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  Царственный Данил Алексеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Проверила: | Большакова-Стрекалова Анна Викторовна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Калининград, 2024

Содержание

[Введение 3](#_Toc184694424)

[1 Решение задач 4](#_Toc184694425)

[1.1 Задача 1 4](#_Toc184694426)

[1.2 Задача 2 8](#_Toc184694427)

[1.3 Задача 3 13](#_Toc184694428)

[1.4 Задача 4 20](#_Toc184694429)

[1.5 Задача 5 25](#_Toc184694430)

[1.6 Задача 6 27](#_Toc184694431)

[1.7 Задача 7 37](#_Toc184694432)

[1.8 Задача 8 39](#_Toc184694433)

[1.9 Задача 9 43](#_Toc184694434)

[1.10 Задача 10 47](#_Toc184694435)

[2. Разработка АИС «Каталог компьютерных игр» 50](#_Toc184694436)

[2.1 ER-диаграмма, диаграмма классов, диаграмма состояния 50](#_Toc184694437)

[2.2 Техническое задание 50](#_Toc184694438)

[2.3 Руководство пользователя 56](#_Toc184694439)

[3. Программирование 1С: 64](#_Toc184694440)

[4. Разработка мобильного приложения 66](#_Toc184694441)

[Заключение 71](#_Toc184694442)

[Список использованных источников 72](#_Toc184694443)

Введение

Во время данной практики продолжается изучение и получение новых знаний о программировании. В ней содержатся такие дисциплины как:

1. Разработка программных модулей
2. Поддержка и тестирование программных модулей
3. Разработка мобильных приложений
4. Системное программирование
5. Программирование 1С

Цели учебной практики:

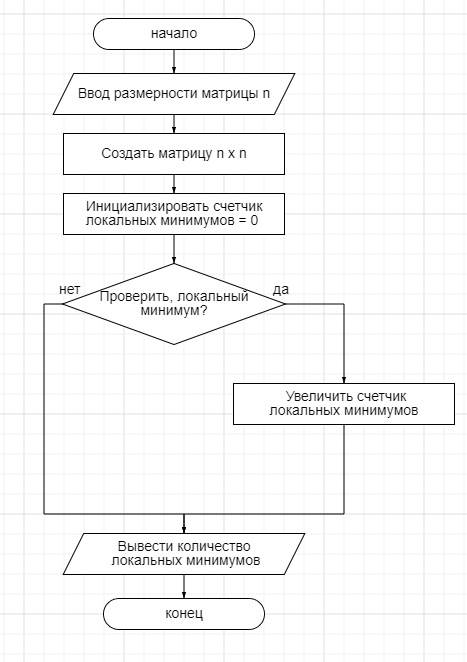
* закрепление и углубление теоретической подготовки обучающегося по программированию;
* приобретение навыков самостоятельной и научно-исследовательской работы;
* приобретение практических навыков и компетенций в сфере профессиональной деятельности;
* разработка АИС;
* разработка сайта;
* составление диаграмм;
* разработка мобильного приложения;

1. Решение задач
   1. Задача 1

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

Таблица 1. Таблица индикаторов

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Параметры |
| random | Random | Входящий создается без параметров |
| matrix | int[][] | Входящий/внутренний используется в методах, не изменяется напрямую |
| n | int | Входящий/внутренний передается в методы и используется для определения размера |
| i | int | Внутренний используется в циклах и методах, не изменяется вне методов |
| j | int | Внутренний используется в циклах и методах, не изменяется вне методов |
| current | int | Внутренний — временная переменная в методе isLocalMinimum |
| count | int | Внутренний — временная переменная в методе countLocalMinima |
| isLocalMinimum | boolean | Параметры: int[][] matrix (входящий/внутренний) — матрица для проверки; int row (входящий) — индекс строки; int col (входящий) — индекс столбца; int n (входящий) — размерность матрицы |
| countLocalMinima | int | Параметры: int[][] matrix (входящий/внутренний) — матрица для подсчета; int n (входящий) — размерность матрицы |



import java.util.Random;

import java.util.Scanner;

public class LocalMinimaMatrix {

private static void fillMatrixWithRandomValues(int[][] matrix, int 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;

}

}

}

private static boolean isLocalMinimum(int[][] matrix, int row, int col, int n) {

int current = matrix[row][col];

for (int i = row - 1; i <= row + 1; i++) {

for (int j = col - 1; j <= col + 1; j++) {

if (i >= 0 && i < n && j >= 0 && j < n && (i != row || j != col)) {

if (matrix[i][j] <= current) {

return false;

}

}

}

}

return true;

}

private static int countLocalMinima(int[][] matrix, int n) {

int count = 0;

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

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

if (isLocalMinimum(matrix, i, j, n)) {

count++;

}

}

}

return count;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int n = scanner.nextInt();

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

fillMatrixWithRandomValues(matrix, n);

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

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

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

System.out.printf("%4d", matrix[i][j]);

}

System.out.println();

}

int localMinimaCount = countLocalMinima(matrix, n);

System.out.println("Количество локальных минимумов: " + localMinimaCount);

scanner.close();

}

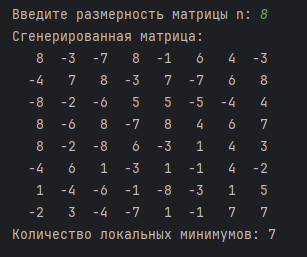


Рисунок 1 – Результат задачи 1

* 1. Задача 2

Реализовать методы сложения, вычитания, умножения и деления объектов (для тех классов, объекты которых могут поддерживать арифметические действия)

Определить класс Интервал с учетом включения/невключения концов. Создать методы по определению пересечения и объединения интервалов, причем интервалы, не имеющие общих точек, пересекаться/объединятся не могут. Объявить массив/список/множество и n интервалов и определить расстояние между самыми удаленными концами.

Таблица 2. Таблица индикаторов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | Название переменной | | |  | | --- | | Тип | | |  | | --- | | Параметр | |
| interval1 | Interval | входящий |
| interval2 | Interval | входящий |
| sum | Interval | выходящий |
| difference | Interval | выходящий |
| product | Interval | выходящий |
| quotient | Interval | выходящий |
| intersection | Interval | выходящий |
| union | Interval | выходящий |
| intervals | Interval[] | входящий |
| distance | double | выходящий |
| n | int | входящий (в методе main) |
| matrix | int[][] | внутренний (в методе main) |
| random | Random | внутренний (в методе fillMatrixWithRandomValues) |
| i | int | внутренний (в циклах for) |
| j | int | внутренний (в циклах for) |
| current | int | внутренний (в методе isLocalMinimum) |
| min | double | внутренний (в методе distanceBetween) |
| max | double | внутренний (в методе distanceBetween) |
| factor | double | входящий (в методе multiply) |

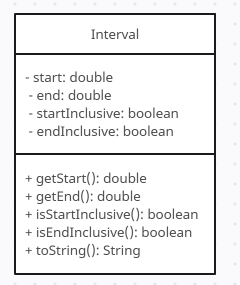


Рисунок 2 – Диаграмма класса

public class Interval {

private double start;

private double end;

private boolean startInclusive;

private boolean endInclusive;

public Interval(double start, double end, boolean startInclusive, boolean endInclusive) {

if (start > end || (start == end && (!startInclusive || !endInclusive))) {

throw new IllegalArgumentException("Некорректный интервал");

}

this.start = start;

this.end = end;

this.startInclusive = startInclusive;

this.endInclusive = endInclusive;

}

public boolean intersects(Interval other) {

return this.start < other.end && this.end > other.start

|| (this.start == other.end && (this.startInclusive || other.endInclusive))

|| (this.end == other.start && (this.endInclusive || other.startInclusive));

}

public Interval union(Interval other) {

if (!this.intersects(other)) {

throw new IllegalArgumentException("Интервалы не пересекаются");

}

double newStart = Math.min(this.start, other.start);

double newEnd = Math.max(this.end, other.end);

boolean newStartInclusive = this.start < other.start || (this.start == other.start && (this.startInclusive || other.startInclusive));

boolean newEndInclusive = this.end > other.end || (this.end == other.end && (this.endInclusive || other.endInclusive));

return new Interval(newStart, newEnd, newStartInclusive, newEndInclusive);

}

public Interval intersection(Interval other) {

if (!this.intersects(other)) {

throw new IllegalArgumentException("Интервалы не пересекаются");

}

double newStart = Math.max(this.start, other.start);

double newEnd = Math.min(this.end, other.end);

boolean newStartInclusive = this.start > other.start || (this.start == other.start && (this.startInclusive || other.startInclusive));

boolean newEndInclusive = this.end < other.end || (this.end == other.end && (this.endInclusive || other.endInclusive));

return new Interval(newStart, newEnd, newStartInclusive, newEndInclusive);

}

public Interval add(Interval other) {

return new Interval(this.start + other.start, this.end + other.end,

this.startInclusive && other.startInclusive,

this.endInclusive && other.endInclusive);

}

public Interval subtract(Interval other) {

return new Interval(this.start - other.end, this.end - other.start,

this.startInclusive && !other.endInclusive,

this.endInclusive && !other.startInclusive);

}

public Interval multiply(double factor) {

if (factor == 0) return new Interval(0, 0, true, true);

double newStart = factor < 0 ? factor \* this.end : factor \* this.start;

double newEnd = factor < 0 ? factor \* this.start : factor \* this.end;

return new Interval(newStart, newEnd, factor > 0 ? this.startInclusive : this.endInclusive,

factor > 0 ? this.endInclusive : this.startInclusive);

}

public Interval divide(double divisor) {

if (divisor == 0) throw new ArithmeticException("Нельзя делить на ноль");

double newStart = (this.start / divisor);

double newEnd = (this.end / divisor);

return new Interval(newStart, newEnd, this.startInclusive, this.endInclusive);

}

public static double distanceBetween(Interval[] intervals) {

if (intervals.length == 0) throw new IllegalArgumentException("Не указаны интервалы");

double min = intervals[0].start;

double max = intervals[0].end;

for (Interval interval : intervals) {

min = Math.min(min, interval.start);

max = Math.max(max, interval.end);

}

return max - min;

}

public double getStart() {

return start;

}

public double getEnd() {

return end;

}

public boolean isStartInclusive() {

return startInclusive;

}

public boolean isEndInclusive() {

return endInclusive;

}

@Override

public String toString() {

return (startInclusive ? "[" : "(") + start + ", " + end + (endInclusive ? "]" : ")");

}

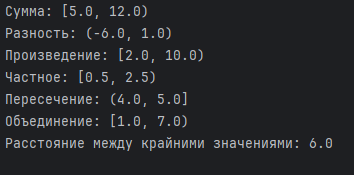


Рисунок 3 – Результат программы задачи 2

* 1. Задача 3

Создать объект класса Дерево, используя классы Лист, Ветка. Методы: зацвести, опасть листьям, покрыться инеем, пожелтеть листьям.

Таблица 3. Таблица индикаторов

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Параметры |
| args | String[] | Входящий |
| leaf1 | Leaf | Входящий |
| leaf2 | Leaf | Входящий |
| branch1 | Branch | Входящий |
| tree | Tree | Входящий |
| leaf3 | Leaf | Входящий |
| branches | List<Branch> | Внутренний |
| branch | Branch | Входящий |
| branch | Branch | Входящий |
| interval | Interval[] | Входящий |
| leaves | List<Leaf> | Внутренний |
| leaf | Leaf | Входящий |
| leaf | Leaf | Входящий |
| color | String | Внутренний |
| isBlooming | boolean | Внутренний |
| isFrosted | boolean | Внутренний |

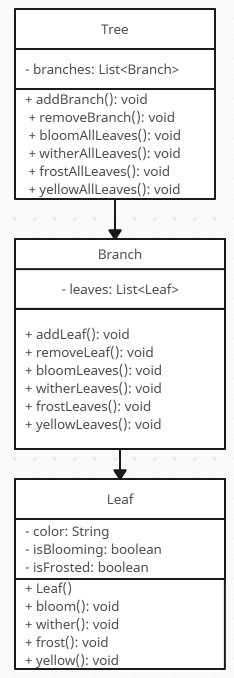


Рисунок 4 – Диаграмма классов задания 3

} import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class Tree {

private List<Branch> branches;

public Tree() {

branches = new ArrayList<>();

}

public void addBranch(Branch branch) {

branches.add(branch);

}

public void removeBranch(Branch branch) {

branches.remove(branch);

}

public void bloomAllLeaves() {

for (Branch branch : branches) {

branch.bloomLeaves();

}

}

public void witherAllLeaves() {

for (Branch branch : branches) {

branch.witherLeaves();

}

}

public void frostAllLeaves() {

for (Branch branch : branches) {

branch.frostLeaves();

}

}

public void yellowAllLeaves() {

for (Branch branch : branches) {

branch.yellowLeaves();

}

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Tree tree = (Tree) o;

return Objects.equals(branches, tree.branches);

}

@Override

public int hashCode() {

return Objects.hash(branches);

}

@Override

public String toString() {

return "Дерево{" +

"ветви=" + branches +

'}';

}

}

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class Branch {

private List<Leaf> leaves;

public Branch() {

leaves = new ArrayList<>();

}

public void addLeaf(Leaf leaf) {

leaves.add(leaf);

}

public void removeLeaf(Leaf leaf) {

leaves.remove(leaf);

}

public void bloomLeaves() {

for (Leaf leaf : leaves) {

leaf.bloom();

}

}

public void witherLeaves() {

for (Leaf leaf : leaves) {

leaf.wither();

}

}

public void frostLeaves() {

for (Leaf leaf : leaves) {

leaf.frost();

}

}

public void yellowLeaves() {

for (Leaf leaf : leaves) {

leaf.yellow();

}

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Branch branch = (Branch) o;

return Objects.equals(leaves, branch.leaves);

}

@Override

public int hashCode() {

return Objects.hash(leaves);

}

@Override

public String toString() {

return "Ветвь{" +

"листья=" + leaves +

'}';

}

}

import java.util.Objects;

public class Leaf {

private String color;

private boolean isBlooming;

private boolean isFrosted;

public Leaf(String color) {

this.color = color;

this.isBlooming = false;

this.isFrosted = false;

}

public void bloom() {

isBlooming = true;

System.out.println("Лист распускается.");

}

public void wither() {

isBlooming = false;

System.out.println("Лист вянет.");

}

public void frost() {

isFrosted = true;

System.out.println("Лист покрывается инеем.");

}

public void yellow() {

color = "желтый";

System.out.println("Лист стал желтым.");

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Leaf leaf = (Leaf) o;

return isBlooming == leaf.isBlooming &&

isFrosted == leaf.isFrosted &&

Objects.equals(color, leaf.color);

}

@Override

public int hashCode() {

return Objects.hash(color, isBlooming, isFrosted);

}

@Override

public String toString() {

return "Лист{" +

"цвет='" + color + '\'' +

", распускается=" + isBlooming +

", покрыт инеем=" + isFrosted +

'}';

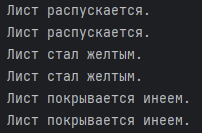


Рисунок 5 – Результат программы 3

* 1. Задача 4

Создать класс Cinema с внутренним классом, с помощью объектов которого можно хранить информацию об адресах кинотеатров, фильмах и времени начала сеансов

Таблица 4. Таблица индикаторов

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип | Параметры |
| address | String | Входящий |
| sessions | List<Session> | Входящий |
| movieTitle | String | Входящий |
| startTime | LocalTime | Входящий |

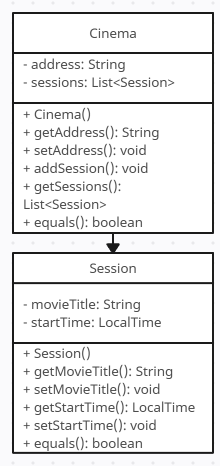


Рисунок 6 – Диаграмма классов задания 4

import java.time.LocalTime;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class Cinema {

private String address;

private List<Session> sessions;

public Cinema(String address) {

this.address = address;

this.sessions = new ArrayList<>();

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public void addSession(String movieTitle, LocalTime startTime) {

sessions.add(new Session(movieTitle, startTime));

}

public List<Session> getSessions() {

return sessions;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Cinema cinema = (Cinema) o;

return Objects.equals(address, cinema.address) &&

Objects.equals(sessions, cinema.sessions);

}

@Override

public int hashCode() {

return Objects.hash(address, sessions);

}

@Override

public String toString() {

return "Кинотеатр{" +

"адрес='" + address + '\'' +

", сеансы=" + sessions +

'}';

}

public static class Session {

private String movieTitle;

private LocalTime startTime;

public Session(String movieTitle, LocalTime startTime) {

this.movieTitle = movieTitle;

this.startTime = startTime;

}

public String getMovieTitle() {

return movieTitle;

}

public void setMovieTitle(String movieTitle) {

this.movieTitle = movieTitle;

}

public LocalTime getStartTime() {

return startTime;

}

public void setStartTime(LocalTime startTime) {

this.startTime = startTime;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Session session = (Session) o;

return Objects.equals(movieTitle, session.movieTitle) &&

Objects.equals(startTime, session.startTime);

}

@Override

public int hashCode() {

return Objects.hash(movieTitle, startTime);

}

@Override

public String toString() {

return "Сеанс{" +

"фильм='" + movieTitle + '\'' +

", время начала=" + startTime +

'}';

}

}

public static void main(String[] args) {

Cinema cinema = new Cinema("123 Главная ул, Город");

cinema.addSession("Матрица", LocalTime.of(19, 0));

cinema.addSession("Начало", LocalTime.of(21, 30));

System.out.println(cinema);

for (Session session : cinema.getSessions()) {

System.out.println("Фильм: " + session.getMovieTitle() + ", Время начала: " + session.getStartTime());

}

}



Рисунок 7 – Результат программы 4

* 1. Задача 5

В тексте найти первую подстроку максимальной длины, не содержащую букв

Таблица 5. Таблица индикаторов

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Параметры |
| text | String | Входящий |
| result | String | Выходящий |
| maxLength | int | Внутренний |
| maxStart | int | Внутренний |
| currentStart | int | Внутренний |
| currentLength | int | Внутренний |
| c | char | Внутренний |
| i | int | Внутренний |



public class Main {

public static void main(String[] args) {

String text = "abc123!@#456def789";

String result = findLongestNonAlphabeticSubstring(text);

System.out.println("Самая длинная непострочная подстрока: " + result);

}

public static String findLongestNonAlphabeticSubstring(String text) {

if (text == null || text.isEmpty()) {

return "";

}

int maxLength = 0;

int maxStart = 0;

int currentStart = 0;

int currentLength = 0;

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

char c = text.charAt(i);

if (Character.isLetter(c)) {

if (currentLength > maxLength) {

maxLength = currentLength;

maxStart = currentStart;

}

currentLength = 0;

} else {

if (currentLength == 0) {

currentStart = i;

}

currentLength++;

}

}

if (currentLength > maxLength) {

maxLength = currentLength;

maxStart = currentStart;

}

return text.substring(maxStart, maxStart + maxLength);

}



Рисунок 8 – Результат программы 5

* 1. Задача 6

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

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

Таблица 6. Таблица индикаторов

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Параметры |
| branches | List | Внутренний |
| branch | Branch | Входящий |
| leaf | Leaf | Входящий |
| leaves | List | Внутренний |
| color | String | Внутренний |
| isBlooming | boolean | Внутренний |
| isFrosted | boolean | Внутренний |
| maxLength | int | Внутренний |
| maxStart | int | Внутренний |
| currentStart | int | Внутренний |
| currentLength | int | Внутренний |
| o | Object | Входящий |
| movieTitle | String | Входящий |
| startTime | LocalTime | Входящий |

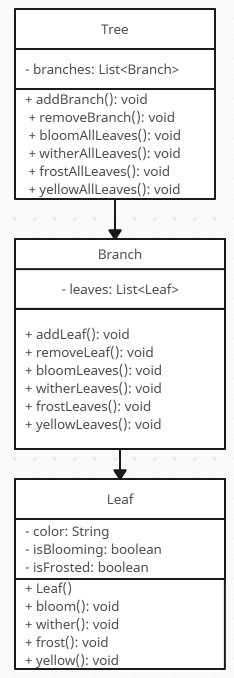


Рисунок 9 – Диаграмма классов задания 6

class InvalidLeafColorException extends Exception {

public InvalidLeafColorException(String message) {

super(message);

}

}

class NoBranchesException extends Exception {

public NoBranchesException(String message) {

super(message);

}

}

class MemoryLimitExceededException extends Exception {

public MemoryLimitExceededException(String message) {

super(message);

}

}  
import java.util.Objects;

public class Leaf {

private String color;

private boolean isBlooming;

private boolean isFrosted;

public Leaf(String color) throws InvalidLeafColorException {

if (color == null || color.isEmpty()) {

throw new InvalidLeafColorException("Цвет листа не может быть пустым.");

}

this.color = color;

this.isBlooming = false;

this.isFrosted = false;

}

public void bloom() {

isBlooming = true;

System.out.println("Лист распускается.");

}

public void wither() {

isBlooming = false;

System.out.println("Лист вянет.");

}

public void frost() {

isFrosted = true;

System.out.println("Лист покрывается инеем.");

}

public void yellow() {

color = "желтый";

System.out.println("Лист стал желтым.");

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Leaf leaf = (Leaf) o;

return isBlooming == leaf.isBlooming &&

isFrosted == leaf.isFrosted &&

Objects.equals(color, leaf.color);

}

@Override

public int hashCode() {

return Objects.hash(color, isBlooming, isFrosted);

}

@Override

public String toString() {

return "Лист{" +

"цвет='" + color + '\'' +

", распускается=" + isBlooming +

", покрыт инеем=" + isFrosted +

'}';

}

}  
import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class Branch {

private List<Leaf> leaves;

public Branch() {

leaves = new ArrayList<>();

}

public void addLeaf(Leaf leaf) {

leaves.add(leaf);

}

public void removeLeaf(Leaf leaf) throws NoBranchesException {

if (leaves.isEmpty()) {

throw new NoBranchesException("Нет листьев для удаления.");

}

leaves.remove(leaf);

}

public void bloomLeaves() throws NoBranchesException {

if (leaves.isEmpty()) {

throw new NoBranchesException("Нет листьев для распускания.");

}

for (Leaf leaf : leaves) {

leaf.bloom();

}

}

public void witherLeaves() throws NoBranchesException {

if (leaves.isEmpty()) {

throw new NoBranchesException("Нет листьев для увядания.");

}

for (Leaf leaf : leaves) {

leaf.wither();

}

}

public void frostLeaves() throws NoBranchesException {

if (leaves.isEmpty()) {

throw new NoBranchesException("Нет листьев для покрытия инеем.");

}

for (Leaf leaf : leaves) {

leaf.frost();

}

}

public void yellowLeaves() throws NoBranchesException {

if (leaves.isEmpty()) {

throw new NoBranchesException("Нет листьев для пожелтения.");

}

for (Leaf leaf : leaves) {

leaf.yellow();

}

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Branch branch = (Branch) o;

return Objects.equals(leaves, branch.leaves);

}

@Override

public int hashCode() {

return Objects.hash(leaves);

}

@Override

public String toString() {

return "Ветвь{" +

"листья=" + leaves +

'}';

}

}  
import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

public class Tree {

private List<Branch> branches;

public Tree() {

branches = new ArrayList<>();

}

public void addBranch(Branch branch) {

branches.add(branch);

}

public void removeBranch(Branch branch) throws NoBranchesException {

if (branches.isEmpty()) {

throw new NoBranchesException("Нет ветвей для удаления.");

}

branches.remove(branch);

}

public void bloomAllLeaves() throws NoBranchesException {

if (branches.isEmpty()) {

throw new NoBranchesException("Нет ветвей для распускания листьев.");

}

for (Branch branch : branches) {

branch.bloomLeaves();

}

}

public void witherAllLeaves() throws NoBranchesException {

if (branches.isEmpty()) {

throw new NoBranchesException("Нет ветвей для увядания листьев.");

}

for (Branch branch : branches) {

branch.witherLeaves();

}

}

public void frostAllLeaves() throws NoBranchesException {

if (branches.isEmpty()) {

throw new NoBranchesException("Нет ветвей для покрытия инеем.");

}

for (Branch branch : branches) {

branch.frostLeaves();

}

}

public void yellowAllLeaves() throws NoBranchesException {

if (branches.isEmpty()) {

throw new NoBranchesException("Нет ветвей для пожелтения листьев.");

}

for (Branch branch : branches) {

branch.yellowLeaves();

}

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

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

Tree tree = (Tree) o;

return Objects.equals(branches, tree.branches);

}

@Override

public int hashCode() {

return Objects.hash(branches);

}

@Override

public String toString() {

return "Дерево{" +

"ветви=" + branches +

'}';

}

}



Рисунок 10 – результат задания 6

* 1. Задача 7

Составить программу получения на экране рисунков.

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.geom.\*;  
public class ButterflyDrawing extends JPanel {  
  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 Graphics2D g2d = (Graphics2D) g;  
 g2d.setRenderingHint(RenderingHints.*KEY\_ANTIALIASING*, RenderingHints.*VALUE\_ANTIALIAS\_ON*);  
 Color[] wingColors = {Color.*RED*, Color.*ORANGE*};  
 g2d.setColor(Color.*BLACK*);  
 g2d.fillRect(175, 175, 20, 100);  
 Shape leftTopWing = createWingShape(140, 175, -1, -1);  
 g2d.setColor(wingColors[0]);  
 g2d.fill(leftTopWing);  
 Shape rightTopWing = createWingShape(220, 175, 1, -1);  
 g2d.setColor(wingColors[0]);  
 g2d.fill(rightTopWing);  
 Shape leftBottomWing = createWingShape(140, 275, -1, 1);  
 g2d.setColor(wingColors[1]);  
 g2d.fill(leftBottomWing);  
 Shape rightBottomWing = createWingShape(220, 275, 1, 1);  
 g2d.setColor(wingColors[1]);  
 g2d.fill(rightBottomWing);  
 g2d.drawLine(185, 175, 155, 125);  
 g2d.drawLine(185, 175, 215, 125);  
 }  
 private Shape createWingShape(int x, int y, int xDirection, int yDirection) {  
 GeneralPath wing = new GeneralPath();  
 wing.moveTo(x, y);  
 wing.quadTo(x + 30 \* xDirection, y - 50 \* yDirection, x + 60 \* xDirection, y);  
 wing.quadTo(x + 30 \* xDirection, y + 50 \* yDirection, x, y + 100 \* yDirection);  
 wing.closePath();  
 return wing;  
 }  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Butterfly Drawing");  
 ButterflyDrawing panel = new ButterflyDrawing();  
 frame.add(panel);  
 frame.setSize(400, 400);  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 frame.setVisible(true);  
 }  
}

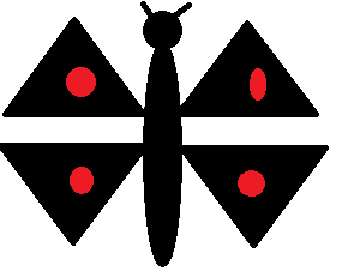


Рисунок 11 – результат задания 7

* 1. Задача 8

Составить описание объектного типа TMatr, обеспечиваю­щего размещение матрицы произвольного размера, предусмо­трев при этом возможность изменения числа строк и столбцов, а также вывода на экран подматрицы любого размера и всей ма­трицы

Таблица 7. Таблица индикаторов

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Параметры |
| matrix | int[][] | Внутренний |
| rows | int | Внутренний |
| cols | int | Внутренний |
| newMatrix | int[][] | Внутренний (в методе resize) |
| i | int | Внутренний (в методе resize, printSubMatrix) |
| j | int | Внутренний (в методе resize, printSubMatrix) |
| row | int | Входящий (в методах set, get, printSubMatrix) |
| col | int | Входящий (в методах set, get, printSubMatrix) |
| value | int | Входящий (в методах set, main) |
| startRow | int | Входящий (в методе printSubMatrix) |
| startCol | int | Входящий (в методе printSubMatrix) |
| numRows | int | Входящий (в методе printSubMatrix) |
| numCols | int | Входящий (в методе printSubMatrix) |

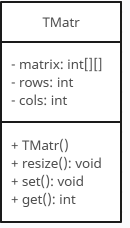


Рисунок 12 – Диаграмма классов задания 8

public class TMatr {

private int[][] matrix;

private int rows;

private int cols;

public TMatr(int rows, int cols) {

this.rows = rows;

this.cols = cols;

matrix = new int[rows][cols];

}

public void resize(int newRows, int newCols) {

int[][] newMatrix = new int[newRows][newCols];

for (int i = 0; i < Math.min(rows, newRows); i++) {

for (int j = 0; j < Math.min(cols, newCols); j++) {

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

}

}

this.rows = newRows;

this.cols = newCols;

this.matrix = newMatrix;

}

public void set(int row, int col, int value) {

if (row >= 0 && row < rows && col >= 0 && col < cols) {

matrix[row][col] = value;

} else {

throw new IndexOutOfBoundsException("Index out of bounds");

}

}

public int get(int row, int col) {

if (row >= 0 && row < rows && col >= 0 && col < cols) {

return matrix[row][col];

} else {

throw new IndexOutOfBoundsException("Index out of bounds");

}

}

public void printMatrix() {

for (int[] row : matrix) {

for (int value : row) {

System.out.print(value + " ");

}

System.out.println();

}

}

public void printSubMatrix(int startRow, int startCol, int numRows, int numCols) {

if (startRow < 0 || startCol < 0 || startRow + numRows > rows || startCol + numCols > cols) {

throw new IndexOutOfBoundsException("Submatrix exceeds matrix bounds");

}

for (int i = startRow; i < startRow + numRows; i++) {

for (int j = startCol; j < startCol + numCols; j++) {

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

}

System.out.println();

}

}

public static void main(String[] args) {

TMatr matrix = new TMatr(3, 3);

int value = 1;

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

matrix.set(i, j, value++);

}

}

System.out.println("Full Matrix:");

matrix.printMatrix();

matrix.resize(4, 4);

value = 10;

for (int i = 3; i < 4; i++) {

for (int j = 3; j < 4; j++) {

matrix.set(i, j, value++);

}

}

System.out.println("Matrix after resizing:");

matrix.printMatrix();

System.out.println("Submatrix 2x2 starting from (1,1):");

matrix.printSubMatrix(1, 1, 2, 2);

}

}

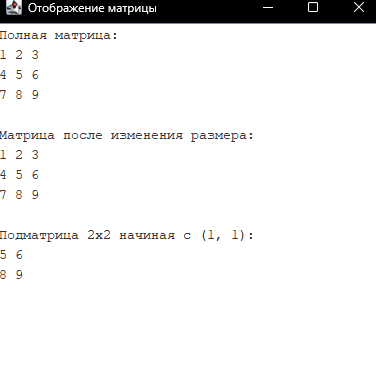


Рисунок 13 – Результат программы 8

* 1. Задача 9

Итальянская игра «Математико»

Таблица 8. Таблица индикаторов

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Параметры |
| number | int | Уникальный номер карточки |
| cards | List<Card> | Коллекция карточек |
| random | Random | Генератор случайных чисел |
| size | final int | Размер игрового поля (5x5) |
| board | int[][] | Двумерный массив для игрового поля |
| drawnCard | Card | Вытянутая карточка |
| row | int | Номер строки на игровом поле |
| col | int | Номер столбца на игровом поле |
| filledCells | int | Количество заполненных ячеек |

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
import java.util.Random;  
import java.util.Scanner;  
class Card {  
 private final int number;  
 public Card(int number) {  
 this.number = number;  
 }  
 public int getNumber() {  
 return number;  
 }  
 @Override  
 public String toString() {  
 return String.*valueOf*(number);  
 }  
}  
class CardDeck {  
 private List<Card> cards;  
 private Random random;  
 public CardDeck() {  
 cards = new ArrayList<>();  
 random = new Random();  
 for (int i = 1; i <= 13; i++) {  
 for (int j = 0; j < 4; j++) {  
 cards.add(new Card(i));  
 }  
 }  
 Collections.*shuffle*(cards); // Перемешиваем колоду  
 }  
 public Card drawCard() {  
 if (cards.isEmpty()) {  
 return null;  
 }  
 return cards.remove(cards.size() - 1);  
 }  
}  
class GameBoard {  
 private final int size = 5;  
 public final int[][] board;  
 public GameBoard() {  
 board = new int[size][size];  
 }  
 public void fillCell(int row, int col, int number) {  
 board[row][col] = number;  
 }  
 public void printBoard() {  
 for (int[] row : board) {  
 for (int cell : row) {  
 System.*out*.print(cell + " ");  
 }  
 System.*out*.println();  
 }  
 }  
 public int evaluate() {  
 int sum = 0;  
 for (int[] row : board) {  
 for (int cell : row) {  
 sum += cell;  
 }  
 }  
 return sum;  
 }  
}  
public class MathGame {  
 private CardDeck cardDeck;  
 private GameBoard gameBoard;  
 private Scanner scanner;  
 public MathGame() {  
 cardDeck = new CardDeck();  
 gameBoard = new GameBoard();  
 scanner = new Scanner(System.*in*);  
 }  
 public void playGame() {  
 int filledCells = 0;  
 while (filledCells < 25) {  
 Card drawnCard = cardDeck.drawCard();  
 if (drawnCard == null) {  
 System.*out*.println("Больше нет карт!");  
 break;  
 }  
 System.*out*.println("Вытащенная карта: " + drawnCard);  
 System.*out*.print("Введите строку (0-4) и столбец (0-4) для размещения карты: ");  
 int row = scanner.nextInt();  
 int col = scanner.nextInt();  
 if (row < 0 || row >= 5 || col < 0 || col >= 5 || gameBoard.board[row][col] != 0) {  
 System.*out*.println("Неверная позиция или ячейка уже заполнена. Попробуйте снова.");  
 continue;  
 }  
 gameBoard.fillCell(row, col, drawnCard.getNumber());  
 filledCells++;  
 gameBoard.printBoard();  
 }  
 System.*out*.println("Игра окончена! Финальная доска:");  
 gameBoard.printBoard();  
 System.*out*.println("Счет: " + gameBoard.evaluate());  
 }  
 public static void main(String[] args) {  
 MathGame game = new MathGame();  
 game.playGame();  
 }  
}

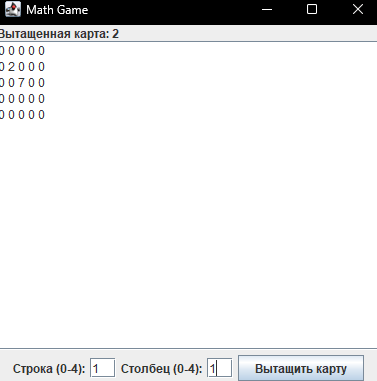
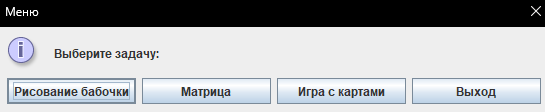


Рисунок 14 – Результат программы 8  
  
Рисунок 15 – Меню выбора зданий 7, 8, 9

* 1. Задача 10

Задача 1.

Индивидуальный вариант:

N DQ ? ; число без знака

Не используя арифметические команды, реализовать оператор

(1 вариант) N := N \* 2

Решение должно содержать не более трёх команд.

Код программы:

section .text

global \_start

\_start:

MOV RAX, N

SHL RAX, 1

MOV N, RAX

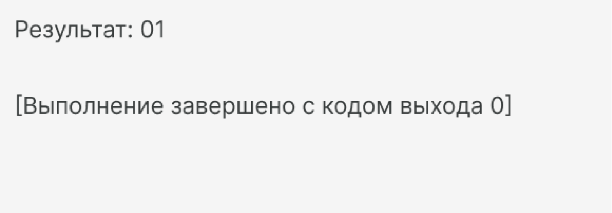


Рисунок 16 – Результат программы Задачи 10

Задача 2.

Индивидуальный вариант:

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

Код программы:

; Предполагается, что x и a передаются на стек:

; x – указатель на переменную x (4 байта)

; a – значение a (4 байта)

section .text

global Padd

Padd:

push ebp

mov ebp, esp

mov eax, [ebp + 8]

mov edx, [ebp + 12]

add dword [eax], edx

pop ebp

ret

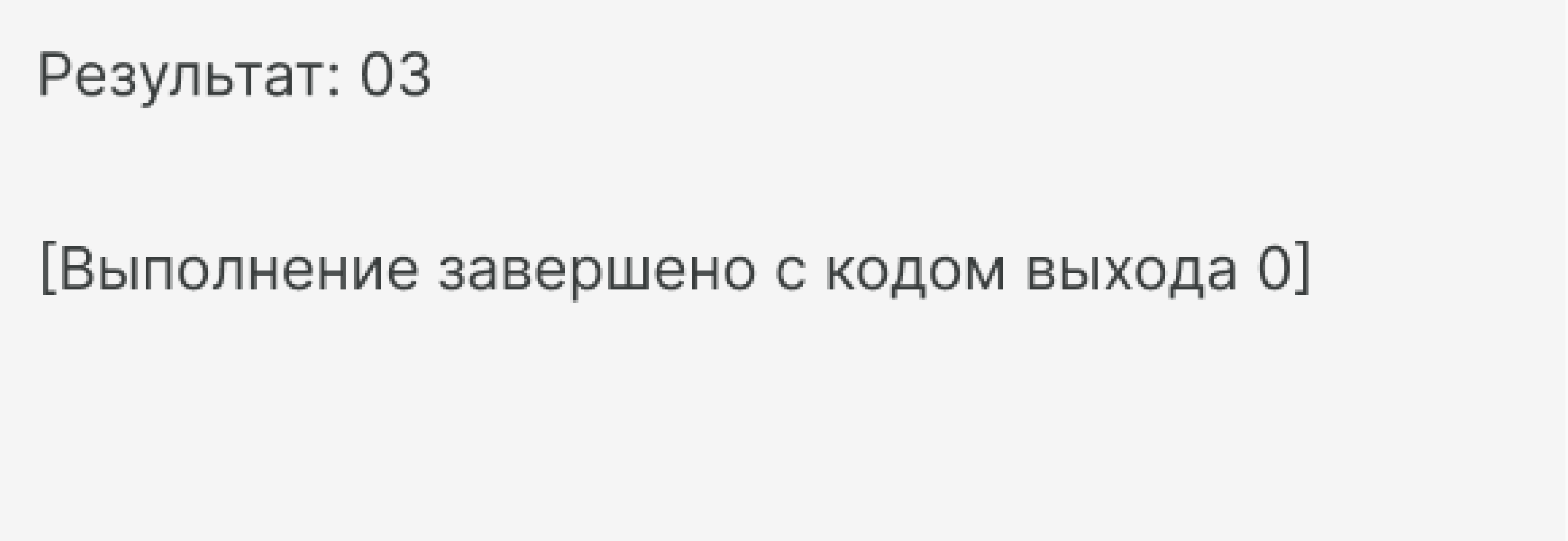


Рисунок 17 – Результат программы Задачи 10

1. Разработка АИС «Каталог компьютерных игр»
   1. ER-диаграмма, диаграмма классов, диаграмма состояния

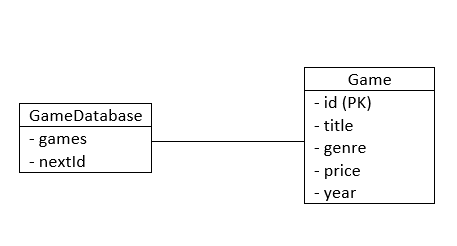


Рисунок 17 – ER диаграмма

## 2.2 Техническое задание

1. Общие сведения

**Название системы:** «Каталог компьютерных игр»  
**Разработчик:** Царственный Данил Алексееввич  
**Основание для разработки:** необходимость автоматизации процесса учета компьютерных игр, упрощения добавления и редактирования данных о них.

2. Назначение системы

2.1. Система предназначена для:

* Упрощения управления данными о компьютерных играх.
* Автоматизации операций добавления, редактирования и удаления данных.
* Удобного просмотра информации о компьютерных играх в табличной форме.

2.2. Система должна обеспечивать:

* Хранение информации о компьютерных играх:
* ID (уникальный идентификатор);
* Название игры;
* Жанр;
* Стоимость;
* Год выпуска.
* Возможность добавления новых записей.
* Возможность редактирования существующих записей.
* Возможность удаления записей.
* Табличное отображение данных игр.

3. Требования к системе

3.1. Функциональные требования:

**Модуль учета компьютерных игр:**

* Хранение информации о играх, включая:
* ID игры (уникальный идентификатор);
* Название;
* Жанр;
* Стоимость;
* Год выпуска.
* Отображение списка игр в таблице.
* Добавление новой игры с вводом всех реквизитов.
* Редактирование данных выбранной игры.
* Удаление данных выбранной игры.

3.2. Нефункциональные требования:

* Удобный графический интерфейс для пользователя.
* Простая и интуитивно понятная структура.
* Программа должна работать на платформе Windows или macOS, с использованием **Java Swing**.

4. Требования к интерфейсу пользователя

4.1. Система должна иметь:

* Таблицу для отображения списка игр с полями:
* ID;
* Название;
* Жанр;
* Стоимость;
* Год выпуска.
* Кнопки для выполнения следующих действий:
* Добавление новой игры;
* Редактирование данных выбранной игры;
* Удаление выбранной игры.

4.2. Окно добавления/редактирования должно содержать поля ввода:

* Название игры;
* Жанр;
* Стоимость;
* Год выпуска.

5. Требования к безопасности

5.1. Система должна:

* Предотвращать удаление или редактирование данных без выбора игры из таблицы.
* Выполнять изменения данных только после подтверждения пользователем.

6. Входные данные

* Название, жанр, стоимость, год выпуска игры при добавлении или редактировании.
* Выбор игры из таблицы для редактирования или удаления.

7. Выходные данные

* Таблица со списком игр, содержащая:
* ID;
* Название;
* Жанр;
* Стоимость;
* Год выпуска.
* Измененные данные игры после редактирования.
* Обновленный список игр после добавления или удаления записи.

8. Основные функции системы

* Добавление новой игры с вводом необходимых данных.
* Редактирование информации выбранной игры.
* Удаление игры из базы данных.
* Отображение списка игр в табличной форме.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Шаги | |  | | --- | |  |  |  | | --- | | Ожидаемый результат | | Статус |
| Тестирование ошибок ввода |  |  |  |  |
| 1 | Ошибка при попытке редактирования без выбора строки | Нажать кнопку "Редактировать", не выбрав игру в таблице. | Отображение сообщения: "Пожалуйста, выберите игру для редактирования. | Успешно пройден |
| 2 | Ошибка при попытке удаления без выбора строки | Нажать кнопку "Удалить", не выбрав игру в таблице. | Отображение сообщения: "Пожалуйста, выберите игру для удаления." | Успешно пройден |
| Проверка отображения данных |  |  |  |  |
| 3 | |  | | --- | |  |  |  | | --- | | Успешное отображение информации об играх | | Открыть приложение. | Список игр отображается в таблице с корректной информацией (ID, Название, Жанр, Стоимость, Год выпуска) | Успешно пройден |
| Тестирование добавления данных |  |  |  |  |
| 4 | Добавление новой игры | Ввести название, жанр, стоимость и год игры в диалоговом окне и нажать "ОК". | Игра добавлена, информация отображается в таблице. | Успешно пройден |
| 5 | Ошибка при добавлении игры с пустыми полями | Открыть диалог "Добавить игру", оставить все поля пустыми и нажать "ОК". | Игра не добавлена, отображение сообщения об ошибке или отсутствие реакции. | Успешно пройден |
| Тестирование редактирования данных |  |  |  |  |
| 6 | Редактирование данных игры | Выбрать игру, изменить данные (название, жанр, стоимость или год) и нажать "ОК". | Данные выбранной игры обновлены, изменения отображаются в таблице. | Успешно пройден |
| Тестирование удаления данных |  |  |  |  |
| 7 | Удаление игры | Выбрать игру в таблице, нажать "Удалить". | Игра удалена, информация о ней отсутствует в таблице. | Успешно пройден |
| 8 | Ошибка при попытке удаления без выбора строки | Нажать "Удалить", не выбрав игру. | Отображение сообщения: "Пожалуйста, выберите игру для удаления." | Успешно пройден |
| Общие проверки функциональности |  |  |  |  |
| 9 | Успешное обновление таблицы | Добавить, отредактировать или удалить игру. | Таблица игр автоматически обновляется после выполнения действия. | Успешно пройден |
| 10 | Ошибка при редактировании с некорректными данными | Выбрать игру, ввести в поле "Стоимость" некорректные данные (например, текст вместо числового значения). | Данные не обновлены, отображение сообщения об ошибке или возвращение к предыдущему состоянию. | Успешно пройден |

## 2.3 Руководство пользователя

1. Общее описание

Программа «Каталог компьютерных игр» разработана для управления базой данных игр. Она позволяет добавлять, редактировать и удалять данные об играх, а также просматривать список в удобной табличной форме. Программа упрощает учет игр и повышает эффективность управления их данными.

2. Системные требования

* Операционная система: Windows 10 и выше
* Память: минимум 4 ГБ ОЗУ
* Свободное место на диске: минимум 500 МБ

3. Установка программы

* Перейдите на github
* Распакуйте архив и запустите файл appdebug.apk

4. Интерфейс программы

Главное окно программы состоит из следующих элементов:

1. **Центральная панель:**  
   Отображает таблицу с данными об играх. Столбцы таблицы:

* ID (уникальный идентификатор игры);
* Название;
* Жанр;
* Стоимость;
* Год выпуска.

1. **Нижняя панель:**  
   Содержит три кнопки для управления записями:

* **Добавить:** Открывает диалоговое окно для ввода данных новой игры.
* **Редактировать:** Открывает диалоговое окно для редактирования данных выбранной игры.
* **Удалить:** Удаляет выбранную запись из таблицы.

5. Основные функции программы

**5.1 Добавление новой игры**

1. Нажмите кнопку **Добавить**.
2. В открывшемся диалоговом окне введите:

* Название игры;
* Жанр;
* Стоимость (только числовое значение);
* Год выпуска (формат: YYYY).
* Нажмите **ОК**, чтобы сохранить данные, или **Отмена**, чтобы закрыть окно без сохранения.
* Новая запись появится в таблице.

**5.2 Редактирование данных игры**

* Выберите запись, которую хотите отредактировать, в таблице.
* Нажмите кнопку **Редактировать**.
* В открывшемся окне измените необходимые данные.
* Нажмите **ОК**, чтобы сохранить изменения.
* Таблица обновится с новыми данными.

**5.3 Удаление игры**

* Выберите запись, которую хотите удалить, в таблице.
* Нажмите кнопку **Удалить**.
* Подтвердите действие в появившемся диалоговом окне.
* Запись будет удалена из таблицы.

6. Сообщения об ошибках

* Если вы попытаетесь отредактировать или удалить запись, не выбрав её в таблице, отобразится сообщение:  
  **"Пожалуйста, выберите игру для редактирования/удаления."**
* При попытке ввода некорректных данных (например, текст вместо числа) появится предупреждение о необходимости исправления.

Код программы:

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.util.ArrayList;

import java.util.List;

// Класс, представляющий игру

class Game {

int id;

String title;

String genre;

double price;

int year;

public Game(int id, String title, String genre, double price, int year) {

this.id = id;

this.title = title;

this.genre = genre;

this.price = price;

this.year = year;

}

@Override

public String toString() {

return "ID: " + id + ", Название: " + title + ", Жанр: " + genre + ", Цена: " + price + ", Год: " + year;

}

}

// Класс для управления "базой данных" игр

class GameDatabase {

private List<Game> games = new ArrayList<>();

private int nextId = 1;

public List<Game> getGames() {

return games;

}

public void addGame(String title, String genre, double price, int year) {

Game game = new Game(nextId++, title, genre, price, year);

games.add(game);

}

public void editGame(int id, String newTitle, String newGenre, double newPrice, int newYear) {

for (Game game : games) {

if (game.id == id) {

game.title = newTitle;

game.genre = newGenre;

game.price = newPrice;

game.year = newYear;

return;

}

}

}

public void deleteGame(int id) {

games.removeIf(game -> game.id == id);

}

}

// Основной класс для графического интерфейса

public class GameCatalogApp extends JFrame {

private GameDatabase database = new GameDatabase();

private DefaultTableModel tableModel;

private JTable gameTable;

public GameCatalogApp() {

setTitle("Каталог игр");

setSize(700, 400);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

// Настройка таблицы для отображения игр

tableModel = new DefaultTableModel(new Object[]{"ID", "Название", "Жанр", "Цена", "Год"}, 0);

gameTable = new JTable(tableModel);

refreshGameTable();

JScrollPane scrollPane = new JScrollPane(gameTable);

add(scrollPane, BorderLayout.CENTER);

// Панель с кнопками

JPanel buttonPanel = new JPanel();

JButton addButton = new JButton("Добавить");

JButton editButton = new JButton("Редактировать");

JButton deleteButton = new JButton("Удалить");

addButton.addActionListener(this::showAddGameDialog);

editButton.addActionListener(this::showEditGameDialog);

deleteButton.addActionListener(this::deleteSelectedGame);

buttonPanel.add(addButton);

buttonPanel.add(editButton);

buttonPanel.add(deleteButton);

add(buttonPanel, BorderLayout.SOUTH);

}

private void refreshGameTable() {

tableModel.setRowCount(0);

for (Game game : database.getGames()) {

tableModel.addRow(new Object[]{game.id, game.title, game.genre, game.price, game.year});

}

}

private void showAddGameDialog(ActionEvent e) {

JTextField titleField = new JTextField();

JTextField genreField = new JTextField();

JTextField priceField = new JTextField();

JTextField yearField = new JTextField();

Object[] fields = {

"Название:", titleField,

"Жанр:", genreField,

"Цена:", priceField,

"Год:", yearField

};

int option = JOptionPane.showConfirmDialog(this, fields, "Добавить игру", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

String title = titleField.getText();

String genre = genreField.getText();

double price;

int year;

try {

price = Double.parseDouble(priceField.getText());

year = Integer.parseInt(yearField.getText());

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Некорректная цена или год.");

return;

}

database.addGame(title, genre, price, year);

refreshGameTable();

}

}

private void showEditGameDialog(ActionEvent e) {

int selectedRow = gameTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Пожалуйста, выберите игру для редактирования.");

return;

}

int id = (int) tableModel.getValueAt(selectedRow, 0);

Game game = database.getGames().stream().filter(g -> g.id == id).findFirst().orElse(null);

JTextField titleField = new JTextField(game.title);

JTextField genreField = new JTextField(game.genre);

JTextField priceField = new JTextField(String.valueOf(game.price));

JTextField yearField = new JTextField(String.valueOf(game.year));

Object[] fields = {

"Название:", titleField,

"Жанр:", genreField,

"Цена:", priceField,

"Год:", yearField

};

int option = JOptionPane.showConfirmDialog(this, fields, "Редактировать игру", JOptionPane.OK\_CANCEL\_OPTION);

if (option == JOptionPane.OK\_OPTION) {

String newTitle = titleField.getText();

String newGenre = genreField.getText();

double newPrice;

int newYear;

try {

newPrice = Double.parseDouble(priceField.getText());

newYear = Integer.parseInt(yearField.getText());

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Некорректная цена или год.");

return;

}

database.editGame(id, newTitle, newGenre, newPrice, newYear);

refreshGameTable();

}

}

private void deleteSelectedGame(ActionEvent e) {

int selectedRow = gameTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Пожалуйста, выберите игру для удаления.");

return;

}

int id = (int) tableModel.getValueAt(selectedRow, 0);

database.deleteGame(id);

refreshGameTable();

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

GameCatalogApp app = new GameCatalogApp();

app.setVisible(true);

});

}

}

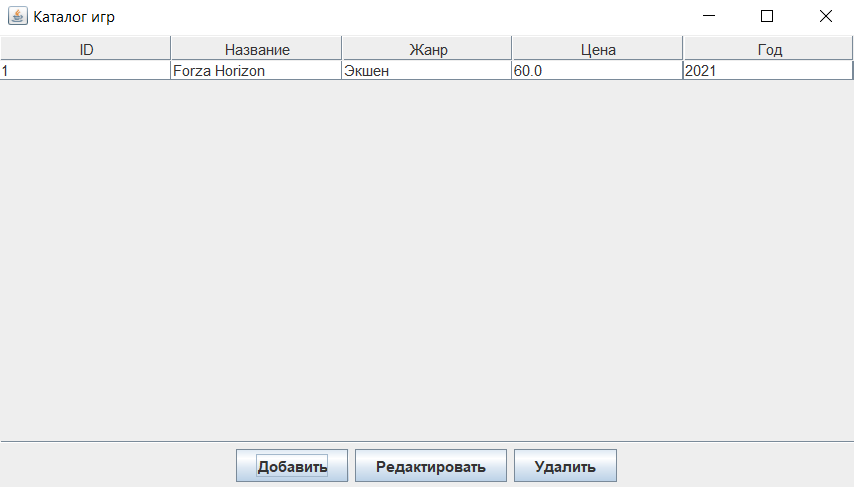


Рисунок 19 – База данных «Каталог компьютерных игр»

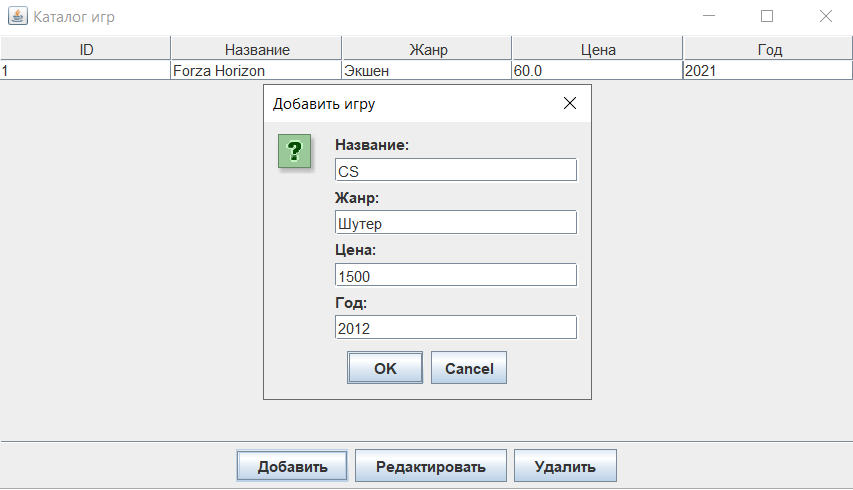


Рисунок 20 – Добавление игры в базу данных

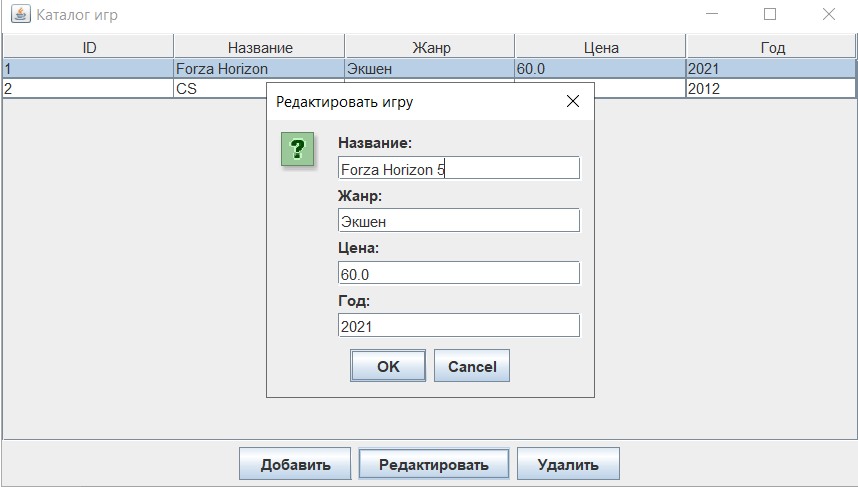


Рисунок 21 – Редактирование базы данных

3. Программирование 1С:

Разработка конфигурации на указанную тему.

Цель разработки: создать автоматизированную и удобную систему для управления Каталогом компьютерных игр.

Функциональные возможности:

Обработка данных: ввод, редактирование, удаление, хранение

Конфигурация: 1С: Предприятие

Используемые инструменты разработки: справочники, отчёты

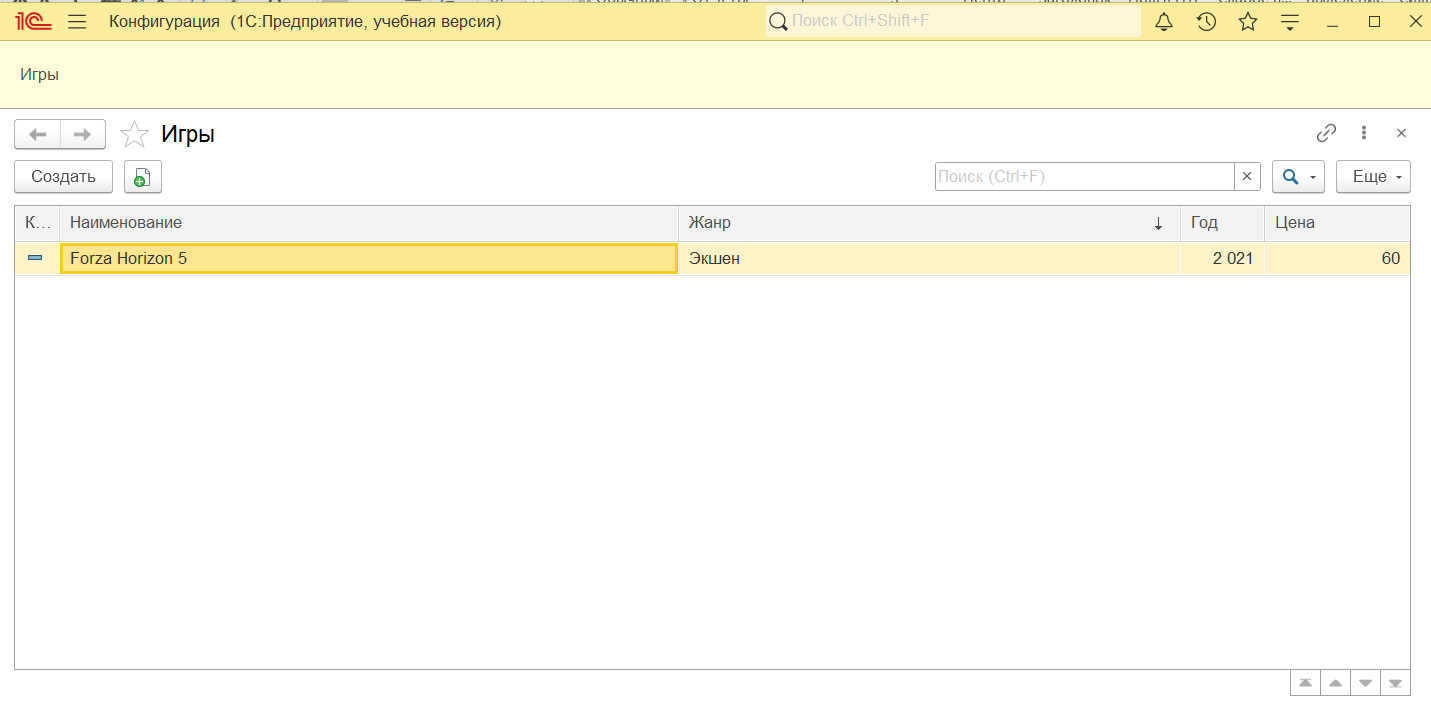


Рисунок 22 – База данных 1С «Каталог компьютерных игр»

4. Разработка мобильного приложения

Приложение «Каталог компьютерных игр» предназначено для использования его в целях просто игр или написания отзыва для игр. Интерфейс приложения достаточно прост и интуитивно понятен.

Платформа: Android

Среда разработки: Android Studio

Язык программирования: Java

Графические элементы: TextView, EditText, ImageView, Button, CardView, ListView, FragmentContainerView.

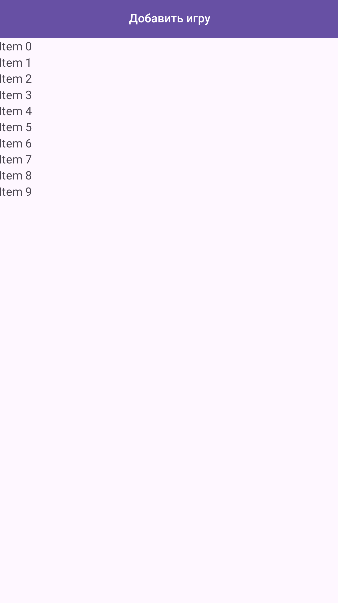


Рисунок 23 – Главный экран

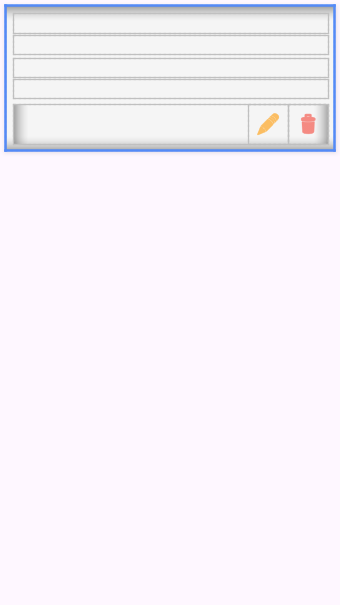


Рисунок 24 – Добавление игры в список

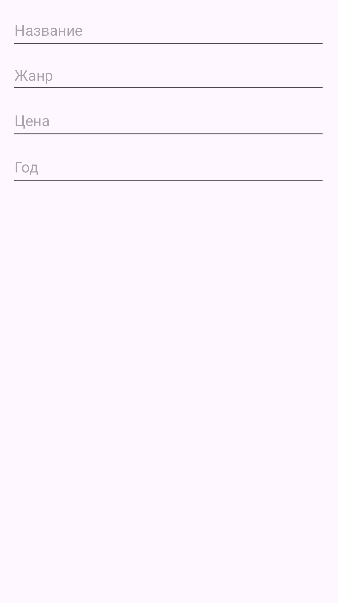


Рисунок 25 – Отображение информации о игре

1. Общие сведения

**1.1 Назначение**  
Мобильное приложение предназначено для управления каталогом компьютерных игр. Оно позволяет пользователю добавлять, редактировать и удалять записи игр, а также просматривать полный список с информацией о каждой игре (название, жанр, цена, год выпуска).

**1.2 Требования к пользователю**

* Устройство под управлением операционной системы Android версии 5.0 (Lollipop) или выше.
* Базовые навыки работы с мобильными приложениями.

2. Описание приложения

**2.1 Основные функции**

* **Просмотр каталога игр**: отображение списка игр с их характеристиками.
* **Добавление игры**: возможность вводить информацию о новой игре (название, жанр, цена, год).
* **Редактирование игры**: изменение существующих записей.
* **Удаление игры**: удаление записи из списка.

**2.2 Интерфейс**  
Приложение состоит из следующих экранов:

* **Главный экран**: отображает таблицу с играми. Здесь также находятся кнопки для управления записями (добавить, редактировать, удалить).
* **Экран добавления/редактирования**: включает поля ввода для названия, жанра, цены и года выпуска игры.
* **Диалоговые окна**: отображаются для подтверждения удаления игры или в случае ошибок (например, некорректный ввод данных).

3. Инструкция по эксплуатации

**3.1 Запуск приложения**

1. Убедитесь, что приложение установлено на вашем устройстве.
2. Нажмите на иконку приложения на главном экране устройства.
3. После запуска вы увидите главный экран с таблицей доступных игр.

**3.2 Добавление новой игры**

1. На главном экране нажмите кнопку **"Добавить"**.
2. В открывшемся окне заполните поля:
   * **Название**: введите название игры.
   * **Жанр**: укажите жанр (например, экшен, стратегия).
   * **Цена**: укажите стоимость в числовом формате.
   * **Год**: введите год выпуска.
3. Нажмите кнопку **"Сохранить"**. Если данные корректны, новая игра появится в списке.

**3.3 Редактирование игры**

1. Выберите игру из списка на главном экране.
2. Нажмите кнопку **"Редактировать"**.
3. В открывшемся окне измените необходимые поля и нажмите **"Сохранить"**.

**3.4 Удаление игры**

1. Выберите игру из списка на главном экране.
2. Нажмите кнопку **"Удалить"**.
3. Подтвердите действие в диалоговом окне.

**3.5 Работа с ошибками**

* Если при добавлении или редактировании данные введены некорректно, приложение уведомит вас об ошибке и предложит внести исправления.

# Заключение

Практика позволила создать рабочий прототип каталога компьютерных игр. В процессе работы были отработаны навыки проектирования и тестирования информационных систем, обеспечена базовая функциональность и выполнены первичные меры безопасности.

Список использованных источников

1 Блох, Д. Java. Эффективное программирование/Д. Блох. – 3-е

изд., – Диалектика , 2022 – 464 с.

2 Белугина, С. В. Разработка программных модулей программного обеспечения для компьютерных систем. Прикладное программирование / С. В. Белугина. – 4-е изд., стер. – Санкт-Петербург: Лань, 2022 – 311 с.

3 Курбатова, И. В. Основы программирования на языке Java: учебное пособие для СПО / И. В. Курбатова, А. В. Печкуров. – Санкт-Петербург: Лань, 2024 – 350 с.

4 Аблязов, Р. Программирование на ассемблере на платформе х86-64 / Р. Аблязов. – Москва: ДМК Пресс, 2020 – 302 с.

5 Васильев, А. Н. Программирование на Java / А. Н. Васильев. – Москва: Эксмо, 2024 – 704 с.

6 Ботрос, С. MySQL по максимуму / С. Ботрос, Д. Тинли. – Санкт-Петербург: Питер, 2023 – 432 с.

7 Сорокин, А. В. Программирование 1С:Предприятие 8.0 / А. В. Сорокин. – Москва: ДМК Пресс, 2020 – 341 с.

8 Колисниченко, Д. Н. Программирование для Android / Д. Н. Колисниченко. – 3-е изд., – Санкт-Петербург: БХВ, 2021 – 288 с.

9 Дарвин, Я. Ф. Сборник рецептов: задачи и решения для разработчиков приложений / Я. Ф. Дарвин. – Москва: Вильямс, 2020 – 768 с.

10 Вагин, Д. В. Современные технологии разработки веб-cайтов / Д. В. Вагин, Р. В. Петров. – Новосибирск: НГТУ, 2023 – 387с.