Пермский филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский университет «Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Шульжик Кирилл Владимирович

**Отчёт по практической работе №10 (Вариант 16)**

студента образовательной программы бакалавриата «Программная инженерия» по направлению подготовки 09.03.04 Программная инженерия

Преподаватель

В.Д Марквирер

**Оглавление**

[1 Постановка задачи 3](#_Toc97315924)

[1.1 Часть 1 3](#_Toc97315925)

[1.2 Часть 2 3](#_Toc97315926)

[1.3 Часть 3 3](#_Toc97315927)

[2 Диаграмма классов 5](#_Toc97315928)

[3 Код программы 6](#_Toc97315929)

[3.1 Код класса Program 6](#_Toc97315930)

[3.2 Код класса Print\_Edition 9](#_Toc97315931)

[3.3 Код класса Book 10](#_Toc97315932)

[3.4 Код класса Journal 11](#_Toc97315933)

[3.5 Код класса Textbook 12](#_Toc97315934)

[3.6 Код класса Film 13](#_Toc97315935)

[3.7 Код класса IdNumber 15](#_Toc97315936)

[3.8 Код интерфейса IInit 15](#_Toc97315937)

[3.9 Код класса SortByPages 15](#_Toc97315938)

[3.10 Код класса IsEqual 16](#_Toc97315939)

[4 Код unit-тестов 17](#_Toc97315940)

[5 Анализ покрытия кода тестами 27](#_Toc97315941)

1 Постановка задачи

1.1 Часть 1

Порядок выполнения работы:

1. Определить иерархию классов (16 вариант – журнал, книга, печатное издание, учебник).
2. Реализовать классы.
3. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в массив, после чего массив просматривается.
4. Реализовать 2 варианта программы: с помощью виртуальных и не виртуальных методов.
5. Без виртуальных функций программа будет работать неправильно! Объяснить почему. Объяснить необходимость виртуальных функций

1.2 Часть 2

Основное содержание работы:

1. Реализовать метод для выполнения заданных запросов. При необходимости (для выполнения запроса) в класс могут быть добавлены новые поля (по сравнению с частью 1). В программе должно быть минимум ввода с клавиатуры. Поля объектов задаются в тексте программы. С клавиатуры вводятся только параметры запроса.
2. Реализовать не менее трех запросов, соответствующих иерархии классов (можно реализовать свои запросы).

Мои запросы:

1. Посчитать суммарное число страниц всех книг.
2. Посчитать количество страниц с еженедельным периодом издания.
3. Посчитать количество учебников по физике

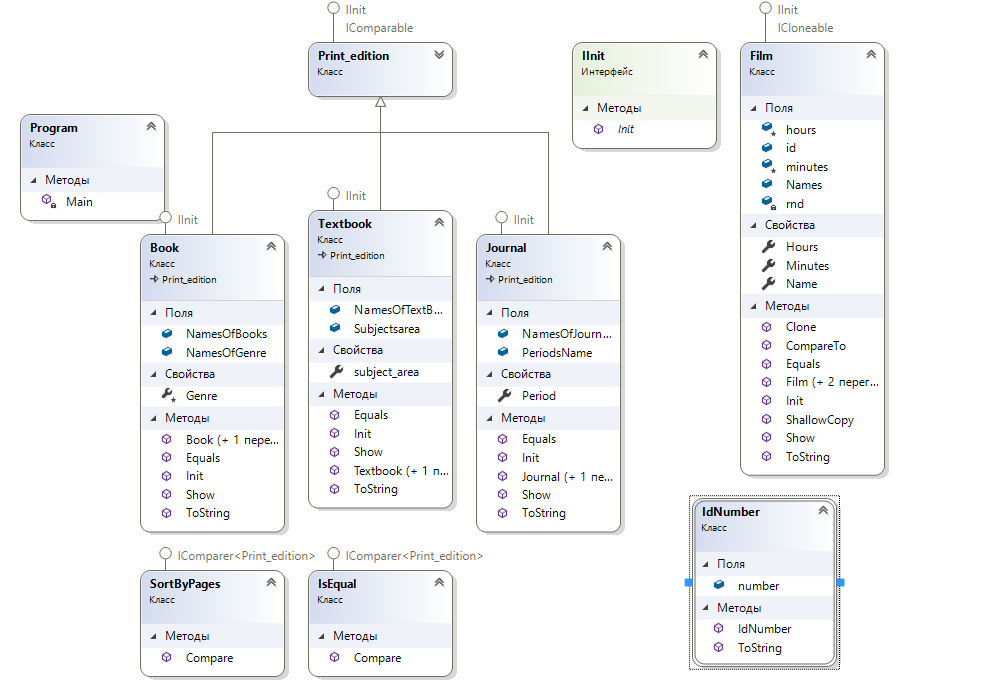
1.3 Часть 3

Задания 3 части формулируются следующим образом:

1. Создать интерфейс IInit, который содержит метод Init() для формирования элементов класса с помощью ДСЧ. Реализовать этот интерфейс в иерархии классов (часть 1 и часть 2).
2. Добавить еще один класс, не относящийся к этой иерархии классов, но реализующий интерфейс IInit (мною реализован класс Film)
3. Создать массив элементов типа IInit и поместить в него экземпляры различных классов иерархии и класса, созданного в задании 2. Выполнить просмотр массива.
4. Создать массив из элементов, относящихся к иерархии классов (часть 1 и часть 2).
5. Реализовать сортировку элементов массива, используя стандартный интерфейс IComparable и метод Sort класса Array.
6. Реализовать сортировку и поиск элемента в массиве, используя стандартный интерфейс ICompare и метод Sort класса Array.
7. Реализовать метод клонирования объектов из интерфейса IClonable. Показать разницу между клонированием и поверхностным копированием объектов.

2 Диаграмма классов

Диаграмма классов представлена на рисунке 1.



**Рисунок 1 – Диаграмма классов**

3 Код программы

3.1 Код класса Program

using System;

namespace Лаба\_10

{

class Program

{

static void Main(string[] args)

{

IsEqual a1 = new();

Print\_edition a3 = new Print\_edition("Семья недели", 5);

Print\_edition a2 = new Print\_edition("Учебный день", 3);

int a4 = a1.Compare(a3, a2);

Console.WriteLine(a4);

Console.WriteLine(string.Compare("Семья недели", "Учебный день"));

Random rand = new();

Print\_edition m = new();

m.Show();

Print\_edition m1 = new("Мастер и Маргарита");

m1.Show();

Print\_edition m2 = new("Ауе виталя", 5);

m2.Show();

Print\_edition m3 = new(5, "Шква");

m3.Show();

Journal j = new("Математические шалости");

j.Show();

Journal j1 = new();

j1.Show();

Journal j2 = new("Теленеделя", 25, "Ежедневная");

j2.Show();

Textbook t = new();

t.Show();

Textbook t1 = new("Линейная алгебра для детей", 22, "Линейная алгебра");

t1.Show();

Textbook t2 = new("Супер учебник");

t2.Show();

Book b = new();

b.Show();

Book b1 = new("Мастер и маргарита");

b1.Show();

Book b2 = new ("Олег газманов о жизни", 12, "Автобиография");

b2.Show();

Console.WriteLine();

Print\_edition[] arr = { m1, m2, j2, t1, b2 };

foreach (Print\_edition a in arr)

a.Show();

Console.WriteLine();

Print\_edition su = new();

su.Init();

su.Show();

Textbook f = new();

f.Init();

f.Show();

Book bb = new();

bb.Init();

bb.Show();

Journal as1 = new();

as1.Init();

as1.Show();

Console.WriteLine();

int sup = rand.Next(0, 4);

Print\_edition[] arr1 = new Print\_edition[500];

for (int i = 0; i < 500; i++)

{

sup = rand.Next(0, 4);

switch(sup)

{

case 0:

Print\_edition pr = new();

pr.Init();

arr1[i] = pr;

break;

case 1:

Textbook pr1 = new();

pr1.Init();

arr1[i] = pr1;

break;

case 2:

Book pr2 = new();

pr2.Init();

arr1[i] = pr2;

break;

case 3:

Journal pr3 = new();

pr3.Init();

arr1[i] = pr3;

break;

}

}

Console.WriteLine("Введите, какой запрос вы хотите выполнить\n1. Посчитать суммарное число страниц всех книг\n2. Посчитать кол-во журналов с еже" +

"недельным периодом издания\n3. Посчитать кол-во учебников по физике");

string s = Console.ReadLine();

while (s != "1" && s != "2" && s != "3")

{

Console.WriteLine("Введите либо 1, либо 2, либо 3!");

s = Console.ReadLine();

}

switch (s)

{

case "1":

int pages = 0;

int count = arr1.Length;

foreach (Print\_edition item in arr1)

pages += item.Pages\_num;

Console.WriteLine($"\nСреднее кол-во страниц всех книг массива = {pages/count}\n");

break;

case "2":

count = 0;

foreach (Print\_edition item in arr1)

{

if (item is Journal t4)

{

if (t4.Period == "Еженедельный")

count += 1;

}

}

Console.WriteLine($"\nКол-во журналов с еженедельным периодом выпуска в массиве = {count}\n");

break;

case "3":

count = 0;

foreach (Print\_edition item in arr1)

{

if (item is Textbook t3)

{

if (t3.subject\_area == "Физика")

count += 1;

}

}

Console.WriteLine($"\nКол-во учебников по физике в массиве = {count}\n");

break;

}

IInit[] arr3 = new IInit[10];

for (int i = 0; i < 8; i++)

{

sup = rand.Next(0, 4);

switch (sup)

{

case 0:

Print\_edition pr = new();

pr.Init();

arr3[i] = pr;

break;

case 1:

Textbook pr1 = new();

pr1.Init();

arr3[i] = pr1;

break;

case 2:

Book pr2 = new();

pr2.Init();

arr3[i] = pr2;

break;

case 3:

Journal pr3 = new();

pr3.Init();

arr3[i] = pr3;

break;

}

}

for (int i = 8; i < 10; i++)

{

Film a = new();

a.Init();

arr3[i] = a;

}

foreach (IInit x in arr3)

Console.WriteLine(x);

Array.Sort(arr1);

Console.WriteLine();

for (int i = 0; i < 10; i ++)

{

Console.WriteLine(arr1[i]);

}

Array.Sort(arr1, new SortByPages());

Console.WriteLine();

for (int i = 0; i < 10; i ++)

{

Console.WriteLine(arr1[i]);

}

Print\_edition n = new Print\_edition("Незивестный учебник 1", 100);

IsEqual sf = new IsEqual();

for (int i = 0; i < 500; i ++)

{

if (sf.Compare(n , arr1[i]) == 0)

{

Console.WriteLine("Элемент \n\n" + n + $"\nнайден! Место в массиве = {i}");

}

}

Film Titanik = new Film(17,"Титаник", 3,14);

Film TitanikClone = (Film)Titanik.Clone();

Film TitanikShallowCopy = (Film)Titanik.ShallowCopy();

Console.WriteLine(Titanik);

Titanik.id.number = 2;

Console.WriteLine();

Console.WriteLine(Titanik);

Console.WriteLine(TitanikClone);

Console.WriteLine(TitanikShallowCopy);

}

}

}

3.2 Код класса Print\_Edition

using System;

namespace Лаба\_10

{

public class Print\_edition: IInit, IComparable

{

protected static Random rnd = new();

private string[] NamesOfPrintEdition = {"Издание печатное 1", "Печатное издание 2"

, "Печатное издание 3", "Российская газета", "Парламентская газета", "Москвичка", "Печаточка",

"Башкортостанские вести", "Кызыл тан", "Омет", "Авраам", "Заря", "Толон", "Комсомол"};

protected int pages\_num;

public string Name { get; set; }

public int Pages\_num

{

get => pages\_num;

set

{

if (value < 0)

{

Console.WriteLine("Ошибка. Отрицательное число страниц");

pages\_num = 0;

}

else

pages\_num = value;

}

}

public Print\_edition()

{

Name = "Noname";

Pages\_num = 0;

}

public Print\_edition(string name = "Noname", int pages = 0)

{

Name = name;

Pages\_num = pages;

}

public Print\_edition(int pages = 0, string name = "Noname")

{

Name = name;

Pages\_num = pages;

}

virtual public void Show()

{

Console.WriteLine($"\nНазвание = {Name}, " +

$"кол-во страниц = {pages\_num}");

}

public virtual void Init()

{

Name = NamesOfPrintEdition[rnd.Next(NamesOfPrintEdition.Length)];

Pages\_num = rnd.Next(1, 2001);

}

public override string ToString()

{

return "Название печатного издания: " + Name + "\n" + "Страниц: " + Pages\_num + "\n";

}

public virtual int CompareTo(object obj)

{

if (obj is Print\_edition p)

{

return string.Compare(this.Name, p.Name);

}

return -1;

}

public override bool Equals(object obj)

{

Print\_edition f = (Print\_edition)obj;

return this.Name == f.Name && this.Pages\_num == f.Pages\_num;

}

}

}

3.3 Код класса Book

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_10

{

public class Book: Print\_edition, IInit

{

public string[] NamesOfBooks = {"Художественная книга", "В поле цветок", "Берёзовая роща", "Квартет роз", "Владимирские тучи",

"Афганистанские сказы", "Цветочная аллея", "Радужный мир", "Балетный занавес", "Романный кинотеатр", "Блеский миг", "Алые тучи", "Кровавый рассвет",

"Преподобный", "Алый лик", "Дива", "Ангел", "Русский бар", "Красное море"};

public string[] NamesOfGenre = {"Детектив", "Фантастика", "Приключения", "Роман", "Эпос",

"Легенды", "Сказки", "Юмор", "Исторический роман", "Любовный роман", "Приключенческий роман",

"Научная фантастика", "Фентези", "Криминальный детектив", "Светский детектив","Исторический детектив"};

protected string Genre { get; set; }

public Book():base()

{

Genre = "NoGenre";

}

public Book(string name = "Noname", int pages = 0, string genre = "NoGenre"):base(name,pages)

{

Genre = genre;

}

override public void Show()

{

base.Show();

Console.WriteLine($"Жанр = {Genre}");

}

public override void Init()

{

Name = NamesOfBooks[rnd.Next(NamesOfBooks.Length)];

Pages\_num = rnd.Next(1, 2001);

Genre = NamesOfGenre[rnd.Next(NamesOfGenre.Length)];

}

public override string ToString()

{

return "Название книги: " + Name + "\n" + "Страниц: " + Pages\_num + "\nЖанр: " + Genre + "\n";

}

public override bool Equals(object obj)

{

Book f = (Book)obj;

return this.Name == f.Name && this.Pages\_num == f.Pages\_num && this.Genre == f.Genre;

}

}

}

3.4 Код класса Journal

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_10

{

public class Journal: Print\_edition, IInit

{

public string Period { get; set; }

public string[] NamesOfJournals = {"ELLE", "Советский спорт. Футбол", "Эксперт","Караван историй",

"SALON-INTERIOR", "CQ", "Дилетант", "Саша и маша", "Наука и религия", "Forbes", "КЛУБ 4x4","5 КОЛЕСО",

"ELLE GIRL", "PSYCHOLOGIES", "РБК", "ПРОФИЛЬ", "Караван историй. Коллекция", "Понимашка", "VOGUE", "ARCHITECTURAL DIGEST",

"TATLER", "Семь дней ТВ-программа"};

public string[] PeriodsName = { "Ежедневный", "Еженедельный", "Ежемесячный", "Ежегодный" };

public Journal():base()

{

Period = "NoPeriod";

}

public Journal(string name = "Noname", int pages = 0, string period = "NoPeriod"):base(name, pages)

{

Period = period;

}

override public void Show()

{

base.Show();

Console.WriteLine($"Период журнала = {Period}");

}

public override void Init()

{

Name = NamesOfJournals[rnd.Next(NamesOfJournals.Length)];

Pages\_num = rnd.Next(1, 2001);

Period = PeriodsName[rnd.Next(PeriodsName.Length)];

}

public override string ToString()

{

return "Название журнала: " + Name + "\n" + "Страниц: " + Pages\_num + "\n" +

"Периодичность выпуска: " + Period + "\n";

}

public override bool Equals(object obj)

{

Journal f = (Journal)obj;

return this.Name == f.Name && this.Pages\_num == f.Pages\_num && this.Period == f.Period;

}

}

}

3.5 Код класса Textbook

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_10

{

public class Textbook: Print\_edition, IInit

{

public string[] NamesOfTextBooks = {"Незивестный учебник 1", "Странный учебник", "Учебник о чем-то" +

"непонятном", "Магический учебник магической науки", "Непонятный учебник непонятного содержания",

"Нужный учебник нужного содержания", "Очень интересный учебник", "учебник всем известной предметной области",

"Учебник о великом", "Учебник о прекрасном", "Учебник о совершенном", "Обязательный к изучению учебник"};

public string[] Subjectsarea = {"Русский язык", "Литература", "Английский язык", "Алгебра",

"Геометрия", "Физика", "Информатика", "История", "Обществознание", "Биология", "Химия"};

public string subject\_area { get; set; }

public Textbook():base()

{

subject\_area = "NoSubjectArea";

}

public Textbook(string name = "NoName", int pages = 0, string subjectarea = "NoSubjectArea"):base(name,pages)

{

subject\_area = subjectarea;

}

override public void Show()

{

base.Show();

Console.WriteLine($"Предметная область = {subject\_area}");

}

public override void Init()

{

Name = NamesOfTextBooks[rnd.Next(NamesOfTextBooks.Length)];

Pages\_num = rnd.Next(1, 2001);

subject\_area = Subjectsarea[rnd.Next(Subjectsarea.Length)];

}

public override string ToString()

{

return "Название учебника: " + Name + "\n" + "Страниц: " + Pages\_num + "\n" +

"Предметная область: " + subject\_area + "\n";

}

public override bool Equals(object obj)

{

Textbook f = (Textbook)obj;

return this.Name == f.Name && this.Pages\_num == f.Pages\_num && this.subject\_area == f.subject\_area;

}

}

}

3.6 Код класса Film

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_10

{

public class Film : IInit, ICloneable

{

static Random rnd = new Random();

public string[] Names = {"Не один дома", "Почти что один дома", "Не один дома 2",

"Восстание ягнят", "Комариная роща", "Берёзовый сок", "Красные стены", "Анюточкины грёзы",

"Белый фильм", "Розовый дом","Однажды в лесу", "Смертоносны ураган", "Дыхание природы"};

protected int minutes;

protected int hours;

public string Name { get; set; }

public IdNumber id;

public int Hours

{

get => hours;

set

{

if (value < 0)

{

Console.WriteLine("Ошибка. Кол-во часов не может быть отрицательным числом");

hours = 0;

}

else

hours = value;

}

}

public int Minutes

{

get => minutes;

set

{

if (value < 0)

{

Console.WriteLine("Ошибка. Кол-во минут не может быть отрицательным числом");

minutes = 0;

}

else if (value > 59)

{

this.Hours += value / 60;

minutes = value % 60;

}

else

minutes = value;

}

}

public Film()

{

Name = "Noname";

Hours = 0;

Minutes = 0;

id = new IdNumber(0);

}

public Film(int idf, string name = "Noname", int hours = 0, int minutes = 0)

{

Name = name;

Hours = hours;

Minutes = minutes;

id = new IdNumber(idf);

}

public Film(int idf, string name = "Noname", int hours = 0)

{

Name = name;

Hours = hours;

Minutes = 0;

id = new IdNumber(idf);

}

public virtual void Init()

{

Name = Names[rnd.Next(Names.Length)];

Minutes = rnd.Next(1, 60);

Hours = rnd.Next(4);

id = new IdNumber(rnd.Next(1,100));

}

public void Show()

{

Console.WriteLine($"Название фильма: {Name}\nДлительность: {Hours} часов, {Minutes} минут"

,$"ID фильма: {id}");

}

public override string ToString()

{

return "Название фильма: " + Name + "\n" + "Длительность: " + Hours + " часов, " + Minutes + " минут" + "\n"+$"ID = {id.number}\n";

}

public object Clone()

{

return new Film(this.id.number, this.Name, this.Hours, this.Minutes);

}

public object ShallowCopy()

{

return this.MemberwiseClone();

}

public override bool Equals(object obj)

{

Film f = (Film)obj;

return this.Name == f.Name && this.Hours == f.Hours && this.Minutes == f.Minutes && this.id.number == f.id.number;

}

public virtual int CompareTo(object obj)

{

if (obj is Film p)

{

return string.Compare(this.Name, p.Name);

}

return -1;

}

}

}

3.7 Код класса IdNumber

public class IdNumber

{

public int number;

public IdNumber(int a)

{

number = a;

}

public override string ToString()

{

return number.ToString();

}

}

3.8 Код интерфейса IInit

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_10

{

public interface IInit

{

void Init();

}

}

3.9 Код класса SortByPages

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_10

{

public class SortByPages : IComparer<Print\_edition>

{

public int Compare(Print\_edition p1, Print\_edition p2)

{

if (p1.Pages\_num < p2.Pages\_num) return -1;

if (p1.Pages\_num == p2.Pages\_num) return 0;

return 1;

}

}

}

3.10 Код класса IsEqual

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Лаба\_10

{

public class IsEqual : IComparer<Print\_edition>

{

public int Compare(Print\_edition p1, Print\_edition p2)

{

if (p1.Name == p2.Name && p1.Pages\_num == p2.Pages\_num)

return 0;

if (string.Compare(p1.Name, p2.Name) == 1 && p1.Pages\_num > p2.Pages\_num)

return 1;

if (string.Compare(p1.Name, p2.Name) == -1 && p1.Pages\_num < p2.Pages\_num)

return -1;

return 4;

}

}

}

4 Код unit-тестов

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Лаба\_10;

namespace TestProject2

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethodf1()

{

Print\_edition a = new Print\_edition();

Print\_edition a1 = new Print\_edition(0, "Noname");

Assert.AreEqual(a, a1);

}

[TestMethod]

public void TestMethodf2()

{

Print\_edition b1 = new Print\_edition("Noname", 0);

Print\_edition b = new Print\_edition();

Assert.AreEqual(b, b1);

}

[TestMethod]

public void TestMethodf3()

{

Print\_edition b1 = new Print\_edition("Noname", -43);

Print\_edition b = new Print\_edition("Noname", 0);

b1.Show();

Assert.AreEqual(b, b1);

}

[TestMethod]

public void TestMethodf4()

{

Print\_edition b = new Print\_edition();

b.Init();

Print\_edition b1 = new Print\_edition();

Assert.AreNotEqual(b, b1);

}

[TestMethod]

public void TestMethodf5()

{

string s = "Название печатного издания: ау\nСтраниц: 0\n";

Print\_edition f4 = new Print\_edition("ау", -4);

string s1 = f4.ToString();

Assert.AreEqual(s, s1);

}

[TestMethod]

public void TestMethodf6()

{

int a = -1;

Print\_edition a1 = new Print\_edition("adf", 4);

Print\_edition a2 = new Print\_edition("bdf", 5);

int g = a1.CompareTo(a2);

Assert.AreEqual(a, g);

}

[TestMethod]

public void TestMethodf7()

{

int a = 0;

Print\_edition a1 = new Print\_edition("adf", 4);

Print\_edition a2 = new Print\_edition("adf", 5);

int g = a1.CompareTo(a2);

Assert.AreEqual(a, g);

}

[TestMethod]

public void TestMethodf8()

{

int a = 1;

Print\_edition a1 = new Print\_edition("bdf", 4);

Print\_edition a2 = new Print\_edition("adf", 5);

int g = a1.CompareTo(a2);

Assert.AreEqual(a, g);

}

[TestMethod]

public void TestMethodf9()

{

int a = -1;

Film a1 = new Film();

Print\_edition a2 = new Print\_edition("adf", 5);

int g = a2.CompareTo(a1);

Assert.AreEqual(a, g);

}

[TestMethod]

public void TestMethodf10()

{

Textbook f = new("Незивестный учебник 1", 5, "Русский язык") ;

Textbook f1 = new(f.NamesOfTextBooks[0], 5, f.Subjectsarea[0]);

Assert.AreEqual(f, f1);

}

[TestMethod]

public void TestMethodf11()

{

Textbook f = new("Noname",0,"NoSubjectArea");

Textbook f1 = new();

Assert.AreEqual(f, f1);

f1.Show();

f1.Init();

}

[TestMethod]

public void TestMethodf12()

{

string b = "Название учебника: Noname\nСтраниц: 0\nПредметная область: NoSubjectArea\n";

Textbook f = new("Noname", 0, "NoSubjectArea");

string a = f.ToString();

Assert.AreEqual(a,b);

}

[TestMethod]

public void TestMethodf13()

{

Textbook a1 = new Textbook();

Textbook a2 = new Textbook("sdf");

Assert.AreNotEqual(a1, a2);

}

[TestMethod]

public void TestMethodf14()

{

Book a1 = new Book();

Book a2 = new Book("Noname", 0, "NoGenre");

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf15()

{

Book a1 = new Book("Художественная книга", 10, "Детектив");

Book a2 = new Book(a1.NamesOfBooks[0], 10, a1.NamesOfGenre[0]);

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf16()

{

Book a1 = new Book();

a1.Init();

Book a2 = new Book();

a2.Show();

Assert.AreNotEqual(a1, a2);

}

[TestMethod]

public void TestMethodf17()

{

string s = "Название книги: Берёзовая роща\nСтраниц: 10\nЖанр: Роман\n";

Book f1 = new Book("Берёзовая роща", 10, "Роман");

Assert.AreEqual(s, f1.ToString());

}

[TestMethod]

public void TestMethodf18()

{

Journal a1 = new Journal();

Journal a2 = new Journal("Noname", 0, "NoPeriod");

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf19()

{

Journal a1 = new Journal("ELLE", 10, "Ежедневный");

Journal a2 = new Journal(a1.NamesOfJournals[0], 10, a1.PeriodsName[0]);

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf20()

{

Journal a1 = new Journal();

a1.Init();

Journal a2 = new Journal();

a2.Show();

Assert.AreNotEqual(a1, a2);

}

[TestMethod]

public void TestMethodf21()

{

string s = "Название журнала: Караван историй. Коллекция\nСтраниц: 1827\nПериодичность выпуска: Ежедневный\n";

Journal f1 = new Journal("Караван историй. Коллекция", 1827, "Ежедневный");

Assert.AreEqual(s, f1.ToString());

}

[TestMethod]

public void TestMethodf22()

{

string s = "11";

IdNumber f = new IdNumber(11);

Assert.AreEqual(s, f.ToString());

}

[TestMethod]

public void TestMethodf23()

{

Film a1 = new Film();

Film a2 = new Film(0, "Noname", 0, 0);

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf24()

{

Film a1 = new Film(-4, "Розали", -4, -4);

Film a2 = new Film(-4, "Розали", 0, 0);

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf25()

{

Film a1 = new Film(-4, "Розали", 43, 127);

Film a2 = new Film(-4, "Розали", 45, 7);

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf26()

{

Film a1 = new Film(-4, "Не один дома", 43);

Film a2 = new Film(-4, a1.Names[0], 43);

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf27()

{

Film a1 = new Film(-4, "Не один дома", 43);

a1.Init();

Film a2 = new Film(-4, a1.Names[0], 43);

a2.Show();

Assert.AreNotEqual(a1, a2);

}

[TestMethod]

public void TestMethodf28()

{

string s = "Название фильма: Восстание ягнят\nДлительность: 0 часов, 24 минут\nID = 93\n";

Film a1 = new Film(93, "Восстание ягнят", 0, 24);

string s1 = a1.ToString();

Assert.AreEqual(s, s1);

}

[TestMethod]

public void TestMethodf29()

{

Film a1 = new();

a1.Init();

Film a2 = (Film)a1.Clone();

a1.id.number = 443;

Assert.AreNotEqual(a1, a2);

}

[TestMethod]

public void TestMethodf30()

{

Film a1 = new();

a1.Init();

Film a2 = (Film)a1.ShallowCopy();

a1.id.number = 434;

Assert.AreEqual(a1, a2);

}

[TestMethod]

public void TestMethodf31()

{

Film a1 = new(10, "Андатра", 4, 5);

Film a2 = new(4, "Бита 2", 3, 4);

Assert.AreEqual(-1, a1.CompareTo(a2));

}

[TestMethod]

public void TestMethodf32()

{

Film a1 = new(10, "Андатра", 4, 5);

Print\_edition a2 = new("Заколкино издание", 4);

Assert.AreEqual(-1, a1.CompareTo(a2));

}

[TestMethod]

public void TestMethodf33()

{

IsEqual a1 = new();

Print\_edition a = new Print\_edition("Семья", 5);

Print\_edition a2 = new Print\_edition("Семья", 5);

Assert.AreEqual(0, a1.Compare(a, a2));

}

[TestMethod]

public void TestMethodf34()

{

IsEqual a1 = new();

Print\_edition a = new Print\_edition("Семья недели", 5);

Print\_edition a2 = new Print\_edition("Семья", 5);

Assert.AreEqual(4, a1.Compare(a, a2));

}

[TestMethod]

public void TestMethodf35()

{

IsEqual a1 = new();

Print\_edition a = new Print\_edition("Учебный день", 5);

Print\_edition a2 = new Print\_edition("Семья недели", 3);

Assert.AreEqual(1, a1.Compare(a, a2));

}

[TestMethod]

public void TestMethodf36()

{

IsEqual a1 = new();

Print\_edition a = new Print\_edition("Учебный день", 5);

Print\_edition a2 = new Print\_edition("Семья недели", 3);

Assert.AreEqual(-1, a1.Compare(a2, a));

}

[TestMethod]

public void TestMethodf37()

{

SortByPages a1 = new();

Print\_edition a = new Print\_edition("Учебный день", 3);

Print\_edition a2 = new Print\_edition("Семья недели", 3);

Assert.AreEqual(0, a1.Compare(a2, a));

}

[TestMethod]

public void TestMethodf38()

{

SortByPages a1 = new();

Print\_edition a = new Print\_edition("Учебный день", 5);

Print\_edition a2 = new Print\_edition("Семья недели", 3);

Assert.AreEqual(1, a1.Compare(a, a2));

}

[TestMethod]

public void TestMethodf39()

{

SortByPages a1 = new();

Print\_edition a = new Print\_edition("Учебный день", 5);

Print\_edition a2 = new Print\_edition("Семья недели", 3);

Assert.AreEqual(-1, a1.Compare(a2, a));

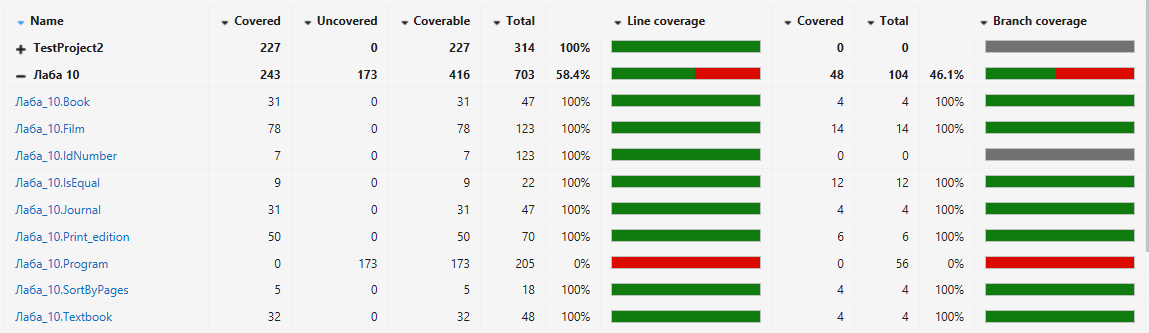
}

}

}

5 Анализ покрытия кода тестами

Покрытие кода тестами представлено на рисунке 2.



**Рисунок 2 – Анализ покрытия кода тестами**