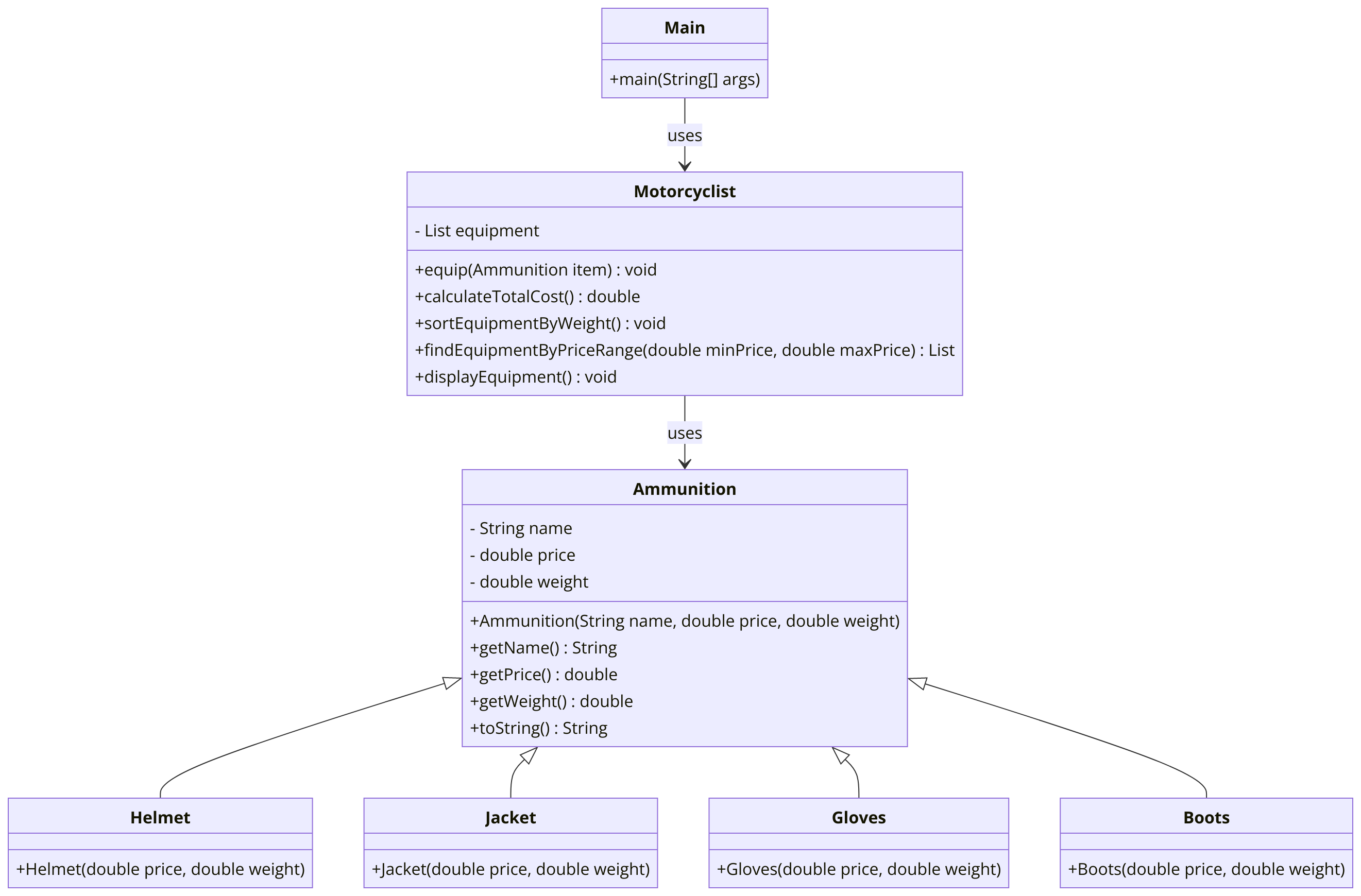


Рисунок 8 – «Схема классов 3»

Код

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;

abstract class Ammunition {

private String name;

private double price;

private double weight;

public Ammunition(String name, double price, double weight) {

this.name = name;

this.price = price;

this.weight = weight;

}

public String getName() {

return name;

}

public double getPrice() {

return price;

}

public double getWeight() {

return weight;

}

@Override

public String toString() {

return name + ": Цена = " + price + ", Вес = " + weight;

}

}

class Helmet extends Ammunition {

public Helmet(double price, double weight) {

super("Шлем", price, weight);

}

}

class Jacket extends Ammunition {

public Jacket(double price, double weight) {

super("Куртка", price, weight);

}

}

class Gloves extends Ammunition {

public Gloves(double price, double weight) {

super("Перчатки", price, weight);

}

}

class Boots extends Ammunition {

public Boots(double price, double weight) {

super("Ботинки", price, weight);

}

}

class Motorcyclist {

private List<Ammunition> equipment = new ArrayList<>();

public void equip(Ammunition item) {

equipment.add(item);

}

public double calculateTotalCost() {

double totalCost = 0;

for (Ammunition item : equipment) {

totalCost += item.getPrice();

}

return totalCost;

}

public void sortEquipmentByWeight() {

Collections.sort(equipment, Comparator.comparingDouble(Ammunition::getWeight));

}

public List<Ammunition> findEquipmentByPriceRange(double minPrice, double maxPrice) {

List<Ammunition> result = new ArrayList<>();

for (Ammunition item : equipment) {

if (item.getPrice() >= minPrice && item.getPrice() <= maxPrice) {

result.add(item);

}

}

return result;

}

public void displayEquipment() {

for (Ammunition item : equipment) {

System.out.println(item);

}

}

}

public class Main {

public static void main(String[] args) {

Motorcyclist motorcyclist = new Motorcyclist();

Scanner scanner = new Scanner(System.in);

// Примерная амуниция

motorcyclist.equip(new Helmet(150.0, 1.5));

motorcyclist.equip(new Jacket(300.0, 3.0));

motorcyclist.equip(new Gloves(50.0, 0.5));

motorcyclist.equip(new Boots(200.0, 2.0));

boolean exit = false;

while (!exit) {

System.out.println("\nМеню:");

System.out.println("1. Показать экипировку");

System.out.println("2. Посчитать общую стоимость экипировки");

System.out.println("3. Отсортировать экипировку по весу");

System.out.println("4. Найти экипировку по диапазону цены");

System.out.println("5. Выход");

System.out.print("Выберите действие: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.println("Экипировка мотоциклиста:");

motorcyclist.displayEquipment();

break;

case 2:

System.out.println("Общая стоимость экипировки: " + motorcyclist.calculateTotalCost());

break;

case 3:

motorcyclist.sortEquipmentByWeight();

System.out.println("Экипировка отсортирована по весу.");

motorcyclist.displayEquipment();

break;

case 4:

System.out.print("Введите минимальную цену: ");

double minPrice = scanner.nextDouble();

System.out.print("Введите максимальную цену: ");

double maxPrice = scanner.nextDouble();

List<Ammunition> foundItems = motorcyclist.findEquipmentByPriceRange(minPrice, maxPrice);

if (foundItems.isEmpty()) {

System.out.println("Экипировка в данном диапазоне не найдена.");

} else {

System.out.println("Экипировка в диапазоне цены:");

for (Ammunition item : foundItems) {

System.out.println(item);

}

}

break;

case 5:

exit = true;

break;

default:

System.out.println("Неверный выбор, попробуйте снова.");

}

}

scanner.close();

}

}

Результат

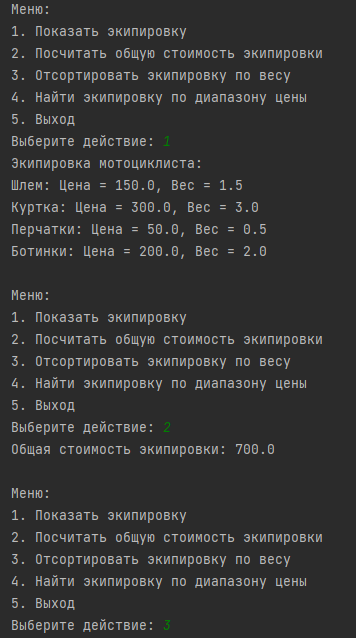


Рисунок 9 – «Результат 3»

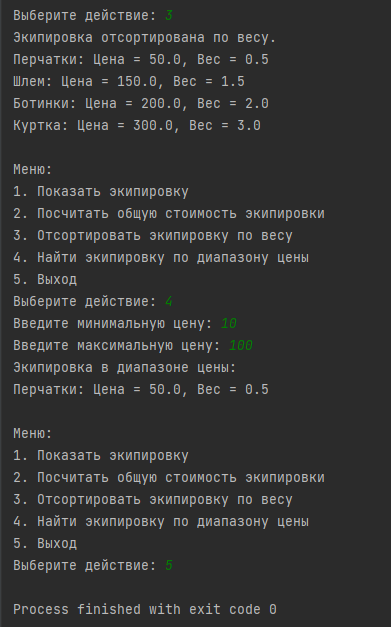


Рисунок 10– «Результат 3 продолжение»

* 1. Задача 4

Создать класс Catalog с внутренним классом, с помощью объектов которого можно хранить информацию об истории выдач книги читателям.

Таблица 5 – «Таблица идентификаторов 4»

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Описание |
| List<Book> | books | Список объектов класса Book, представляющий книги в каталоге. |
| String | title | Название книги. |
| String | author | Автор книги. |
| List<IssueHistory> | issueHistories | Список объектов класса IssueHistory, представляющий историю выдач для конкретной книги. |
| String | readerName | Имя читателя, взявшего книгу. |
| Date | issueDate | Дата выдачи книги. |
| Date | returnDate | Дата возврата книги. |
| Catalog | catalog | Объект класса Catalog, представляющий каталог книг. |
| Book | book1 | Объект класса Book, представляющий первую книгу ("Мастер и Маргарита"). |
| Book | book2 | Объект класса Book, представляющий вторую книгу ("Преступление и наказание"). |

Блок схема

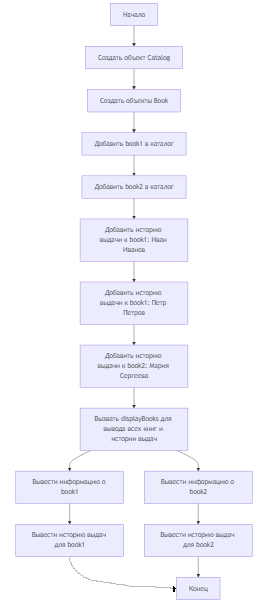


Рисунок 11 – «Блок схема 4»

Схема классов

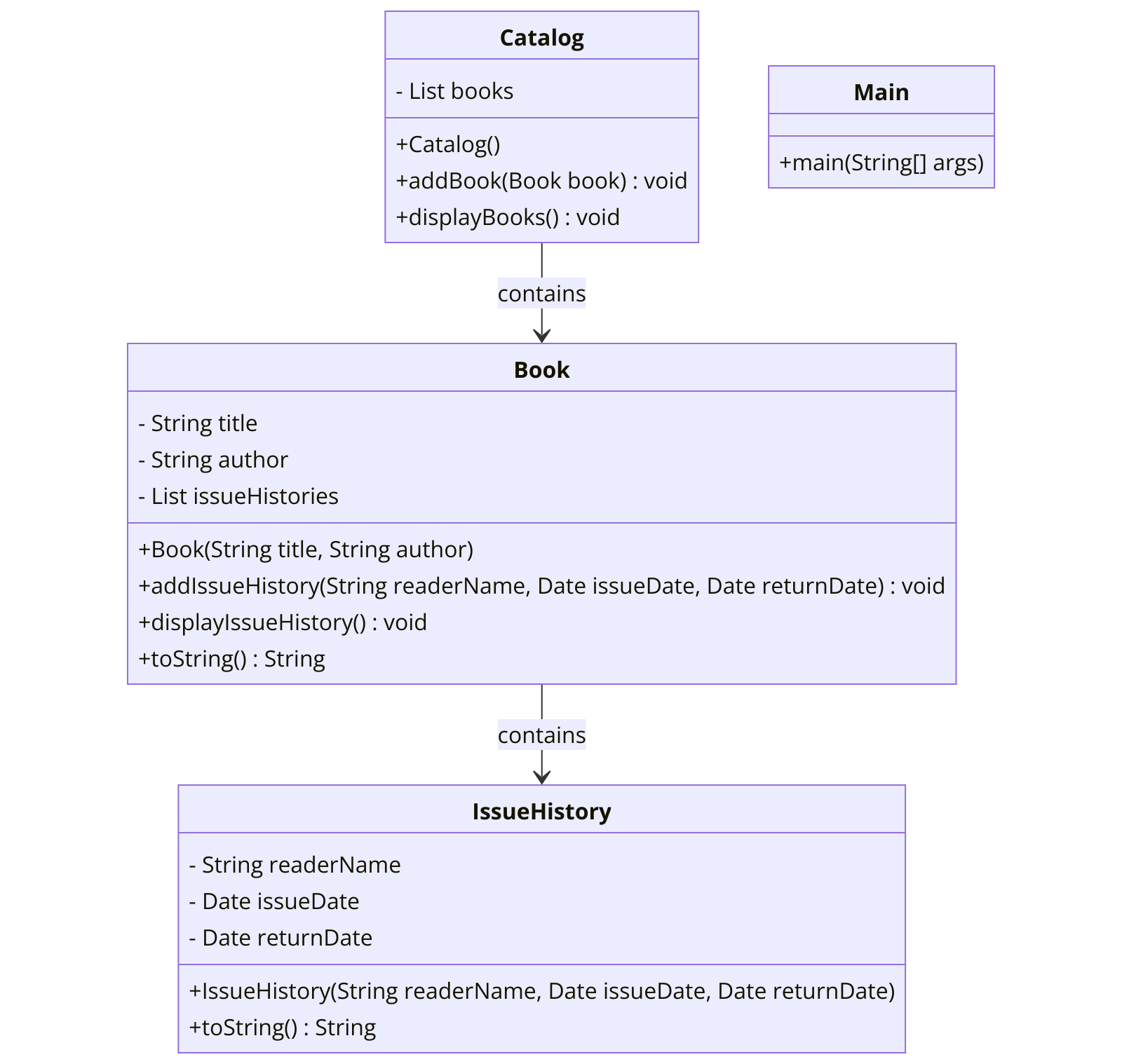


Рисунок 12 – «Схема классов 4»

Код

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class Catalog {

private List<Book> books;

public Catalog() {

this.books = new ArrayList<>();

}

public void addBook(Book book) {

books.add(book);

}

public void displayBooks() {

for (Book book : books) {

System.out.println(book);

book.displayIssueHistory();

}

}

public static class Book {

private String title;

private String author;

private List<IssueHistory> issueHistories;

public Book(String title, String author) {

this.title = title;

this.author = author;

this.issueHistories = new ArrayList<>();

}

public void addIssueHistory(String readerName, Date issueDate, Date returnDate) {

issueHistories.add(new IssueHistory(readerName, issueDate, returnDate));

}

public void displayIssueHistory() {

if (issueHistories.isEmpty()) {

System.out.println("История выдач отсутствует.");

} else {

System.out.println("История выдач книги: " + title);

for (IssueHistory history : issueHistories) {

System.out.println(history);

}

}

}

@Override

public String toString() {

return "Книга: " + title + ", Автор: " + author;

}

private static class IssueHistory {

private String readerName;

private Date issueDate;

private Date returnDate;

public IssueHistory(String readerName, Date issueDate, Date returnDate) {

this.readerName = readerName;

this.issueDate = issueDate;

this.returnDate = returnDate;

}

@Override

public String toString() {

return "Читатель: " + readerName + ", Дата выдачи: " + issueDate + ", Дата возврата: " + returnDate;

}

}

}

public static void main(String[] args) {

Catalog catalog = new Catalog();

Book book1 = new Book("Мастер и Маргарита", "Михаил Булгаков");

Book book2 = new Book("Преступление и наказание", "Фёдор Достоевский");

catalog.addBook(book1);

catalog.addBook(book2);

book1.addIssueHistory("Иван Иванов", new Date(2024, 8, 1), new Date(2024, 8, 15));

book1.addIssueHistory("Петр Петров", new Date(2024, 9, 1), new Date(2024, 9, 10));

book2.addIssueHistory("Мария Сергеева", new Date(2024, 7, 1), new Date(2024, 7, 15));

catalog.displayBooks();

}

}

Результат

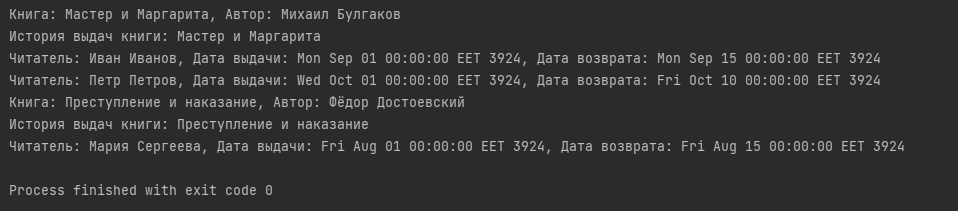


Рисунок 13 – «Результат 4»

* 1. Задача 5

Создать программу обработки текста учебника по программированию с использованием классов: Символ, Слово, Предложение, Абзац, Лексема, Листинг, Знак препинания и др. Во всех задачах с формированием текста заменять табуляции и последовательности пробелов одним пробелом. Предварительно текст следует разобрать на составные части, выполнить одно из перечисленных ниже заданий и вывести полученный результат.

Изменить программу из предыдущего примера так, чтобы она осуществляла форматирование с выравниванием по обоим краям. Для этого добавить дополнительные пробелы между словами.

Таблица 6 – «Таблица идентификаторов 5»

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Описание |
| char | character | Символ, представляющий одну букву или знак препинания в тексте. |
| List<Symbol> | symbols | Список объектов класса Symbol, представляющий символы (буквы) в слове. |
| List<Object> | components | Список компонентов предложения, который может содержать как слова (Word), так и знаки препинания (Punctuation). |
| List<Sentence> | sentences | Список объектов класса Sentence, представляющий предложения в абзаце. |
| List<Paragraph> | paragraphs | Список объектов класса Paragraph, представляющий абзацы в тексте. |
| String | text | Исходный текст, который нужно форматировать. |
| int | lineWidth | Ширина строки для выравнивания текста по обоим краям. |
| StringTokenizer | tokenizer | Объект для разбивки предложения на слова и знаки препинания. |
| StringBuilder | sb | Объект для построения строкового представления слов, предложений или абзацев. |
| String[] | sentenceArray | Массив строк, представляющий предложения в абзаце. |
| String[] | paragraphArray | Массив строк, представляющий абзацы в тексте. |
| String[] | words | Массив строк, представляющий слова в предложении. |
| int | totalLength | Общая длина всех слов в предложении без пробелов. |
| int | spacesNeeded | Количество пробелов, необходимых для выравнивания текста по ширине строки. |
| int | gapsBetweenWords | Количество промежутков между словами в предложении. |
| int | spacesPerGap | Количество пробелов, которые нужно добавить в каждый промежуток между словами для выравнивания. |
| int | extraSpaces | Дополнительные пробелы, которые нужно распределить между словами для полного выравнивания строки. |
| Listing | listing | Объект класса Listing, представляющий текст, разбитый на абзацы. |

Блок схема



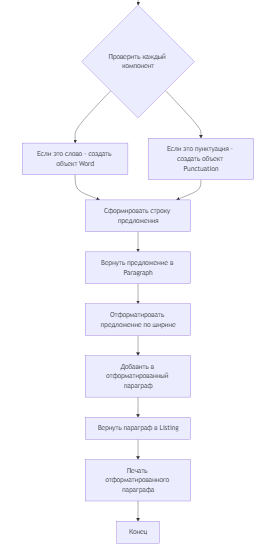


Рисунок 14 – «Блок схема 5»

Схема классов

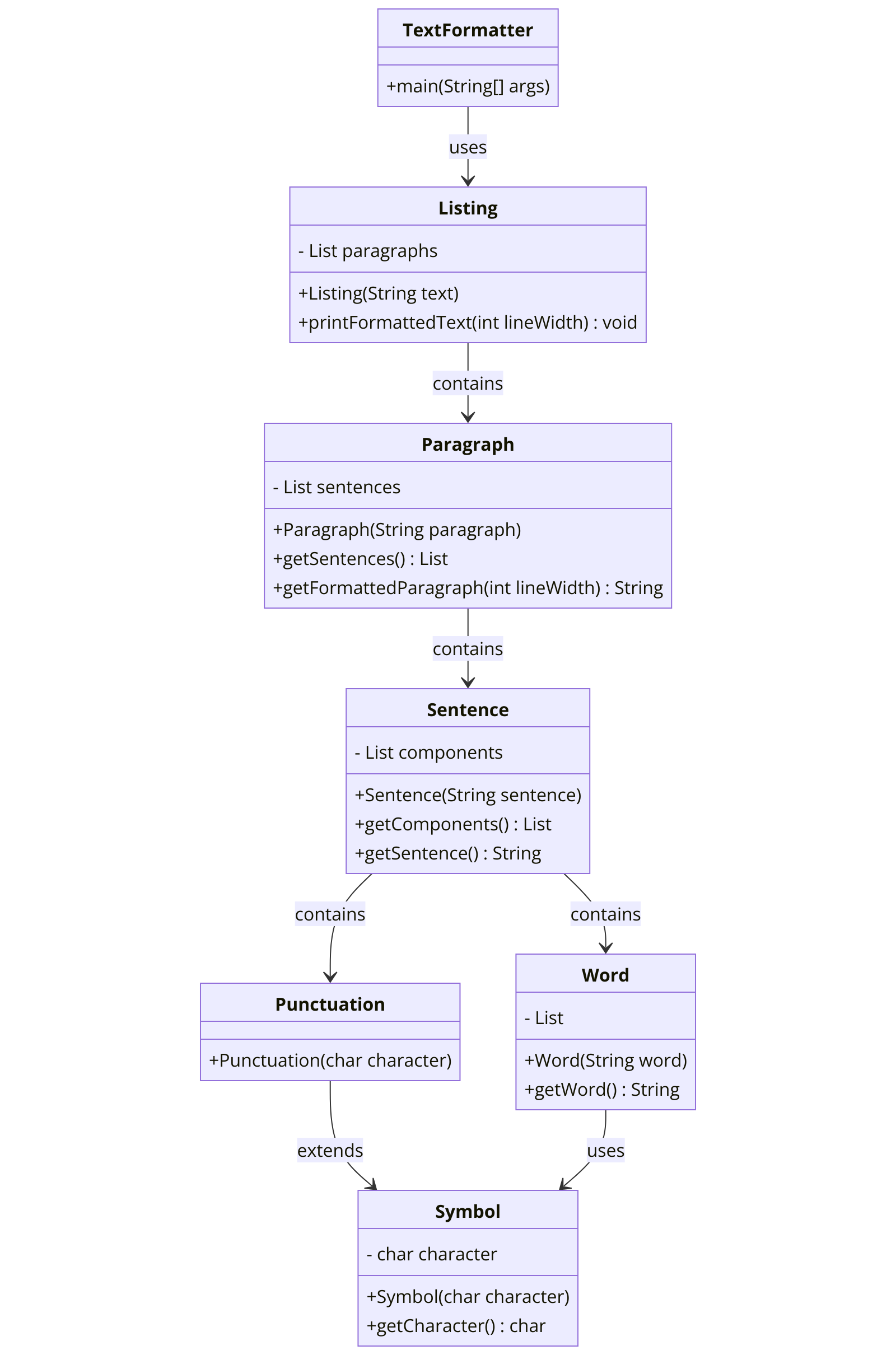


Рисунок 15 – «Схема классов 5»

Код

import java.util.ArrayList;

import java.util.List;

import java.util.StringTokenizer;

class Symbol {

private char character;

public Symbol(char character) {

this.character = character;

}

public char getCharacter() {

return character;

}

}

class Punctuation extends Symbol {

public Punctuation(char character) {

super(character);

}

}

class Word {

private List<Symbol> symbols;

public Word(String word) {

symbols = new ArrayList<>();

for (char c : word.toCharArray()) {

symbols.add(new Symbol(c));

}

}

public String getWord() {

StringBuilder sb = new StringBuilder();

for (Symbol s : symbols) {

sb.append(s.getCharacter());

}

return sb.toString();

}

}

class Sentence {

private List<Object> components;

public Sentence(String sentence) {

components = new ArrayList<>();

StringTokenizer tokenizer = new StringTokenizer(sentence, " ,.!?;:()-", true);

while (tokenizer.hasMoreTokens()) {

String token = tokenizer.nextToken();

if (",.!?;:()-".contains(token)) {

components.add(new Punctuation(token.charAt(0)));

} else if (!token.trim().isEmpty()) {

components.add(new Word(token));

}

}

}

public List<Object> getComponents() {

return components;

}

public String getSentence() {

StringBuilder sb = new StringBuilder();

for (Object component : components) {

if (component instanceof Word) {

sb.append(((Word) component).getWord()).append(" ");

} else if (component instanceof Punctuation) {

sb.append(((Punctuation) component).getCharacter());

}

}

return sb.toString().trim();

}

}

class Paragraph {

private List<Sentence> sentences;

public Paragraph(String paragraph) {

sentences = new ArrayList<>();

String[] sentenceArray = paragraph.split("(?<=[.!?])\\s+");

for (String sentence : sentenceArray) {

sentences.add(new Sentence(sentence));

}

}

public List<Sentence> getSentences() {

return sentences;

}

public String getFormattedParagraph(int lineWidth) {

StringBuilder formattedParagraph = new StringBuilder();

for (Sentence sentence : sentences) {

String formattedSentence = justifyText(sentence.getSentence(), lineWidth);

formattedParagraph.append(formattedSentence).append("\n");

}

return formattedParagraph.toString();

}

private String justifyText(String text, int lineWidth) {

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

int totalLength = 0;

for (String word : words) {

totalLength += word.length();

}

int spacesNeeded = lineWidth - totalLength;

int gapsBetweenWords = words.length - 1;

StringBuilder justifiedText = new StringBuilder();

if (gapsBetweenWords > 0) {

int spacesPerGap = spacesNeeded / gapsBetweenWords;

int extraSpaces = spacesNeeded % gapsBetweenWords;

for (int i = 0; i < words.length - 1; i++) {

justifiedText.append(words[i]);

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

justifiedText.append(" ");

}

if (extraSpaces > 0) {

justifiedText.append(" ");

extraSpaces--;

}

}

justifiedText.append(words[words.length - 1]);

} else {

justifiedText.append(text);

}

return justifiedText.toString();

}

}

class Listing {

private List<Paragraph> paragraphs;

public Listing(String text) {

paragraphs = new ArrayList<>();

String[] paragraphArray = text.split("\n\n");

for (String paragraph : paragraphArray) {

paragraphs.add(new Paragraph(paragraph));

}

}

public void printFormattedText(int lineWidth) {

for (Paragraph paragraph : paragraphs) {

System.out.println(paragraph.getFormattedParagraph(lineWidth));

}

}

}

public class TextFormatter {

public static void main(String[] args) {

String text = "Java is a popular programming language. It is widely used for building enterprise applications.\n\n"

+ "Java has many features. Some of them are platform independence and security.";

Listing listing = new Listing(text);

int lineWidth = 50;

listing.printFormattedText(lineWidth);

}

}

Результат

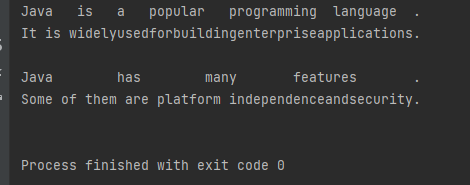


Рисунок 16 – «Результат 5»

* 1. Задача 6

Таблица идентификаторов

Блок схема

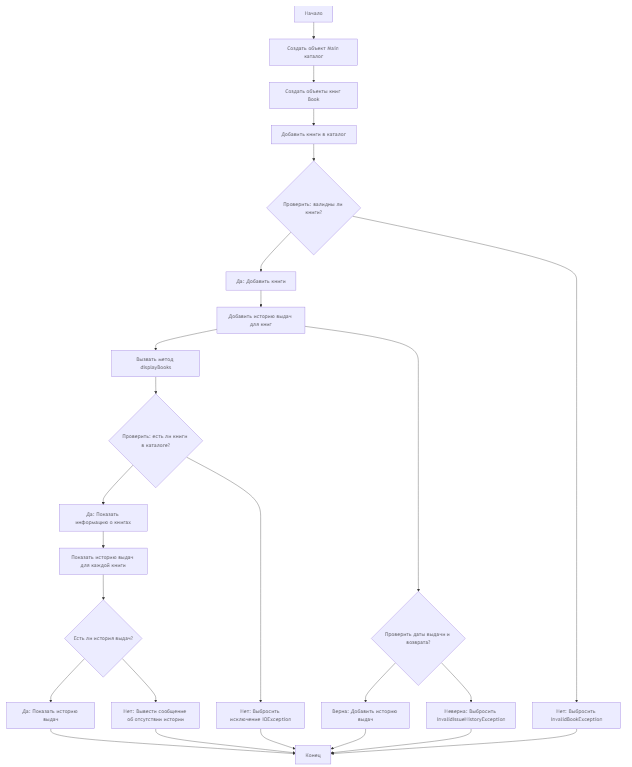


Рисунок 17 – «Блок схема 6»

Схема классов

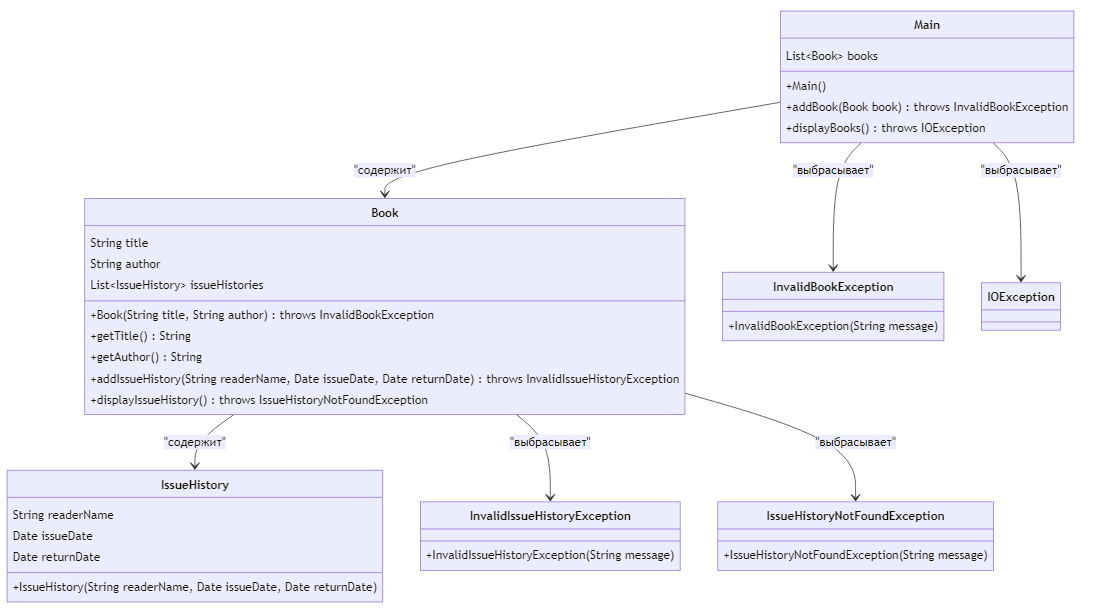


Рисунок 18 – «Блок-схема 6»

Код

import java.io.IOException;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class Main {

private List<Book> books;

public Main() {

this.books = new ArrayList<>();

}

public void addBook(Book book) throws InvalidBookException {

if (book == null || book.getTitle().isEmpty() || book.getAuthor().isEmpty()) {

throw new InvalidBookException("Недопустимое значение книги.");

}

books.add(book);

}

public void displayBooks() throws IOException {

if (books.isEmpty()) {

throw new IOException("Каталог пуст, книг нет.");

}

for (Book book : books) {

System.out.println(book);

try {

book.displayIssueHistory();

} catch (IssueHistoryNotFoundException e) {

System.err.println(e.getMessage());

}

}

}

public static class Book {

private String title;

private String author;

private List<IssueHistory> issueHistories;

public Book(String title, String author) throws InvalidBookException {

if (title == null || title.isEmpty() || author == null || author.isEmpty()) {

throw new InvalidBookException("Название книги и автор не могут быть пустыми.");

}

this.title = title;

this.author = author;

this.issueHistories = new ArrayList<>();

}

public String getTitle() {

return title;

}

public String getAuthor() {

return author;

}

public void addIssueHistory(String readerName, Date issueDate, Date returnDate) throws InvalidIssueHistoryException {

if (issueDate.after(returnDate)) {

throw new InvalidIssueHistoryException("Дата выдачи не может быть позже даты возврата.");

}

issueHistories.add(new IssueHistory(readerName, issueDate, returnDate));

}

public void displayIssueHistory() throws IssueHistoryNotFoundException {

if (issueHistories.isEmpty()) {

throw new IssueHistoryNotFoundException("История выдач отсутствует для книги: " + title);

} else {

System.out.println("История выдач книги: " + title);

for (IssueHistory history : issueHistories) {

System.out.println(history);

}

}

}

@Override

public String toString() {

return "Книга: " + title + ", Автор: " + author;

}

private static class IssueHistory {

private String readerName;

private Date issueDate;

private Date returnDate;

public IssueHistory(String readerName, Date issueDate, Date returnDate) {

this.readerName = readerName;

this.issueDate = issueDate;

this.returnDate = returnDate;

}

@Override

public String toString() {

return "Читатель: " + readerName + ", Дата выдачи: " + issueDate + ", Дата возврата: " + returnDate;

}

}

}

public static void main(String[] args) {

try {

Main catalog = new Main();

Book book1 = new Book("Мастер и Маргарита", "Михаил Булгаков");

Book book2 = new Book("Преступление и наказание", "Фёдор Достоевский");

catalog.addBook(book1);

catalog.addBook(book2);

book1.addIssueHistory("Иван Иванов", new Date(124, 8, 1), new Date(124, 8, 15));

book1.addIssueHistory("Петр Петров", new Date(124, 9, 1), new Date(124, 9, 10));

book2.addIssueHistory("Мария Сергеева", new Date(124, 7, 1), new Date(124, 7, 15));

catalog.displayBooks();

} catch (InvalidBookException | InvalidIssueHistoryException e) {

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

} catch (IOException e) {

System.err.println("Ошибка ввода/вывода: " + e.getMessage());

} catch (OutOfMemoryError e) {

System.err.println("Недостаточно памяти для обработки: " + e.getMessage());

}

}

}

// Классы пользовательских исключений

class InvalidBookException extends Exception {

public InvalidBookException(String message) {

super(message);

}

}

class InvalidIssueHistoryException extends Exception {

public InvalidIssueHistoryException(String message) {

super(message);

}

}

class IssueHistoryNotFoundException extends Exception {

public IssueHistoryNotFoundException(String message) {

super(message);

}

}

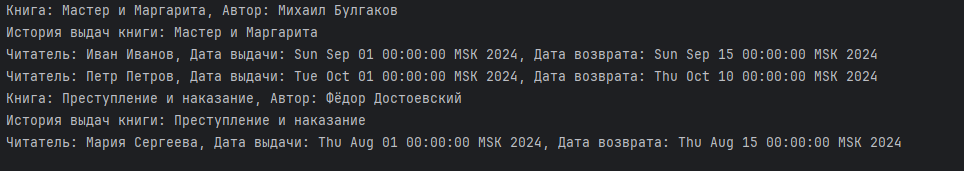
Результат

Рисунок 19 – «Результат 6»

* 1. Задача 7

В треугольной пирамиде построить сечение, проходящее через боковое ребро и медиану основания.

Таблица 8 – «Таблица идентификаторов 7»

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Описание |
| double | x, y, z | Координаты точки в 3D пространстве. |
| Point3D | start, end | Начальная и конечная точки для отрезка (Edge). |
| Point3D | vertex1, vertex2, vertex3 | Вершины треугольника (Triangle). |
| Point3D | apex | Вершина пирамиды (Pyramid). |
| Triangle | base | Основание пирамиды, представленное как треугольник (Triangle). |
| Edge | sideEdge | Боковое ребро пирамиды (Pyramid). |
| Edge | median | Медиана основания треугольника (Triangle). |
| List<Point3D> | points | Список точек, представляющих сечение (Section). |
| Section | section | Объект класса Section, представляющий сечение, проходящее через боковое ребро и медиану основания пирамиды. |
| Point3D | apex | Вершина пирамиды. |
| Point3D | baseVertex1, baseVertex2, baseVertex3 | Вершины основания пирамиды. |

Блок схема

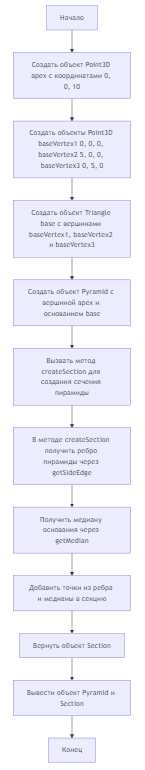


Рисунок 20 – «Блок схема 7»

Схема классов

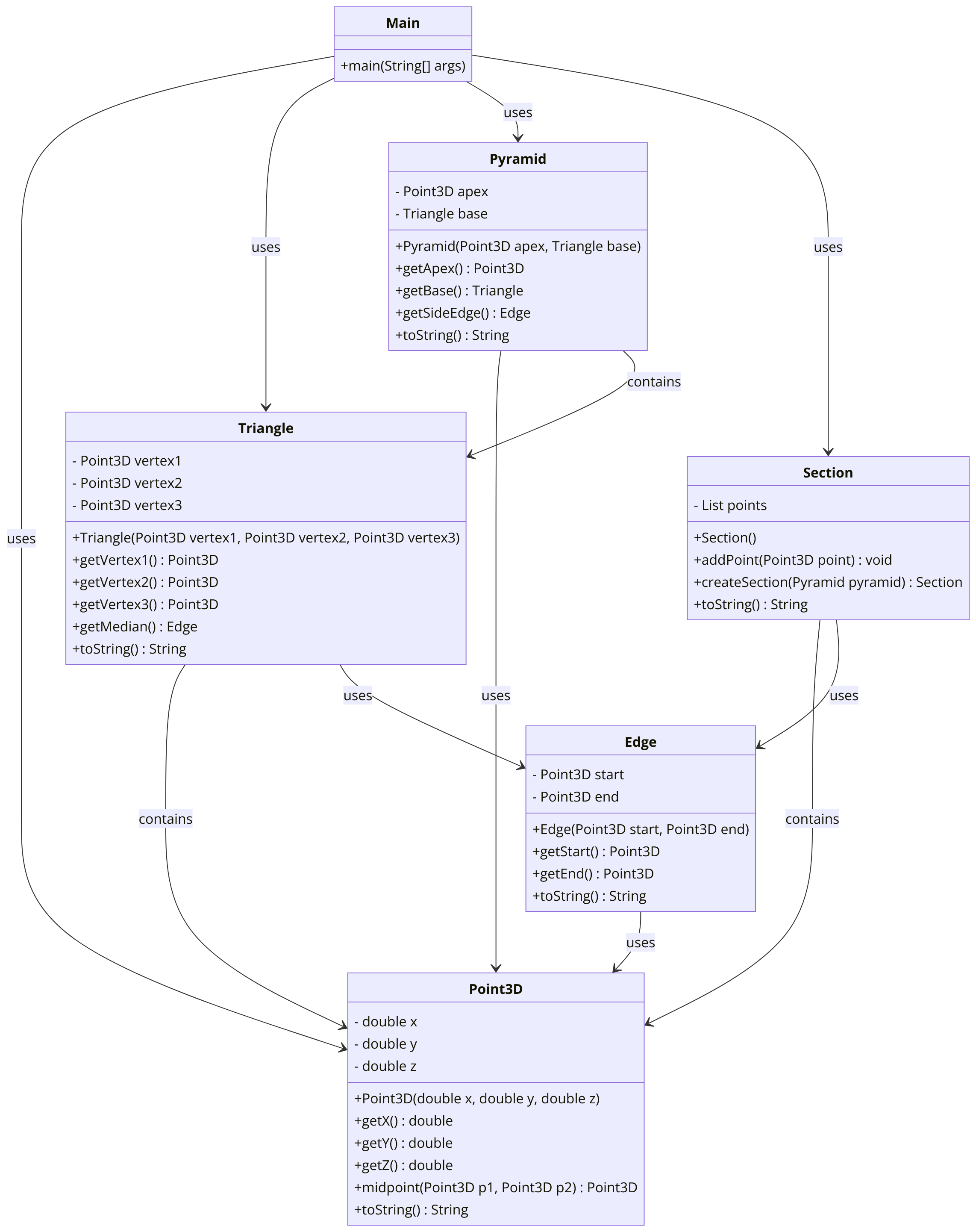


Рисунок 21– «Блок-схема 7»

Код

package geometry;

public class Point3D {

private double x, y, z;

public Point3D(double x, double y, double z) {

this.x = x;

this.y = y;

this.z = z;

}

public double getX() {

return x;

}

public double getY() {

return y;

}

public double getZ() {

return z;

}

public static Point3D midpoint(Point3D p1, Point3D p2) {

return new Point3D((p1.getX() + p2.getX()) / 2, (p1.getY() + p2.getY()) / 2, (p1.getZ() + p2.getZ()) / 2);

}

@Override

public String toString() {

return "(" + x + ", " + y + ", " + z + ")";

}

}

package geometry;

public class Edge {

private Point3D start;

private Point3D end;

public Edge(Point3D start, Point3D end) {

this.start = start;

this.end = end;

}

public Point3D getStart() {

return start;

}

public Point3D getEnd() {

return end;

}

@Override

public String toString() {

return "Edge from " + start + " to " + end;

}

}

package geometry;

public class Triangle {

private Point3D vertex1;

private Point3D vertex2;

private Point3D vertex3;

public Triangle(Point3D vertex1, Point3D vertex2, Point3D vertex3) {

this.vertex1 = vertex1;

this.vertex2 = vertex2;

this.vertex3 = vertex3;

}

public Point3D getVertex1() {

return vertex1;

}

public Point3D getVertex2() {

return vertex2;

}

public Point3D getVertex3() {

return vertex3;

}

public Edge getMedian() {

Point3D midpoint = Point3D.midpoint(vertex2, vertex3);

return new Edge(vertex1, midpoint);

}

@Override

public String toString() {

return "Triangle with vertices " + vertex1 + ", " + vertex2 + ", " + vertex3;

}

}

package geometry;

public class Pyramid {

private Point3D apex;

private Triangle base;

public Pyramid(Point3D apex, Triangle base) {

this.apex = apex;

this.base = base;

}

public Point3D getApex() {

return apex;

}

public Triangle getBase() {

return base;

}

public Edge getSideEdge() {

return new Edge(apex, base.getVertex1());

}

@Override

public String toString() {

return "Pyramid with apex " + apex + " and base " + base;

}

}

package geometry;

import java.util.ArrayList;

import java.util.List;

public class Section {

private List<Point3D> points;

public Section() {

points = new ArrayList<>();

}

public void addPoint(Point3D point) {

points.add(point);

}

@Override

public String toString() {

return "Section points: " + points;

}

public static Section createSection(Pyramid pyramid) {

Section section = new Section();

Edge sideEdge = pyramid.getSideEdge();

Edge median = pyramid.getBase().getMedian();

section.addPoint(sideEdge.getStart());

section.addPoint(sideEdge.getEnd());

section.addPoint(median.getStart());

section.addPoint(median.getEnd());

return section;

}

}

package geometry;

public class Main {

public static void main(String[] args) {

Point3D apex = new Point3D(0, 0, 10); // Вершина пирамиды

Point3D baseVertex1 = new Point3D(0, 0, 0);

Point3D baseVertex2 = new Point3D(5, 0, 0);

Point3D baseVertex3 = new Point3D(0, 5, 0);

Triangle base = new Triangle(baseVertex1, baseVertex2, baseVertex3);

Pyramid pyramid = new Pyramid(apex, base);

Section section = Section.createSection(pyramid);

System.out.println(pyramid);

System.out.println(section);

}

}

Результат

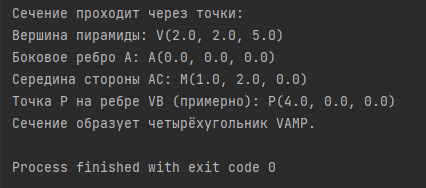


Рисунок 22 – «Результат 7»

* 1. Задача 8

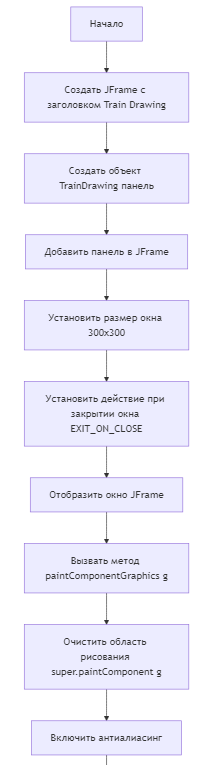
Составить программу получения на экране рисунков, изображенных в таблице.

Паровозик.

Таблица 9 – «Таблица идентификаторов 8»

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Определение |
| Graphics | g | Объект класса Graphics, используемый для отрисовки фигур и текста на компонентах. |
| Graphics2D | g2d | Объект класса Graphics2D, который является расширением Graphics и предоставляет более сложные функции рисования. |
| JFrame | frame | Основное окно приложения, содержащее панель для рисования (TrainDrawing). |
| TrainDrawing | panel | Объект класса TrainDrawing, представляющий панель для отрисовки поезда. |

Блок схема



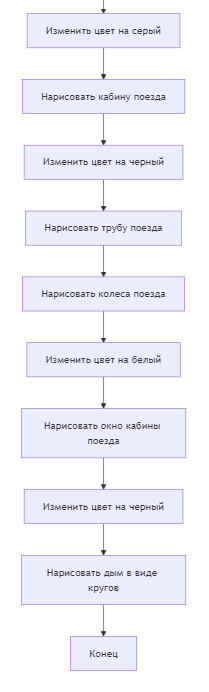


Рисунок 23 – «Блок схема 8»

Схема классов

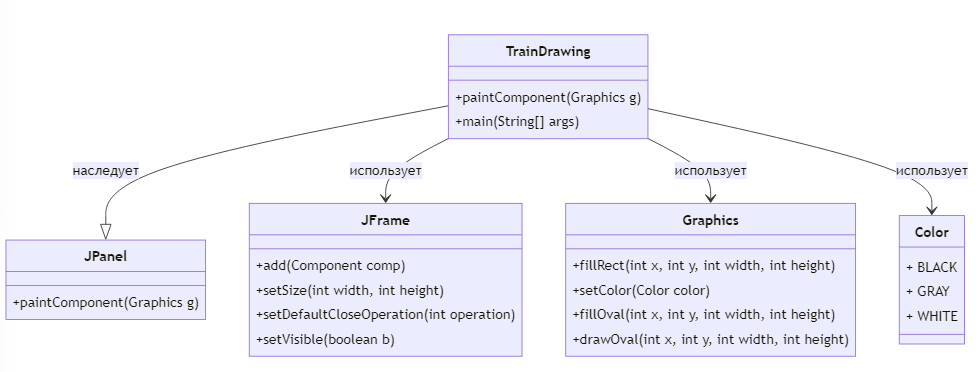


Рисунок 24 – «Схема классов 8»

Код

import javax.swing.\*;

import java.awt.\*;

public class TrainDrawing extends JPanel {

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

Graphics2D g2d = (Graphics2D) g;

g2d.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2d.fillRect(50, 100, 150, 50);

g2d.setColor(Color.GRAY);

g2d.fillRect(150, 60, 50, 90);

g2d.setColor(Color.BLACK);

g2d.fillRect(80, 60, 20, 40);

g2d.setColor(Color.BLACK);

g2d.fillOval(60, 140, 30, 30);

g2d.fillOval(100, 140, 30, 30);

g2d.fillOval(140, 140, 30, 30);

g2d.fillOval(180, 140, 30, 30);

g2d.setColor(Color.WHITE);

g2d.fillRect(160, 70, 20, 20);

g2d.setColor(Color.BLACK);

g2d.drawOval(90, 30, 20, 20);

g2d.drawOval(110, 20, 20, 20);

g2d.drawOval(130, 10, 20, 20);

g2d.drawOval(150, 0, 20, 20);

}

public static void main(String[] args) {

JFrame frame = new JFrame("Train Drawing");

TrainDrawing panel = new TrainDrawing();

frame.add(panel);

frame.setSize(300, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

Результат

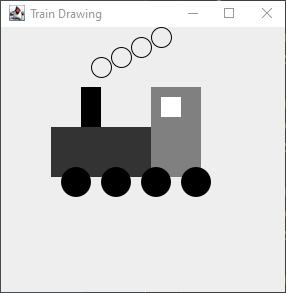


Рисунок 25– «Результат 8»

* 1. Задача 9

Игра Математико

Таблица 10 – «Таблица идентификаторов 9»

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Определение |
| int | GRID\_SIZE | Константа, представляющая размер игрового поля (5x5). |
| int | DECK\_SIZE | Константа, представляющая количество карт в колоде (52 карты). |
| int[] | CARD\_NUMBERS | Массив, представляющий номера на картах (от 1 до 13). |
| int[][] | grid | Двумерный массив, представляющий игровое поле размером 5x5. |
| List<Integer> | deck | Список, представляющий колоду карт, из которой можно вытаскивать карты в случайном порядке. |
| Random | random | Объект класса Random, используемый для генерации случайных чисел при выборе карты и размещении на поле. |
| int | totalScore | Переменная для хранения общего счета игры, вычисленного на основе конфигурации игрового поля. |
| int[] | column | Временный массив для хранения чисел в столбце при вычислении счета за столбец. |
| int[] | diagonal1 | Массив для хранения чисел первой диагонали (с верхнего левого угла до нижнего правого угла). |
| int[] | diagonal2 | Массив для хранения чисел второй диагонали (с верхнего правого угла до нижнего левого угла). |
| Map<Integer, Integer> | countMap | Словарь для подсчета количества вхождений каждого числа в строке (строке, столбце или диагонали). |
| int | score | Переменная для временного хранения счета за одну строку (строку, столбец или диагональ) на игровом поле. |

Блок схема

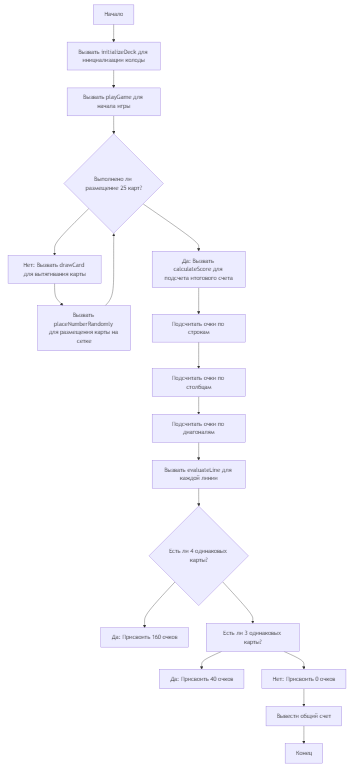


Рисунок 26 – «Блок схема 9»

Схема классов

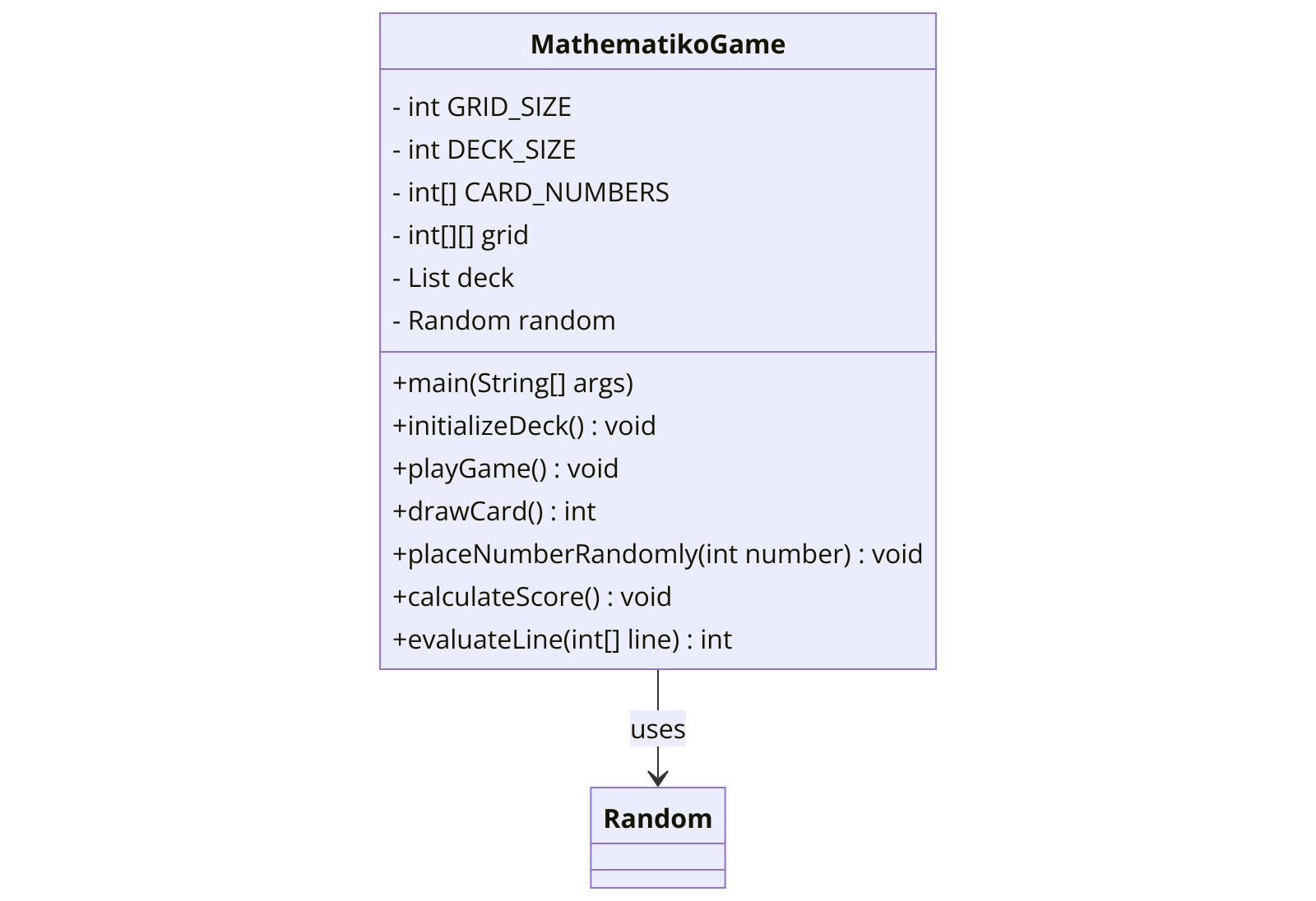


Рисунок 27 – «Схема классов 9»

Код

import java.util.\*;

public class MathematikoGame {

static final int GRID\_SIZE = 5;

static final int DECK\_SIZE = 52;

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

static int[][] grid = new int[GRID\_SIZE][GRID\_SIZE];

static List<Integer> deck = new ArrayList<>();

static Random random = new Random();

public static void main(String[] args) {

initializeDeck();

playGame();

calculateScore();

}

public static void initializeDeck() {

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

for (int num : CARD\_NUMBERS) {

deck.add(num);

}

}

Collections.shuffle(deck);

}

public static void playGame() {

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

int card = drawCard();

System.out.println("Card drawn: " + card);

placeNumberRandomly(card);

}

}

public static int drawCard() {

return deck.remove(random.nextInt(deck.size()));

}

public static void placeNumberRandomly(int number) {

while (true) {

int row = random.nextInt(GRID\_SIZE);

int col = random.nextInt(GRID\_SIZE);

if (grid[row][col] == 0) {

grid[row][col] = number;

System.out.println("Placed number " + number + " at (" + row + ", " + col + ")");

break;

}

}

}

public static void calculateScore() {

int totalScore = 0;

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

totalScore += evaluateLine(grid[i]);

}

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

int[] column = new int[GRID\_SIZE];

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

column[i] = grid[i][j];

}

totalScore += evaluateLine(column);

}

int[] diagonal1 = new int[GRID\_SIZE];

int[] diagonal2 = new int[GRID\_SIZE];

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

diagonal1[i] = grid[i][i];

diagonal2[i] = grid[i][GRID\_SIZE - i - 1];

}

totalScore += evaluateLine(diagonal1);

totalScore += evaluateLine(diagonal2);

System.out.println("Total Score: " + totalScore);

}

public static int evaluateLine(int[] line) {

int score = 0;

Map<Integer, Integer> countMap = new HashMap<>();

for (int num : line) {

countMap.put(num, countMap.getOrDefault(num, 0) + 1);

}

if (countMap.containsValue(4)) {

score = 160;

} else if (countMap.containsValue(3)) {

score = 40;

}

return score;

}

}

Результат

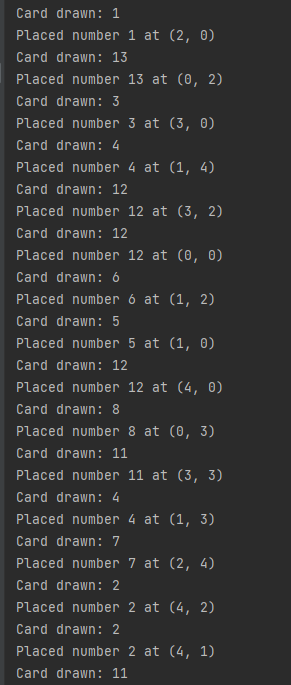


Рисунок 28 – «Результат 9»

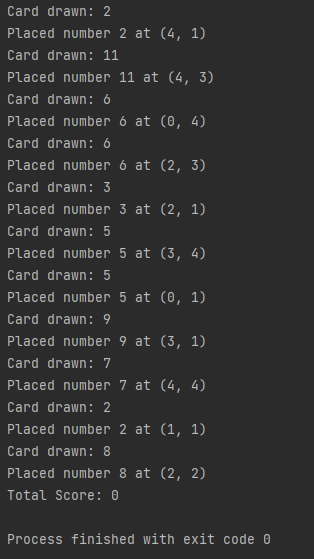


Рисунок 29 – «Результат 9 продолжение»

* 1. Задание 10

Задание 1 Вариант 1

Использование оператора N := N \* 2 можно реализовать через логический сдвиг влево. Код на ассемблере:  
Код

section .data

N dd 5

section .text

global\_start

\_start:

mov eax, [N]

EAX shl eax, 1

mov [N], eax

mov eax, 60

xor edi, edi

syscall

Задание 2 Вариант 1

Реализация сложения через стек:

Код

section .data

x dd 20

a dd 20

section .text

global \_start

\_start:

push ax

push bx

mov eax, [x]

eax mov ebx, [a]

add eax, ebx

mov [x], eax

bx pop ax

mov eax, 60

xor edi, edi

syscall

1. Разработка АИС
   1. Задание 1,2,3  
      ER – диаграмма

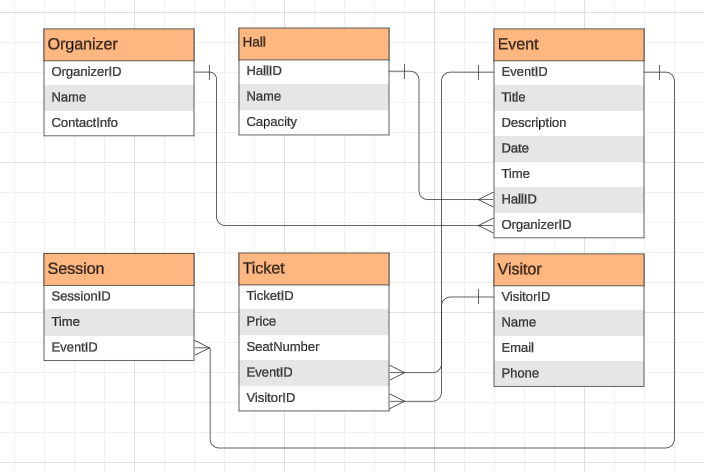


Рисунок 30 – «Диаграмма данных»

2.2 Задание 4

1. Диаграмма вариантов использования (Use Case Diagram)

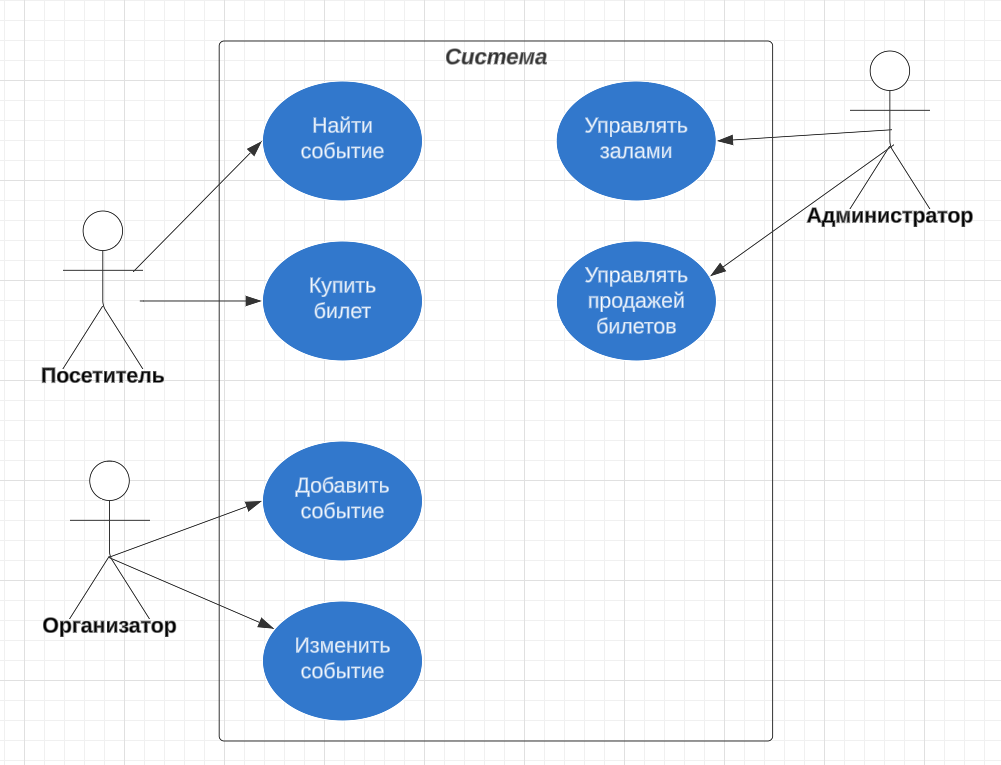


Рисунок 31 – «Диаграмма вариантов»

1. Диаграмма классов (Class Diagram)

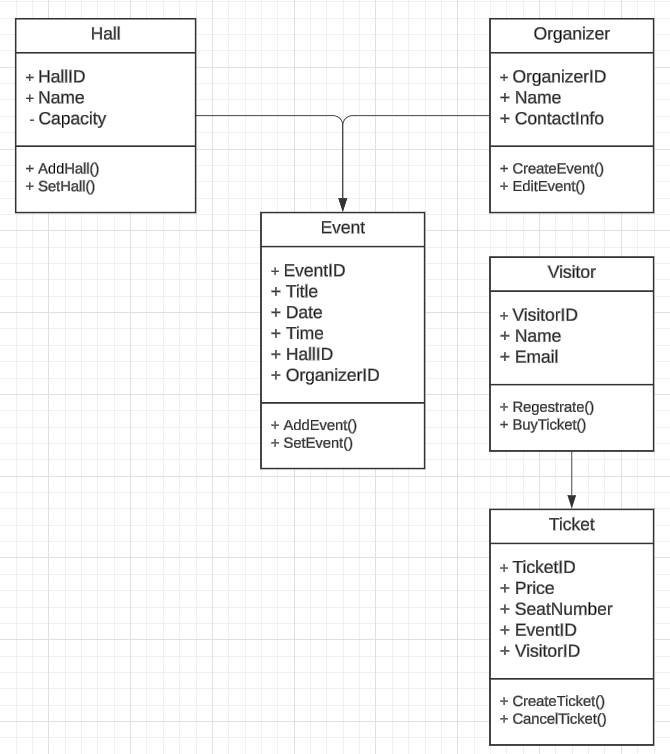


Рисунок 32 – «Диаграмма классов киноконцертного комплекса»

1. Диаграмма последовательностей (Sequence Diagram)

Покупка билета

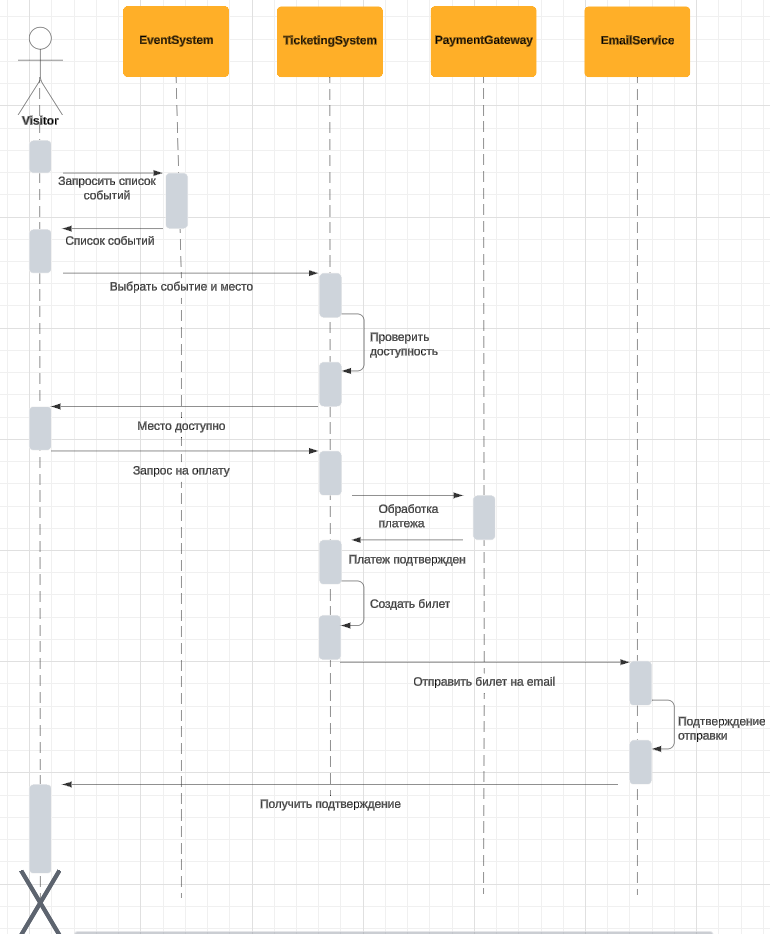


Рисунок 33 – «Диаграмма последовательностей»

1. Диаграмма кооперации (Collaboration Diagram)

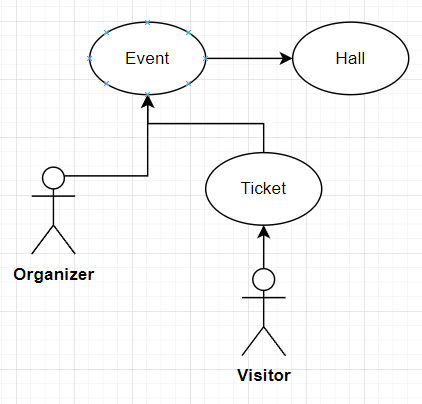


Рисунок 34 – «Диаграмма кооперации»

1. Диаграмма состояний (State Diagram)

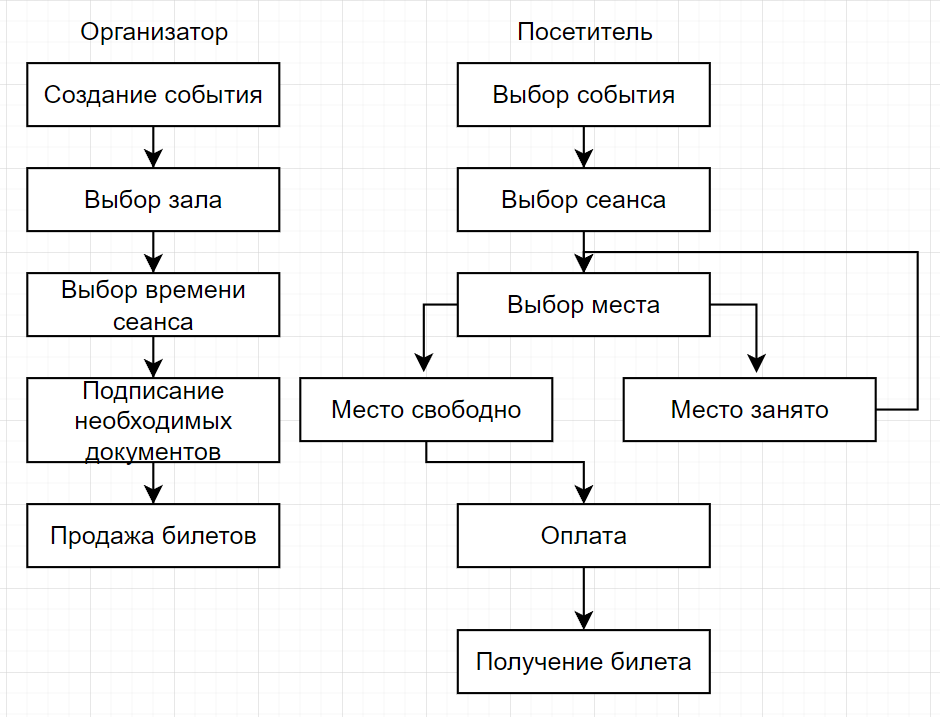


Рисунок 35 – «Диаграмма состояний»

1. Диаграмма деятельности (Activity Diagram)



Рисунок 36 – «Диаграмма деятельности»

2.3 Задание 5

1. Общие сведения

1.1. Наименование системы: Информационная система управления киноконцертным комплексом (ИС ККК). 1.2. Заказчик: Киноконцертный комплекс (наименование организации). 1.3. Разработчик: Команда разработчиков программного обеспечения (наименование организации или отдела). 1.4. Цель разработки: Автоматизация процессов управления мероприятиями, продажа билетов, управление залами и взаимодействие с посетителями, организаторами и администраторами киноконцертного комплекса. 1.5. Основание для разработки: Решение руководства киноконцертного комплекса о необходимости внедрения системы для повышения эффективности бизнес-процессов и улучшения пользовательского опыта. 1.6. Используемые стандарты: ГОСТ 34.601-90, ГОСТ 34.602-89 (ЕСПД), другие стандарты и нормативные документы, действующие в области проектирования информационных систем.

2. Назначение системы

ИС ККК предназначена для:

* Управления мероприятиями: создание, модерация, публикация, отмена.
* Управления залами: резервирование залов, контроль их состояния.
* Продажи билетов: резервирование, оплата, аннулирование билетов.

Взаимодействия с организаторами мероприятий и посетителями комплекса.

Сбора статистики и отчетов для администрации.

3. Требования к системе

3.1. Требования к функциональности:

* Создание и управление мероприятиями организаторами.
* Модерация мероприятий администраторами.
* Управление залами для мероприятий.
* Продажа и резервирование билетов посетителями через веб-интерфейс.
* Поддержка системы оплаты и возврата билетов.
* Интеграция с внешними платёжными системами.
* Уведомление посетителей о статусе мероприятий и билетов (email, SMS).
* Формирование отчетности для организаторов и администраторов.

3.2. Требования к надежности:

* Обеспечение высокой доступности системы (не менее 99% времени).
* Гарантированная сохранность данных в случае сбоев.

3.3. Требования к интерфейсу:

* Веб-интерфейс для организаторов, посетителей и администраторов.
* Удобная навигация и интуитивно понятные элементы интерфейса.

3.4. Требования к безопасности:

* Аутентификация и авторизация пользователей.
* Шифрование данных при передаче (SSL/TLS).
* Защита от несанкционированного доступа и атак (SQL-инъекции, XSS и т.д.).

3.5. Требования к техническим характеристикам:

* Поддержка многопользовательского режима.
* Производительность: система должна обеспечивать обработку до 5000 билетов в день.
* Масштабируемость: возможность расширения функционала системы.

4. Требования к документированию

* Руководство пользователя.
* Руководство администратора.
* Техническая документация для разработчиков (описание API, структура базы данных).

5. Этапы и сроки разработки

Этап 1: Анализ предметной области — 1 месяц.

Этап 2: Разработка и согласование ТЗ — 2 недели.

Этап 3: Проектирование архитектуры и базы данных — 1 месяц.

Этап 4: Разработка системы — 3 месяца.

Этап 5: Тестирование — 1 месяц.

Этап 6: Внедрение и обучение персонала — 2 недели.

6. Порядок приемки

Приемка будет производиться поэтапно после выполнения каждого этапа разработки.

Финальная приемка — после тестирования и предоставления документации.

2.4 Спецификация системы и алгоритмы

2.4.1 Спецификация системы

Входные данные:

* Данные о мероприятиях (название, дата, время, организатор, описание).
* Информация о залах (вместимость, доступность).
* Данные о посетителях (профиль пользователя, заказы билетов).
* Данные о билетах (статус, информация об оплате, QR-код для входа).
* Информация об оплате (данные о платежах).

Выходные данные:

* Информация о доступных мероприятиях и билетах для пользователей.
* Подтверждение бронирования и оплаты билетов.
* Уведомления для посетителей (о статусе билетов, мероприятиях).
* Отчёты для организаторов и администраторов (количество проданных билетов, статистика посещаемости).
* Данные для интеграции с платёжными системами и отчеты по транзакциям.

2.4.2 Основной алгоритм учета (укрупненная блок-схема)

Запуск системы: Инициализация системы и подключение к базе данных.

Авторизация пользователя:

* Проверка прав доступа (организатор, администратор, посетитель).

Основной процесс:

* Для организатора: создание, редактирование и управление мероприятиями.
* Для администратора: модерация мероприятий, управление залами.
* Для посетителя: просмотр мероприятий, бронирование и покупка билетов.

Обработка данных:

* Создание/обновление данных в базе.
* Взаимодействие с платёжными системами (оплата, возврат средств).
* Уведомление пользователей.

Отчеты и статистика:

* Формирование отчётов по билетам и мероприятиям.
* Генерация данных для администратора и организаторов.

Завершение работы:

* Выход из системы.
* Завершение сеанса.

2.4.3 Детализация функции системы: Покупка билета

Алгоритм покупки билета:

* Посетитель выбирает мероприятие.
* Отображение списка доступных мероприятий.
* Выбор конкретного мероприятия и места в зале.

Проверка доступности билета:

* Система проверяет наличие свободных мест на выбранное мероприятие.
* Если место доступно, пользователь может продолжить.

Оформление заказа:

* Посетитель вводит данные для оформления заказа.
* Система создает временную бронь на билет.

Оплата билета:

* Перенаправление посетителя на платёжный шлюз.
* Подтверждение успешной оплаты.

Подтверждение покупки:

* После успешной оплаты билет помечается как "Оплачен".
* Система отправляет билет на электронную почту посетителя.

Завершение:

* Билет сохраняется в системе с уникальным QR-кодом для сканирования на входе.

Заключение

Техническое задание и алгоритмы работы системы обеспечивают формализацию требований и процессов для разработки информационной системы киноконцертного комплекса. Система должна автоматизировать основные процессы взаимодействия с посетителями, организаторами и администраторами, предоставляя удобные инструменты для управления мероприятиями и билетами.

2.5 Разработка структурной и функциональной схемы АИС

Автоматизированная информационная система "Киноконцертный Комплекс" разработана с использованием C# в среде Unity и базируется на модульной архитектуре. Структура системы включает следующие основные модули:

* Модуль авторизации и регистрации: обеспечивает безопасность доступа к системе, разделяя права клиентов и организаторов.
* Модуль управления мероприятиями: доступен только для организаторов и позволяет создавать, редактировать и удалять события.
* Модуль покупки билетов: предназначен для клиентов, предоставляя интерфейс для выбора и приобретения билетов.
* Модуль просмотра текущих событий: отображает все доступные мероприятия для клиентов.
* Модуль управления купленными билетами: позволяет клиентам просматривать и управлять своими покупками.
* База данных MySQL: хранит информацию о пользователях, мероприятиях и билетах.

2.6 Проектирование и реализация интерфейса

Интерфейс пользователя был разработан с упором на интуитивность и простоту использования. Использовались следующие инструментальные средства:

* Среда разработки: Visual Studio с интеграцией Unity.
* Язык программирования: C# для логики приложения.
* Unity UI Toolkit: для создания визуальных элементов интерфейса.

Основные окна приложения:

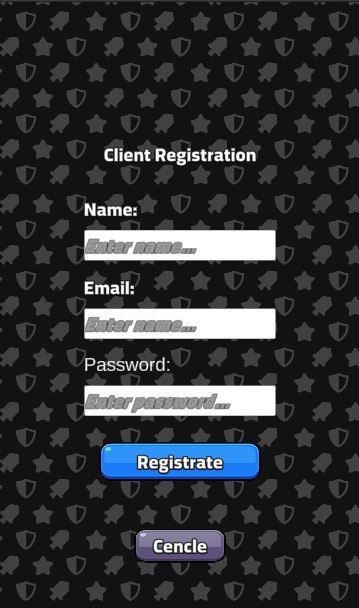


Рисунок 37 – «Окно регистрации клиента»

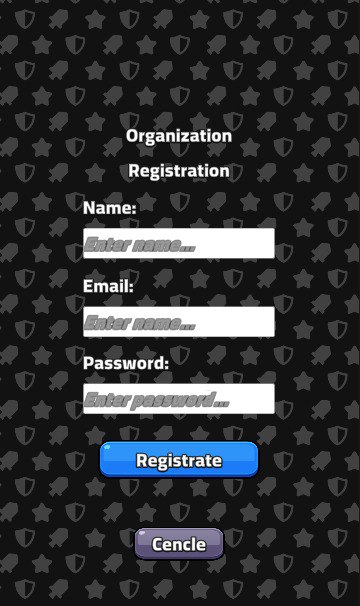


Рисунок 38 – «Окно регистрации организации»

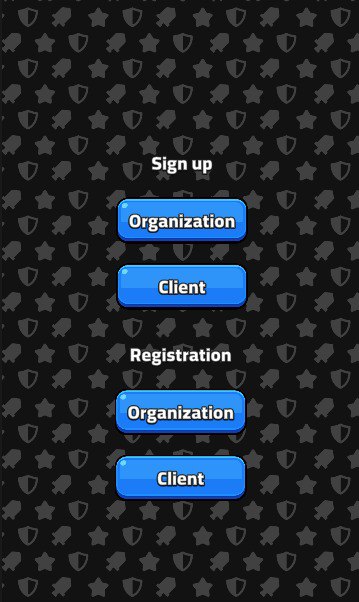


Рисунок 39 – «Окно выбора входа»



Рисунок 40 – «Окно текущих событий»



Рисунок 41 – «Окно купленных билетов»

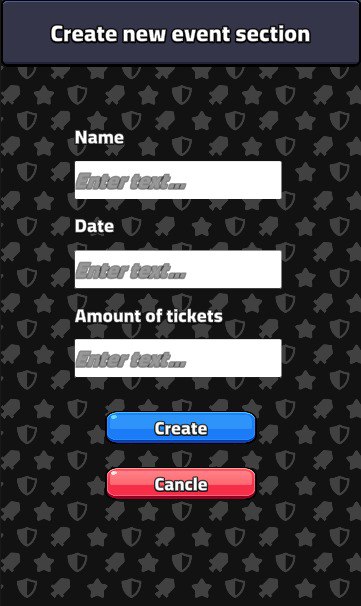


Рисунок 42 – «Окно создания события»



Рисунок 43 – «Окно покупки билетов»

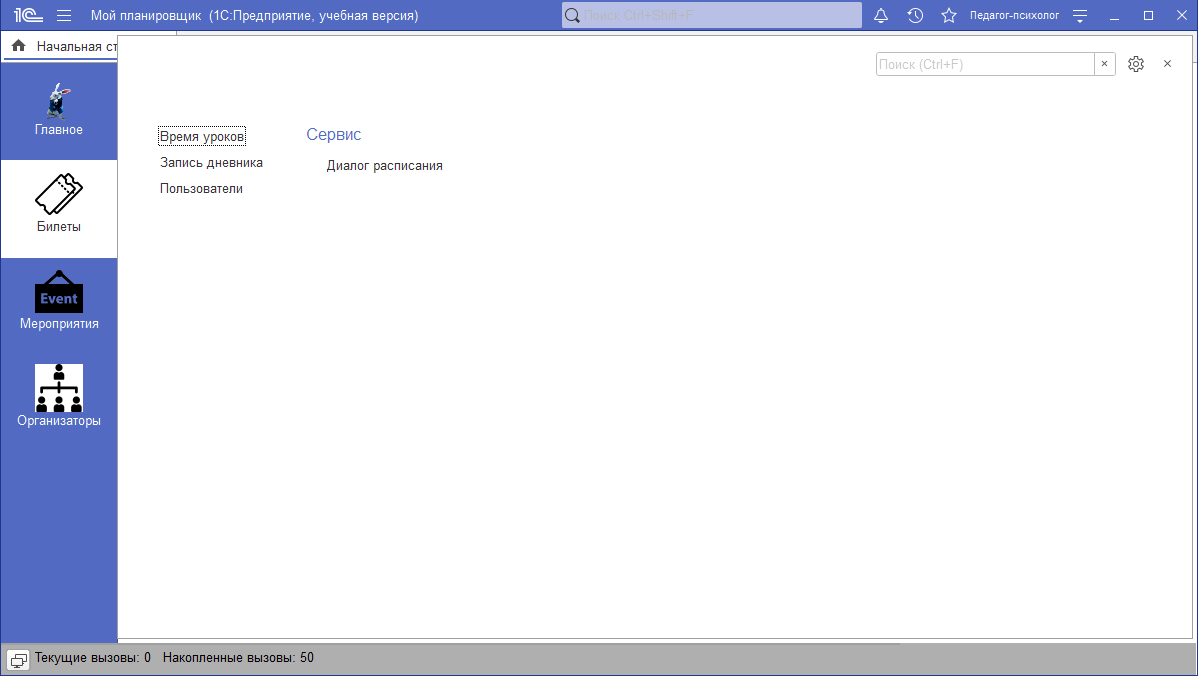
Окна приложения на 1C:  


Рисунок 44 – «Окно билеты 1C»

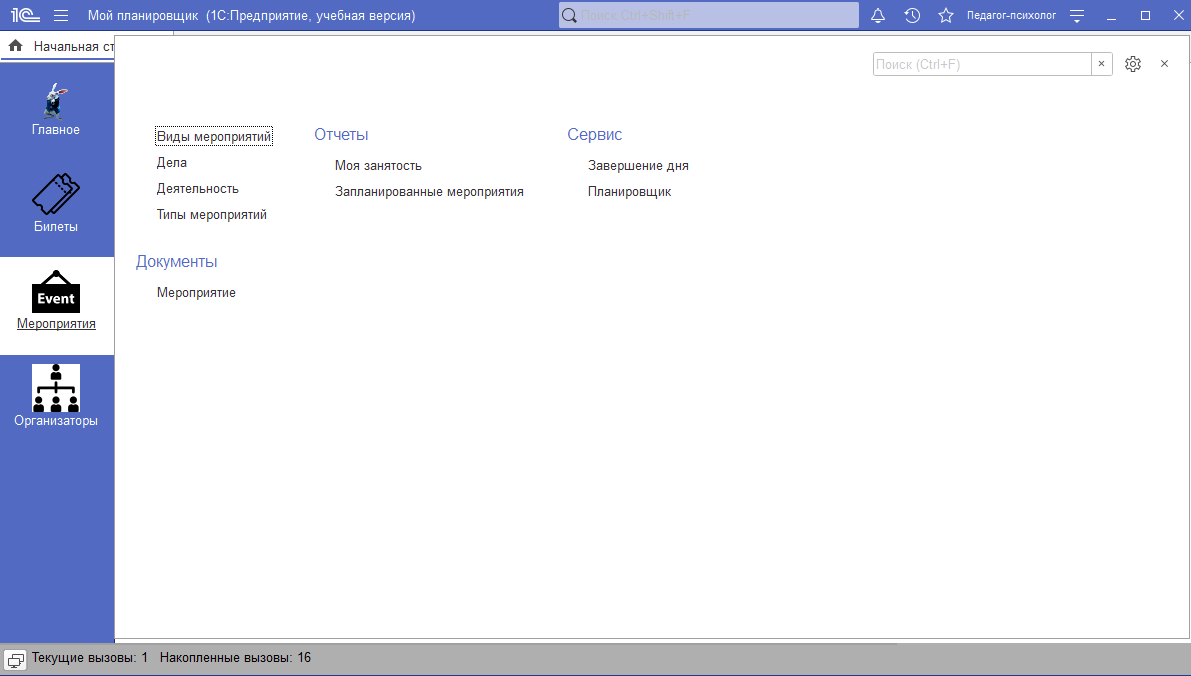


Рисунок 45 – «Окно мероприятия 1C»

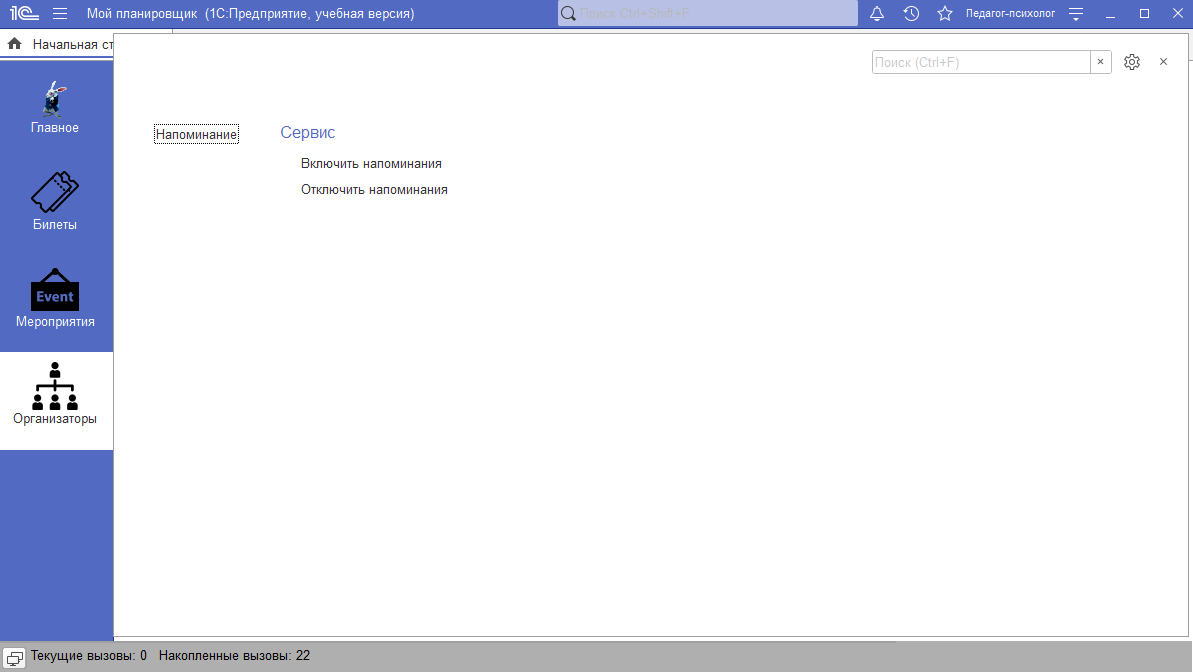


Рисунок 46 – «Окно организаторы 1C»

2.7 Разработка встроенной системы справочного руководства и «Руководства пользователя»

Для повышения удобства использования была разработана встроенная система справки:

* "Всплывающие" подсказки: при наведении курсора на элементы интерфейса появляется краткая информация о функции элемента.
* Раздел "Помощь": содержит подробное руководство пользователя с описанием всех возможностей системы.

2.8 Проведение тестирования

Был создан подробный тест-план, включающий:

* Функциональное тестирование: проверка всех функций системы на соответствие требованиям.
* Модульное тестирование: использование отладочных классов для проверки отдельных модулей.
* Интеграционное тестирование: проверка взаимодействия между модулями.

Использовались модули тестирования, встроенные в Unity, а также сторонние инструменты, такие как NUnit.

2.9 Функциональное тестирование

Для каждой функции были определены наборы входных данных и сценарии тестирования:

* Регистрация пользователя
  + Ввод корректных данных: ожидаемый результат — успешная регистрация.
  + Ввод некорректных данных (например, пустое поле email): ожидаемый результат — отображение сообщения об ошибке.
* Авторизация
  + Правильные учетные данные: успешный вход.
  + Неправильный пароль: сообщение об ошибке.
* Покупка билета
  + Доступные билеты: успешная покупка.
  + Недостаточное количество билетов: сообщение об отсутствии мест.
* Создание мероприятия
  + Ввод всех необходимых данных: мероприятие создается и отображается в списке.
  + Пропуск обязательного поля: отображение сообщения об ошибке.

Протокол тестирования оформлен в соответствии со стандартами и включает в себя подробные результаты каждого теста.

2.10 Формирование отчета по практике

Отчет по практике представлен в виде пояснительной записки, содержащей следующие разделы:

* Введение: описание цели и задач проекта.
* Анализ предметной области: обоснование необходимости разработки системы.
* Техническое задание: перечисление функциональных требований.
* Описание разработки: детальное описание архитектуры и модулей.
* Тестирование: представление результатов тестирования.
* Заключение: выводы и рекомендации по дальнейшему развитию.

2.11 Особые условия: обеспечение механизма авторизации и разграничения прав

Система предусматривает два типа пользователей с различными правами доступа:

* Клиенты
  + Могут регистрироваться, авторизоваться, просматривать текущие события, покупать билеты и просматривать купленные билеты.
* Организаторы
  + Имеют доступ к созданию, редактированию и удалению мероприятий.
  + Могут просматривать статистику посещаемости.

Разграничение прав реализовано через механизм ролей, проверяемых при входе в систему и при доступе к определенным функциям.

3 Разработка сайта, мобильного приложения и тестирование

3.1 Разработка мобильного приложения для АИС "Киноконцертный Комплекс"

Для расширения доступности и удобства использования системы была разработана мобильная версия приложения. Мобильное приложение предоставляет пользователям возможность взаимодействовать с АИС "Киноконцертный Комплекс" через смартфоны и планшеты.

Ключевые особенности мобильного приложения:

Платформы: разработано для операционных систем Android и iOS с использованием технологии Xamarin.Forms на языке C#, что обеспечивает кроссплатформенность и единый код базы.

Основной функционал:

Клиенты могут регистрироваться, авторизоваться, просматривать текущие события, покупать билеты и управлять купленными билетами.

Организаторы имеют возможность создавать мероприятия и управлять ими.

Интерфейс пользователя:

Оптимизирован для мобильных устройств с учетом особенностей сенсорного ввода.

Интуитивно понятный дизайн, соответствующий гайдлайнам платформ Android и iOS.

Взаимодействие с сервером:

Используется RESTful API для обмена данными между приложением и серверной частью.

Безопасность данных обеспечивается через использование протокола HTTPS и токенов аутентификации.

База данных:

Приложение взаимодействует с базой данных MySQL, обеспечивая синхронизацию данных в реальном времени.

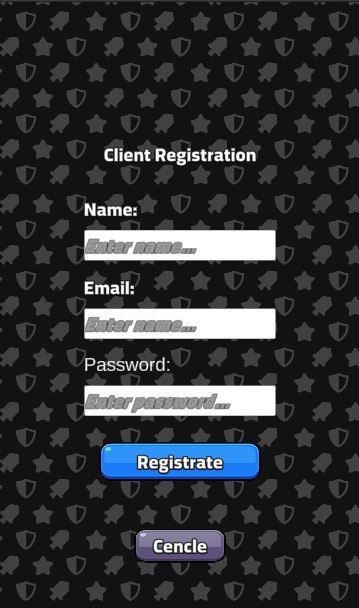


Рисунок 47 – «Окно регистрации клиента»

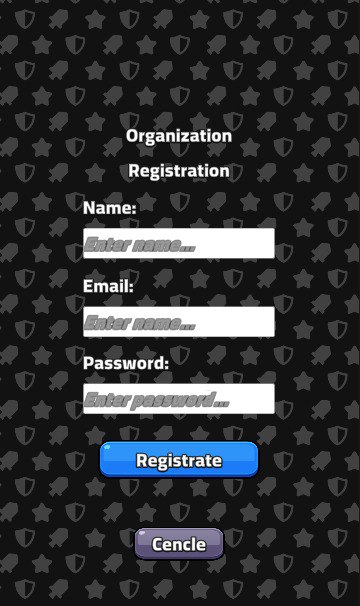


Рисунок 48 – «Окно регистрации организации»

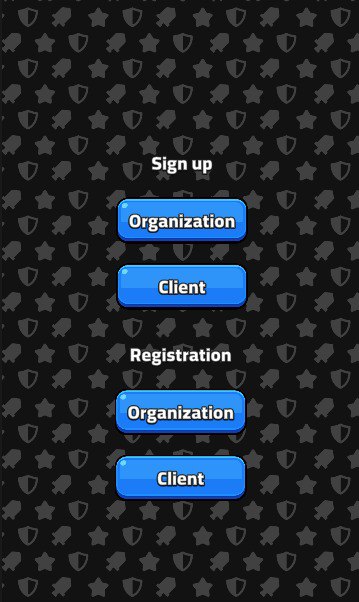


Рисунок 49 – «Окно выбора входа»



Рисунок 50 – «Окно текущих событий»



Рисунок 51 – «Окно купленных билетов»

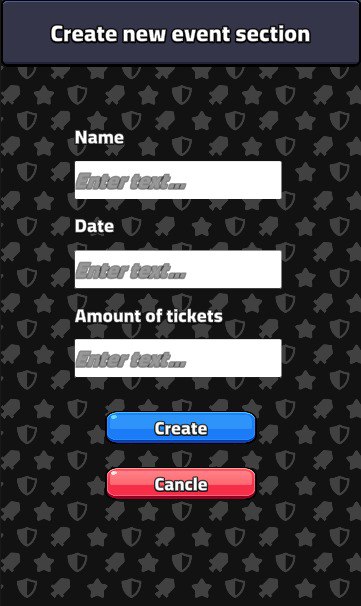


Рисунок 52 – «Окно создания события»



Рисунок 53 – «Окно покупки билетов»

3.2 Создание сайта с использованием CMS

Сайт разработан с использованием:

* HTML
* CSS
* JS

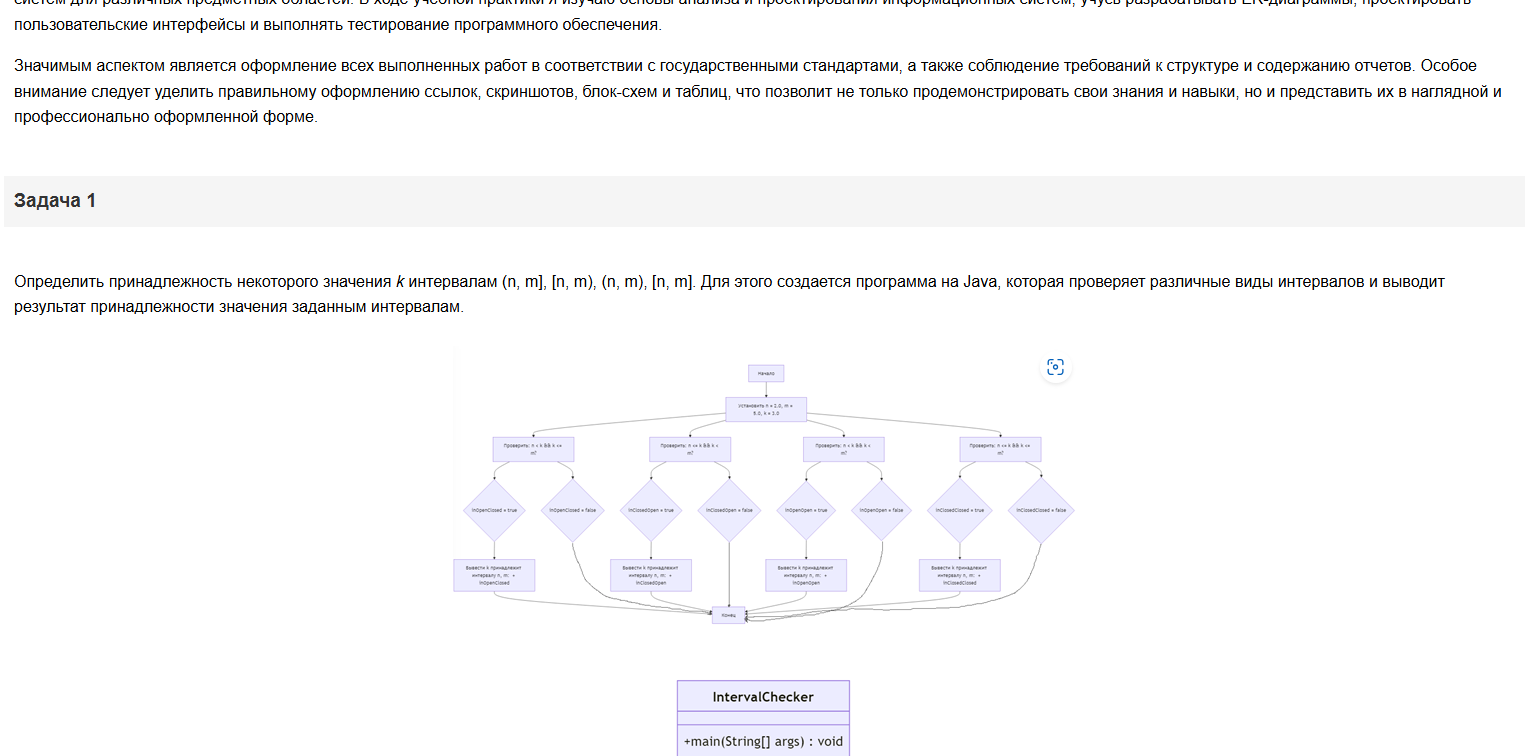


Рисунок 54 – «Окно сайта»

Ссылка на сайт: https://4amer.github.io/cooSite/

3.3 Оценка программного продукта с точки зрения эффективности использования ресурсов и применение паттерна проектирования

Программный продукт "АИС Киноконцертный Комплекс" демонстрирует эффективное использование ресурсов благодаря оптимизированному коду и рациональной архитектуре. Для улучшения структуры и поддержки расширяемости системы был применен паттерн проектирования MVC (Model-View-Controller). Выбор данного паттерна обоснован необходимостью разделения логики приложения на независимые компоненты, что упрощает разработку, тестирование и дальнейшее обслуживание системы.

Разработка "Руководства системному программисту" в соответствии со стандартом ЕСПД

Краткое содержание руководства:

1. Введение

* Назначение программы.
* Общие сведения о системе.

2. Описание программы

* Архитектура системы.
* Используемые технологии и инструменты (C#, Unity, MySQL).

3. Установка и настройка

* Требования к оборудованию и программному обеспечению.
* Инструкции по установке серверной и клиентской частей.
* Настройка базы данных MySQL.

4. Структура кода

* Описание основных модулей и их взаимодействия.
* Комментарии по структуре каталогов и файлов.

5. Инструкции по модификации

* Рекомендации по добавлению нового функционала.
* Советы по оптимизации кода.
* Особенности работы с паттерном MVC в данном проекте.

6. Отладка и тестирование

* Используемые инструменты для тестирования.
* Советы по устранению возможных ошибок.

7. Обслуживание и поддержка

* Планирование обновлений.
* Управление версиями.

Руководство обеспечивает системного программиста всей необходимой информацией для эффективного сопровождения и развития программного продукта в соответствии со стандартом ЕСПД.

3.4 Предложения по дальнейшей оптимизации

Оптимизация базы данных: внедрение индексов для ускорения запросов, нормализация таблиц.

Улучшение производительности: использование асинхронных методов для операций ввода-вывода.

Рефакторинг кода: удаление дублирующегося кода, улучшение читаемости и поддерживаемости.

Кэширование данных: внедрение механизма кэширования для часто запрашиваемой информации.

3.5 Проведение оптимизационных процедур

Оптимизированы SQL-запросы: переработаны сложные запросы для уменьшения времени их выполнения.

Внедрено кэширование: реализован кэш для статических данных, что снизило нагрузку на базу данных.

Асинхронность: критические участки кода переписаны с использованием асинхронных методов.

Тестирование производительности: проведены нагрузочные тесты, подтверждающие улучшение быстродействия системы.

Заключение

В ходе разработки автоматизированной информационной системы "Киноконцертный Комплекс" были успешно выполнены все поставленные задачи. Созданная система обеспечивает эффективное взаимодействие между клиентами и организаторами, предоставляя широкий спектр функциональных возможностей, таких как покупка билетов, создание мероприятий и управление ими. Использование языка программирования C# в среде Unity позволило реализовать модульную архитектуру приложения, обеспечивая гибкость и масштабируемость.

База данных MySQL обеспечила надежное хранение и быстрый доступ к данным пользователей, мероприятий и транзакций. Применение паттерна проектирования MVC способствовало улучшению структуры кода и упрощению процесса разработки. Проведенное тестирование подтвердило стабильность и функциональность системы, соответствие ее требованиям и стандартам качества.

Разработка мобильного приложения расширила возможности системы, предоставив пользователям удобный доступ к функционалу через мобильные устройства на платформах Android и iOS. Это повысило уровень удовлетворенности пользователей и расширило аудиторию системы.

В процессе работы были проведены оптимизационные процедуры, направленные на повышение эффективности использования ресурсов и улучшение производительности приложения. Разработанное "Руководство системному программисту" обеспечивает подробные инструкции по установке, настройке и модификации системы, что облегчит ее дальнейшее сопровождение и развитие.

Таким образом, проект "АИС Киноконцертный Комплекс" достиг своих целей, предоставляя функциональное и удобное решение для управления мероприятиями и продажей билетов в киноконцертном комплексе.

Список использованных источников

1. Албахари Дж., Албахари Б. "C# 8.0 и платформы .NET Core 3.0. Профессиональное программирование", Вильямс, 2019.
2. Troelsen A. "Pro C# 8 with .NET Core 3", Apress, 2020.
3. Hocking J. "Unity in Action: Multiplatform Game Development in C# with Unity", Manning Publications, 2020.
4. MySQL Documentation - Официальная документация по MySQL.

https://dev.mysql.com/doc/

1. Microsoft Docs - Документация по C# и .NET.

https://docs.microsoft.com/ru-ru/dotnet/csharp/

1. Gamma E., Helm R., Johnson R., Vlissides J. "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, 1994.
2. Freeman E., Freeman E. "Head First Design Patterns", O'Reilly Media, 2004.
3. Fischer A., Kang J. "Xamarin.Forms Solutions: Cross-platform Mobile Development", Apress, 2018.
4. Xamarin.Forms Documentation - Официальная документация по Xamarin.Forms. <https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/>
5. Unity User Manual - Руководство пользователя Unity. https://docs.unity3d.com/Manual/index.html