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 №11 (Вариант 16)**

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

Преподаватель

В.Д. Марквирер

**Оглавление**

[1 Задание 11.1 3](#_Toc100002603)

[1.1 Диаграмма классов 3](#_Toc100002604)

[1.2 Иерархия классов для объектов, помещаемых в коллекцию 3](#_Toc100002605)

[1.3 Описание методов, используемых для работы с коллекцией 3](#_Toc100002606)

[1.4 Пример демонстрационной программы 3](#_Toc100002607)

[2 Задание 11.2 4](#_Toc100002608)

[2.1 Диаграмма классов 4](#_Toc100002609)

[2.2 Иерархия классов для объектов, помещаемых в коллекцию 4](#_Toc100002610)

[2.3 Описание методов, используемых для работы с коллекцией 5](#_Toc100002611)

[2.4 Пример демонстрационной программы 5](#_Toc100002612)

[3 Задание 11.3 6](#_Toc100002613)

[3.1 Диаграмма классов 6](#_Toc100002614)

[3.2 Иерархия классов для объектов, помещаемых в коллекцию 7](#_Toc100002615)

[3.3 Пример демонстрационной программы 7](#_Toc100002616)

[3.4 Полученные результаты 7](#_Toc100002617)

[3.5 Анализ полученных результатов 8](#_Toc100002618)

[ПРИЛОЖЕНИЕ А 9](#_Toc100002619)

[ПРИЛОЖЕНИЕ Б 10](#_Toc100002620)

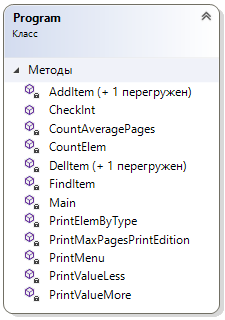
[ПРИЛОЖЕНИЕ В 28](#_Toc100002621)

[ПРИЛОЖЕНИЕ Г 35](#_Toc100002622)

[ПРИЛОЖЕНИЕ Д 38](#_Toc100002623)

1 Задание 11.1

1.1 Диаграмма классов

Диаграмма классов 11.1 задания представлена на рисунке 1.

**Рисунок 1 – Диаграмма классов 11.1 задания**

1.2 Иерархия классов для объектов, помещаемых в коллекцию

См. [ПРИЛОЖЕНИЕ А](#ПриложениеА).

1.3 Описание методов, используемых для работы с коллекцией

Для работы с коллекцией List<Print\_edition> использовались следующие методы:

1) void Add(T item) – добавление нового элемента в список.

2) bool Remove(T item) – удаление элемента из списка.

3) bool Contains(T item) – проверка наличия элемента в списке.

4) int IndexOf(T item) – возвращает индекс первого вхождения элемента в списке.

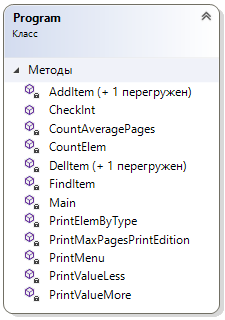
5) void Sort() – сортировка списка.

1.4 Пример демонстрационной программы

См. [ПРИЛОЖЕНИЕ Б](#ПриложениеБ).

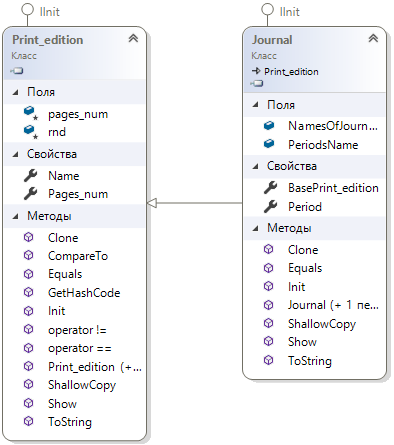
2 Задание 11.2

2.1 Диаграмма классов

Диаграмма 11.2 задания представлена на рисунке 2

**Рисунок 2 – Диаграмма классов 11.2 задания**

2.2 Иерархия классов для объектов, помещаемых в коллекцию

Иерархия классов для объектов, помещаемых в коллекцию, представлена на рисунке 3.

**Рисунок 3 – Иерархия классов объектов, помещаемых в коллекцию**

2.3 Описание методов, используемых для работы с коллекцией

Для работы с коллекцией Dictionary<Print\_edition, Journal> использовались следующие методы:

1) void Add(T key, K value) – добавление нового элемента в словарь.

2) bool ContainsKey(K key) – проверяет наличие элемента с определённым ключом в словаре.

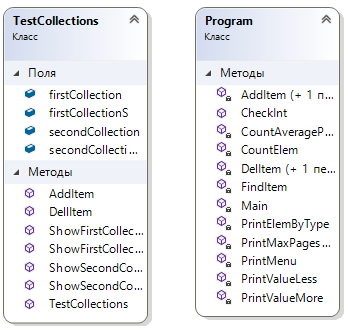
3) bool Remove(K key) – удаляет по ключу элемент из словаря.

2.4 Пример демонстрационной программы

См. [ПРИЛОЖЕНИЕ В](#ПриложениеВ).

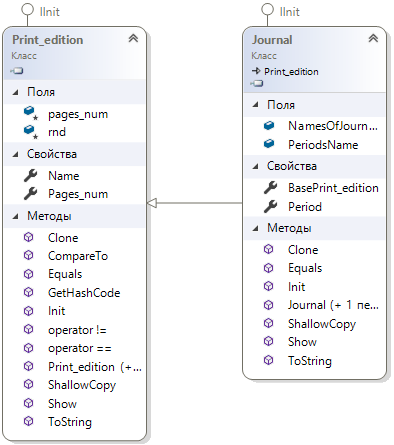
3 Задание 11.3

3.1 Диаграмма классов

Диаграмма 11.3 задания представлена на рисунке 4

**Рисунок 4 – Диаграмма классов задания 11.3**

3.2 Иерархия классов для объектов, помещаемых в коллекцию

Иерархия классов для объектов, помещаемых в коллекцию, представлена на рисунке 5.

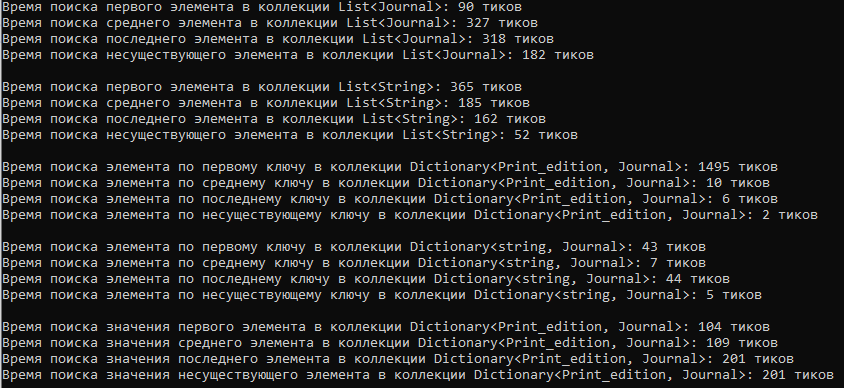
**Рисунок 5 – Иерархия классов для объектов, помещаемых в коллекцию**

3.3 Пример демонстрационной программы

См. [ПРИЛОЖЕНИЕ Г](#ПриложениеГ), [Д](#ПриложениеД).

3.4 Полученные результаты

Полученный результаты представлены на рисунке 6



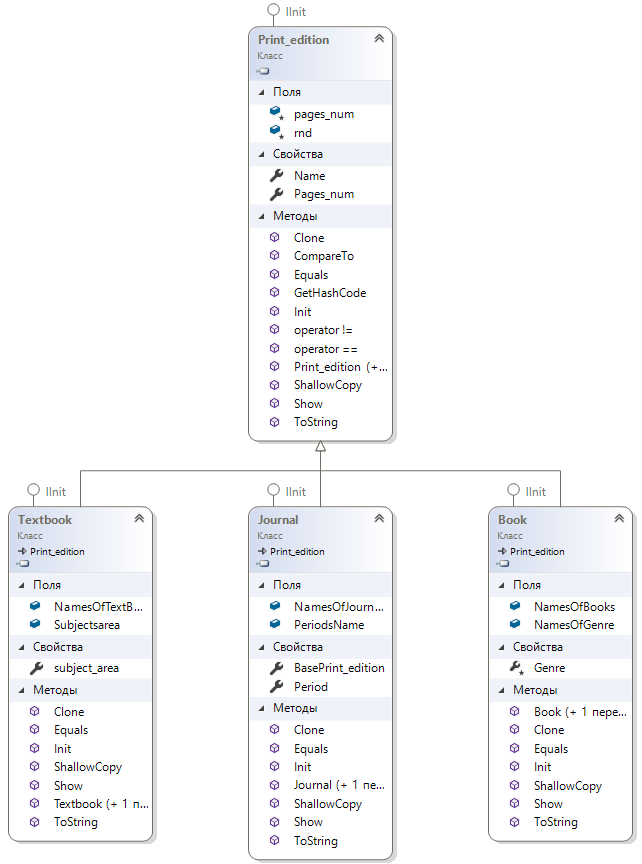
**Рисунок 6 – Полученные результаты**

3.5 Анализ полученных результатов

Как можно заметить, самой быстродействующей коллекцией оказался словарь типа Dictionary<string, Journal>. Я думаю, что, во-первых, словарь быстрее списка, потому что в словаре используется хеширование (поиск элемента по ключу в словаре гораздо быстрее поиска элемента в списке), и время поиска, считывания, записи данных остаётся постоянным. Во-вторых, словарь Dictionary<string, Journal> быстрее словаря Dictionary<Print\_edition, Journal>, потому что системе легче обрабатывать базовый тип, нежели объект созданного мною класса, который, в свою очередь, строится на основе базовых типов.

ПРИЛОЖЕНИЕ А

**Диаграмма классов**



ПРИЛОЖЕНИЕ Б

**Пример демонстрационной программы задания 11.1**

using System;

using Лаба\_10;

using System.Collections.Generic;

using System.Diagnostics;

namespace \_11\_lab

{

public class Program

{

public static void CheckInt(out int x, string msg = "", int left = Int32.MinValue, int right = Int32.MaxValue)

{

x = 0;

if (right < left && right != Int32.MaxValue && left != Int32.MinValue)

Console.WriteLine("Неверно заданы границы диапазона");

else

{

if (msg != "")

Console.WriteLine(msg);

string bug = Console.ReadLine();

while ((!int.TryParse(bug, out x) || int.Parse(bug) < left || int.Parse(bug) > right))

{

Console.WriteLine();

if (left != Int32.MinValue && right != Int32.MaxValue && left == right)

{

Console.WriteLine($"Введите число {left}");

bug = Console.ReadLine();

}

else if (left != Int32.MinValue && right != Int32.MaxValue && right - left == 1)

{

Console.WriteLine($"Введите либо {left}, либо {right}");

bug = Console.ReadLine();

}

else if (left != Int32.MinValue && right != Int32.MaxValue)

{

Console.WriteLine($"Введите целое число, большее или равное {left} и меньшее или равное {right}");

bug = Console.ReadLine();

}

else if (left != Int32.MinValue)

{

Console.WriteLine($"Введите целое число, большее или равное {left}");

bug = Console.ReadLine();

}

else if (right != Int32.MaxValue)

{

Console.WriteLine($"Введите целое число, меньшее или равное {right}");

bug = Console.ReadLine();

}

else

{

Console.WriteLine($"Введите целое число");

bug = Console.ReadLine();

}

}

x = int.Parse(bug);

}

}

static void AddItem(ref List<Print\_edition> Prints)

{

string Name; int Pages;

Console.WriteLine("Введите, какого типа вы хотите добавить объект в коллекцию");

Console.WriteLine(@"1. Book

2. Journal

3. Print edition

4. Textbook");

string inputItem = Console.ReadLine();

while (inputItem != "1" && inputItem != "2" && inputItem != "3" &&

inputItem != "4")

{

Console.WriteLine("Введите существующий тип объекта");

inputItem = Console.ReadLine();

}

Console.WriteLine("\nВведите, каким способом вы хотите добавить объект в коллекцию");

Console.WriteLine(@"1. С клавиатуры

2. Автоматически");

string howAdd = Console.ReadLine();

while (howAdd != "1" && howAdd != "2")

{

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

howAdd = Console.ReadLine();

}

switch (inputItem)

{

case "1":

if (howAdd == "2")

{

Book b = new();

b.Init();

Prints.Add(b);

}

if (howAdd == "1")

{

Console.WriteLine("\nВведите название книги, которую вы хотите добавить");

Name = Console.ReadLine();

Console.WriteLine("\nВведите кол-во страниц книги, которую вы хотите добавить");

Pages = 0;

CheckInt(out Pages, "", 0);

Console.WriteLine("\nВведите название жанра книги, которую вы хотите добавить");

string Genre = Console.ReadLine();

Book b = new Book(Name, Pages, Genre);

Prints.Add(b);

}

Console.WriteLine("\nЭлемент добавлен\n");

break;

case "2":

if (howAdd == "2")

{

Journal j = new();

j.Init();

Prints.Add(j);

}

if (howAdd == "1")

{

Console.WriteLine("\nВведите название журнала, который вы хотите удалить");

Name = Console.ReadLine();

Console.WriteLine("\nВведите кол-во страниц у журнала, который вы хотите удалить");

CheckInt(out Pages, "", 0);

Console.WriteLine("\nВведите период журнала, который вы хотите удалить");

string Period = Console.ReadLine();

Journal j = new Journal(Name, Pages, Period);

Prints.Add(j);

}

Console.WriteLine("\nЭлемент добавлен\n");

break;

case "3":

if (howAdd == "1")

{

Console.WriteLine("\nВведите название печатного издания, которое вы хотите удалить");

Name = Console.ReadLine();

Console.WriteLine("\nВведите кол-во страниц печатного издания, которое вы хотите удалить");

CheckInt(out Pages, "", 0);

Print\_edition p = new Print\_edition(Name, Pages);

Prints.Add(p);

}

if (howAdd == "2")

{

Print\_edition temp = new();

temp.Init();

Prints.Add(temp);

}

Console.WriteLine("\nЭлемент добавлен\n");

break;

case "4":

if (howAdd == "2")

{

Textbook t = new();

t.Init();

Prints.Add(t);

}

if (howAdd == "1")

{

Console.WriteLine("\nВведите название учебника, который вы хотите добавить");

Name = Console.ReadLine();

Console.WriteLine("\nВведите кол-во страниц у учебника, который вы хотите добавить");

CheckInt(out Pages, "", 0);

Console.WriteLine("\nВведите предметную область учебника, который вы хотите добавить");

string SubjectArea = Console.ReadLine();

Textbook t = new Textbook(Name, Pages, SubjectArea);

Prints.Add(t);

}

Console.WriteLine("\nЭлемент добавлен\n");

break;

}

}

static void DelItem(ref List<Print\_edition> Prints)

{

Console.WriteLine("Введите, какого типа вы хотите удалить объект в коллекции");

Console.WriteLine(@"1. Book

2. Journal

3. Print edition

4. Textbook");

Console.WriteLine();

string inputItem = Console.ReadLine();

while (inputItem != "1" && inputItem != "2" && inputItem != "3" &&

inputItem != "4")

{

Console.WriteLine("\nВведите существующий тип объекта");

inputItem = Console.ReadLine();

}

switch (inputItem)

{

case "1":

Console.WriteLine("\nВведите название книги, которую вы хотите удалить");

string Name = Console.ReadLine();

Console.WriteLine("\nВведите кол-во страниц книги, которую вы хотите удалить");

int Pages = 0;

CheckInt(out Pages, "", 0);

Console.WriteLine("\nВведите название жанра книги, которую вы хотите удалить");

string Genre = Console.ReadLine();

Book b = new Book(Name, Pages, Genre);

if (Prints.Contains(b))

{

Prints.Remove(b);

Console.WriteLine("\nЭлемент удалён\n");

}

else

Console.WriteLine("\nЭлемент не найден\n");

break;

case "2":

Console.WriteLine("\nВведите название журнала, который вы хотите удалить");

Name = Console.ReadLine();

Console.WriteLine("\nВведите кол-во страниц у журнала, который вы хотите удалить");

CheckInt(out Pages, "", 0);

Console.WriteLine("\nВведите период журнала, который вы хотите удалить");

string Period = Console.ReadLine();

Journal j = new Journal(Name, Pages, Period);

if (Prints.Contains(j))

{

Prints.Remove(j);

Console.WriteLine("\nЭлемент удалён");

}

else

Console.WriteLine("\nЭлемент не найден");

break;

case "3":

Console.WriteLine("\nВведите название печатного издания, которое вы хотите удалить");

Name = Console.ReadLine();

Console.WriteLine("\nВведите кол-во страниц печатного издания, которое вы хотите удалить");

CheckInt(out Pages, "", 0);

Print\_edition p = new Print\_edition(Name, Pages);

if (Prints.Contains(p))

{

Prints.Remove(p);

Console.WriteLine("\nЭлемент удалён");

}

else

Console.WriteLine("\nЭлемент не найден");

break;

case "4":

Console.WriteLine("\nВведите название учебника, который вы хотите удалить");

Name = Console.ReadLine();

Console.WriteLine("\nВведите кол-во страниц у элемента, который вы хотите удалить");

CheckInt(out Pages, "", 0);

Console.WriteLine("\nВведите предметную область элемента, который вы хотите удалить");

string SubjectArea = Console.ReadLine();

Textbook t = new Textbook(Name, Pages, SubjectArea);

if (Prints.Contains(t))

{

Prints.Remove(t);

Console.WriteLine("\nЭлемент удалён");

}

else

Console.WriteLine("\nЭлемент не найден");

break;

}

}

static void CountElem(ref List<Print\_edition> Prints)

{

int bookCount = 0; int journalCount = 0; int textbookCount = 0; int printEditionCount = 0;

foreach (Print\_edition x in Prints)

{

if (x is Journal)

journalCount++;

if (x is Textbook)

textbookCount++;

if (x is Journal == false && x is Textbook == false && x is Book == false)

printEditionCount++;

if (x is Book)

bookCount++;

}

Console.WriteLine($"\nКол-во объектов типа Journal в коллекции - {journalCount}\n" +

$"Кол-во объектов типа Textbook в коллекции - {textbookCount}\n" +

$"Кол-во объектов типа Print\_edition в коллекции - {printEditionCount}\n" +

$"Кол-во объектов типа Book в коллекции - {bookCount}\n");

}

static void PrintElemByType(ref List<Print\_edition> Prints)

{

Console.WriteLine();

Console.WriteLine(@"Введите, элементы какого типа вы хотите вывести на печать в коллекции

1. Journal

2. Book

3. Print\_edition

4. Textbook");

Console.WriteLine();

bool isFind = false;

string answer = Console.ReadLine();

Console.WriteLine();

while (answer != "1" && answer != "2" && answer != "3" && answer != "4")

{

Console.WriteLine("Введите число от 1 до 4");

answer = Console.ReadLine();

Console.WriteLine();

}

if (answer == "1")

{

foreach (Print\_edition x in Prints)

{

if (x is Journal)

{

isFind = true;

Console.WriteLine(x);

}

}

if (!isFind)

Console.WriteLine("В коллекции нет элементов типа Journal");

}

if (answer == "2")

{

foreach (Print\_edition x in Prints)

{

if (x is Book)

{

isFind = true;

Console.WriteLine(x);

}

}

if (!isFind)

Console.WriteLine("В коллекции нет элементов типа Book");

}

if (answer == "3")

{

foreach (Print\_edition x in Prints)

{

if (x is Journal == false && x is Textbook == false && x is Book == false)

{

Console.WriteLine(x);

isFind = true;

}

}

if (!isFind)

Console.WriteLine("В коллекции нет элементов типа Print\_edition");

}

if (answer == "4")

{

foreach (Print\_edition x in Prints)

{

if (x is Textbook)

{

Console.WriteLine(x);

isFind = true;

}

}

if (!isFind)

Console.WriteLine("В коллекции нет элементов типа Textbook");

}

}

static void CountAveragePages(ref List<Print\_edition> Prints)

{

int count = 0; int length = 0;

foreach (Print\_edition x in Prints)

{

length++;

count += x.Pages\_num;

}

Console.WriteLine($"\nСреднее число страниц у объектах коллекции {count / length}\n");

}

static string PrintMenu()

{

Console.WriteLine(@"Выберите пункт меню

1. Добавить объект в коллекцию

2. Удалить объект из коллекции

3. Распечатать элементы коллекции

0. Выход");

string answer = Console.ReadLine();

Console.WriteLine();

while (answer != "0" && answer != "1" && answer != "2" && answer != "3")

{

Console.WriteLine("Введите целое число от 0 до 3");

answer = Console.ReadLine();

}

return answer;

}

static void FindItem(ref List<Print\_edition> Prints)

{

Console.WriteLine("\nВведите, какого типа вы хотите найти объект в коллекции");

Console.WriteLine(@"1. Book

2. Journal

3. Print edition

4. Textbook");

string inputItem = Console.ReadLine();

while (inputItem != "1" && inputItem != "2" && inputItem != "3" &&

inputItem != "4")

{

Console.WriteLine("Введите существующий тип объекта");

inputItem = Console.ReadLine();

}

switch (inputItem)

{

case "1":

Console.WriteLine("\nВведите название книги");

string Name = Console.ReadLine();

CheckInt(out int Pages, "\nВведите кол-во страниц книги", 0);

Console.WriteLine("\nВведите жанр книги");

string Genre = Console.ReadLine();

Book bFind = new Book(Name, Pages, Genre);

if (Prints.Contains(bFind))

Console.WriteLine($"\nЭлемент найден, место в коллекции {Prints.IndexOf(bFind) + 1}");

else

Console.WriteLine("\nЭлемент не найден");

break;

case "2":

Console.WriteLine("\nВведите название журнала");

Name = Console.ReadLine();

CheckInt(out Pages, "\nВведите кол-во страниц журнала", 0);

Console.WriteLine("\nВведите период журнала");

string Period = Console.ReadLine();

Journal j = new Journal(Name, Pages, Period);

if (Prints.Contains(j))

{

Console.WriteLine($"\nЭлемент найден, место в коллекции {Prints.IndexOf(j)+1}");

}

else

{

Console.WriteLine("\nЭлемент не найден");

}

break;

case "3":

Console.WriteLine("\nВведите название печатного издания");

Name = Console.ReadLine();

CheckInt(out Pages, "\nВведите кол-во страниц печатного издания");

Print\_edition p = new Print\_edition(Name, Pages);

if (Prints.Contains(p))

{

Console.WriteLine($"\nЭлемент найден, место в коллекции {Prints.IndexOf(p) + 1}");

}

else

{

Console.WriteLine("\nЭлемент не найден");

}

break;

case "4":

Console.WriteLine("\nВведите название учебника");

Name = Console.ReadLine();

CheckInt(out Pages, "\nВведите кол-во страниц учебника");

Console.WriteLine("\nВведите предметную область учебника");

string SubjectArea = Console.ReadLine();

Textbook t = new Textbook(Name, Pages, SubjectArea);

if (Prints.Contains(t))

{

Console.WriteLine($"\nЭлемент найден, место в коллекции {Prints.IndexOf(t) + 1}");

}

else

{

Console.WriteLine("\nЭлемент не найден");

}

break;

}

}

static void Main(string[] args)

{

Random rand = new Random();

List<Print\_edition> Prints = new();

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

{

int classNumber = rand.Next(1, 5);

if (classNumber == 1)

{

Book b = new();

b.Init();

Prints.Add(b);

}

if (classNumber == 2)

{

Journal j1 = new();

j1.Init();

Prints.Add(j1);

}

if (classNumber == 3)

{

Print\_edition p = new();

p.Init();

Prints.Add(p);

}

if (classNumber == 4)

{

Textbook t = new();

t.Init();

Prints.Add(t);

}

} // Добавление элементов в коллекцию

string answer = "sdf";

while (answer != "0")

{

answer = PrintMenu();

switch (answer)

{

case "0":

break;

case "1":

AddItem(ref Prints);

break;

case "2":

DelItem(ref Prints);

break;

case "3":

foreach (Print\_edition x in Prints)

Console.WriteLine(x);

break;

}

}

Console.WriteLine(@"Введите, какой запрос вы хотите выбрать

1. Распечатать кол-во элементов каждого типа в коллекции

2. Вывести все элементы определённого типа в коллекции

3. Посчитать среднее число страниц в объектах всех типов коллекции");

answer = Console.ReadLine();

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

{

Console.WriteLine("Введите число от 1 до 3");

answer = Console.ReadLine();

}

if (answer == "1")

CountElem(ref Prints);

if (answer == "2")

PrintElemByType(ref Prints);

if (answer == "3")

CountAveragePages(ref Prints);

Console.WriteLine("Нажмите любую клавишу, чтобы продолжить\n");

Console.ReadKey();

List<Print\_edition> Prints\_clone = new();

List<Print\_edition> Prints\_Shallow\_Copy = new();

foreach (Print\_edition x in Prints)

{

if (x is Book == false && x is Textbook == false && x is Journal == false)

{

Prints\_clone.Add((Print\_edition)x.Clone());

Prints\_Shallow\_Copy.Add((Print\_edition)x.ShallowCopy());

}

if (x is Book b)

{

Prints\_clone.Add((Book)b.Clone());

Prints\_Shallow\_Copy.Add((Book)b.ShallowCopy());

}

if (x is Textbook t)

{

Prints\_clone.Add((Textbook)t.Clone());

Prints\_Shallow\_Copy.Add((Textbook)t.ShallowCopy());

}

if (x is Journal j1)

{

Prints\_clone.Add((Journal)j1.Clone());

Prints\_Shallow\_Copy.Add((Journal)j1.ShallowCopy());

}

}

Console.WriteLine("Исходная коллекция\n");

foreach (Print\_edition x in Prints)

Console.WriteLine(x);

Prints[0].Pages\_num = 9999;

Console.WriteLine("Изменённая коллекция\n");

foreach (Print\_edition x in Prints)

Console.WriteLine(x);

Console.WriteLine("Клон\n");

foreach (Print\_edition x in Prints\_clone)

Console.WriteLine(x);

Console.WriteLine("Поверхностное копирование\n");

foreach (Print\_edition x in Prints\_Shallow\_Copy)

Console.WriteLine(x);

Console.WriteLine("Отсортированная по страницам коллекция\n");

Prints.Sort(new SortByPages());

foreach (Print\_edition x in Prints)

Console.WriteLine(x);

FindItem(ref Prints);

Console.WriteLine("Чтобы перейти ко второй части задания нажмите любую клавишу");

Console.ReadKey();

Console.Clear();

}

}

}

ПРИЛОЖЕНИЕ В

**Пример демонстрационной программы задания 11.2**

using System;

using Лаба\_10;

using System.Collections.Generic;

using System.Diagnostics;

namespace \_11\_lab

{

public class Program

{

public static void CheckInt(out int x, string msg = "", int left = Int32.MinValue, int right = Int32.MaxValue)

{

x = 0;

if (right < left && right != Int32.MaxValue && left != Int32.MinValue)

Console.WriteLine("Неверно заданы границы диапазона");

else

{

if (msg != "")

Console.WriteLine(msg);

string bug = Console.ReadLine();

while ((!int.TryParse(bug, out x) || int.Parse(bug) < left || int.Parse(bug) > right))

{

Console.WriteLine();

if (left != Int32.MinValue && right != Int32.MaxValue && left == right)

{

Console.WriteLine($"Введите число {left}");

bug = Console.ReadLine();

}

else if (left != Int32.MinValue && right != Int32.MaxValue && right - left == 1)

{

Console.WriteLine($"Введите либо {left}, либо {right}");

bug = Console.ReadLine();

}

else if (left != Int32.MinValue && right != Int32.MaxValue)

{

Console.WriteLine($"Введите целое число, большее или равное {left} и меньшее или равное {right}");

bug = Console.ReadLine();

}

else if (left != Int32.MinValue)

{

Console.WriteLine($"Введите целое число, большее или равное {left}");

bug = Console.ReadLine();

}

else if (right != Int32.MaxValue)

{

Console.WriteLine($"Введите целое число, меньшее или равное {right}");

bug = Console.ReadLine();

}

else

{

Console.WriteLine($"Введите целое число");

bug = Console.ReadLine();

}

}

x = int.Parse(bug);

}

}

static string PrintMenu()

{

Console.WriteLine(@"Выберите пункт меню

1. Добавить объект в коллекцию

2. Удалить объект из коллекции

3. Распечатать элементы коллекции

0. Выход");

string answer = Console.ReadLine();

Console.WriteLine();

while (answer != "0" && answer != "1" && answer != "2" && answer != "3")

{

Console.WriteLine("Введите целое число от 0 до 3");

answer = Console.ReadLine();

}

return answer;

}

static void AddItem(ref Dictionary<Print\_edition, Journal> DictCollection)

{

Console.WriteLine();

Console.WriteLine(@"Введите, каким способом вы хотите добавить элемент

1. С клавиатуры

2. Автоматически");

string answer = Console.ReadLine();

while (answer != "1" && answer != "2")

{

Console.WriteLine("\nВведите или 1, или 2");

answer = Console.ReadLine();

}

if (answer == "1")

{

Console.WriteLine("\nВведите название печатного издания");

string Name = Console.ReadLine();

CheckInt(out int PagesP, "\nВведите кол-во страниц печатного издания", 0);

Print\_edition AddElem = new Print\_edition(PagesP, Name);

while (DictCollection.ContainsKey(AddElem))

{

Console.WriteLine("\nТакой ключ уже есть в коллекции, введите другой");

Console.WriteLine("\nВведите название печатного издания");

Name = Console.ReadLine();

CheckInt(out PagesP, "\nВведите кол-во страниц печатного издания", 0);

AddElem = new Print\_edition(PagesP, Name);

}

Console.WriteLine("\nВведите название журнала");

Name = Console.ReadLine();

CheckInt(out PagesP, "\nВведите кол-во страниц журнала", 0);

Console.WriteLine("\nВведите период журнала");

string Period = Console.ReadLine();

Journal AddElemJ = new Journal(Name, PagesP, Period);

DictCollection.Add(AddElem, AddElemJ);

Console.WriteLine("\nЭлемент добавлен\n");

}

if (answer == "2")

{

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

AddItem.Init();

while (DictCollection.ContainsKey(AddItem))

{

AddItem.Init();

}

Journal AddItemJ = new Journal();

AddItemJ.Init();

DictCollection.Add(AddItem, AddItemJ);

Console.WriteLine("\nЭлемент добавлен");

}

}

static void DelItem(ref Dictionary<Print\_edition, Journal> DictCollection)

{

Console.WriteLine("\nВведите название ключа элемента, который вы хотите удалить");

string Name = Console.ReadLine();

Console.WriteLine("Введите кол-во страниц ключа элемента, который вы хотите удалить");

CheckInt(out int Pages, "", 0);

Print\_edition p = new Print\_edition(Name, Pages);

if (DictCollection.ContainsKey(p))

{

DictCollection.Remove(p);

Console.WriteLine("\nЭлемент удалён\n");

}

else

Console.WriteLine("\nКлюч не найден\n");

}

static void PrintValueMore(ref Dictionary<Print\_edition, Journal> DictCollection)

{

foreach(var x in DictCollection)

{

if (x.Value.Pages\_num > 1000)

Console.WriteLine(x.Value);

}

}

static void PrintValueLess(ref Dictionary<Print\_edition, Journal> DictCollection)

{

foreach (var x in DictCollection)

{

if (x.Value.Pages\_num < 1000)

Console.WriteLine(x.Value);

}

}

static void PrintMaxPagesPrintEdition(ref Dictionary<Print\_edition, Journal> DictCollection)

{

int max = 0;

Print\_edition temp = new();

foreach(var x in DictCollection)

{

if (x.Key.Pages\_num > max)

{

max = x.Key.Pages\_num;

temp = x.Key;

}

}

Console.WriteLine(temp);

}

static void Main(string[] args)

{

Dictionary<Print\_edition, Journal> DictCollection = new();

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

{

Print\_edition elem = new();

elem.Init();

Journal Jelem = new();

Jelem.Init();

DictCollection.Add(elem, Jelem);

}

string answer = "sdf";

while (answer != "0")

{

answer = PrintMenu();

switch (answer)

{

case "0":

break;

case "1":

AddItem(ref DictCollection);

break;

case "2":

DelItem(ref DictCollection);

break;

case "3":

foreach (var x in DictCollection)

Console.WriteLine($"Key:\n\n{x.Key}\nValue:\n\n{x.Value}");

break;

}

}

Console.WriteLine(@"Введите, какой запрос вы хотите выбрать

1. Вывести значения с числом страниц большим 1000

2. Вывести значения с числом страниц меньшим 1000

3. Вывести ключ с максимальным числом страниц");

answer = Console.ReadLine();

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

{

Console.WriteLine("Введите число от 1 до 3");

answer = Console.ReadLine();

}

Console.WriteLine();

if (answer == "1")

PrintValueMore(ref DictCollection);

if (answer == "2")

PrintValueLess(ref DictCollection);

if (answer == "3")

PrintMaxPagesPrintEdition(ref DictCollection);

Console.WriteLine("Нажмите любую клавишу, чтобы продолжить");

Console.ReadKey();

Console.Clear();

Console.WriteLine("Исходная коллекция\n\n");

foreach (var x in DictCollection)

{

Console.WriteLine($"Key:\n\n{x.Key}\nValue:\n\n{x.Value}");

}

Dictionary<Print\_edition, Journal> DictCollectionClone = new();

Print\_edition temp = new();

foreach (var x in DictCollection)

{

temp = (Print\_edition)x.Key.Clone();

Journal tempJ = (Journal)x.Value.Clone();

DictCollectionClone.Add(temp, tempJ);

}

Dictionary<Print\_edition, Journal> DictCollectionShallowCopy = new();

foreach (var x in DictCollection)

{

temp = (Print\_edition)x.Key.ShallowCopy();

Journal tempJ = (Journal)x.Value.ShallowCopy();

DictCollectionShallowCopy.Add(temp, tempJ);

}

Console.WriteLine("\n");

temp = new();

foreach (var x in DictCollection)

{

temp = x.Key;

break;

}

Journal j = new Journal();

j.Init();

DictCollection[temp] = j;

Console.WriteLine("Изменённая коллекция\n\n");

foreach (var x in DictCollection)

{

Console.WriteLine($"Key:\n\n{x.Key}\nValue:\n\n{x.Value}");

}

Console.WriteLine("\n\nКлон\n\n");

foreach (var x in DictCollectionClone)

{

Console.WriteLine($"Key:\n\n{x.Key}\nValue:\n\n{x.Value}");

}

Console.WriteLine("\n\nПоверхностное копирование\n\n");

foreach (var x in DictCollectionShallowCopy)

{

Console.WriteLine($"Key:\n\n{x.Key}\nValue:\n\n{x.Value}");

}

var SortedDict = new SortedDictionary<Print\_edition, Journal>(DictCollection, new SortByPages());

Console.WriteLine("\n\nОтсортированная коллекция\n\n");

foreach (var x in SortedDict)

Console.WriteLine($"Key:\n\n{x.Key}\nValue:\n\n{x.Value}");

Console.WriteLine("Введите название ключа, который вы хотите найти");

string Name = Console.ReadLine();

CheckInt(out int PagesNum, "\nВведите кол-во страниц ключа, который вы хотите найти", 0);

temp = new Print\_edition(PagesNum, Name);

if (DictCollection.ContainsKey(temp))

Console.WriteLine("\nЭлемент найден");

else

Console.WriteLine("\nЭлемент не найден");

}

}

}

**ПРИЛОЖЕНИЕ Г**

**Класс TestCollections**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Лаба\_10;

using \_11\_lab;

namespace \_11\_lab

{

public class TestCollections

{

public List<Journal> firstCollection = new();

public List<string> firstCollectionS = new();

public Dictionary<Print\_edition, Journal> secondCollection = new();

public Dictionary<string, Journal> secondCollectionS = new();

public TestCollections()

{

for (int i = 0; i < 1000; i++)

{

Journal temp = new Journal();

temp.Init();

Print\_edition tempP = temp.BasePrint\_edition;

while (!secondCollectionS.TryAdd(tempP.ToString(), temp))

{

temp.Init();

tempP = temp.BasePrint\_edition;

}

firstCollection.Add(temp);

firstCollectionS.Add(temp.ToString());

secondCollection.Add(tempP, temp);

}

}

public void ShowFirstCollection()

{

foreach (Journal j in firstCollection)

Console.WriteLine(j);

}

public void ShowFirstCollectionS()

{

foreach (string x in firstCollectionS)

Console.WriteLine(x);

}

public void ShowSecondCollection()

{

foreach (var x in secondCollection)

Console.WriteLine($"Key:\n\n{x.Key}\nValue:\n\n{x.Value}");

}

public void ShowSecondCollectionS()

{

foreach (var x in secondCollectionS)

Console.WriteLine($"Key:\n\n{x.Key}\nValue:\n\n{x.Value}");

}

public void AddItem()

{

Console.WriteLine();

Console.WriteLine(@"Введите, каким способом вы хотите добавить элемент

1. С клавиатуры

2. Автоматически");

string answer = Console.ReadLine();

while (answer != "1" && answer != "2")

{

Console.WriteLine("Введите или 1, или 2");

answer = Console.ReadLine();

}

if (answer == "1")

{

Console.WriteLine("\nВведите название журнала");

string Name = Console.ReadLine();

Program.CheckInt(out int PagesP, "\nВведите кол-во страниц журнала", 0);

Console.WriteLine("\nВведите период журнала");

string Period = Console.ReadLine();

Journal addElem = new Journal(Name, PagesP, Period);

while (secondCollection.ContainsKey(addElem.BasePrint\_edition))

{

Console.WriteLine("\nТакой ключ (название - кол-во страниц) уже содержится в словаре, введите элемент заново");

Console.WriteLine("\nВведите название журнала");

Name = Console.ReadLine();

Program.CheckInt(out PagesP, "\nВведите кол-во страниц журнала", 0);

Console.WriteLine("\nВведите период журнала");

Period = Console.ReadLine();

addElem = new Journal(Name, PagesP, Period);

}

firstCollection.Add(addElem);

firstCollectionS.Add(addElem.ToString());

secondCollection.Add(addElem.BasePrint\_edition, addElem);

secondCollectionS.Add(addElem.BasePrint\_edition.ToString(), addElem);

}

if (answer == "2")

{

Journal addElem = new Journal();

addElem.Init();

while (secondCollection.ContainsKey(addElem.BasePrint\_edition))

{

addElem.Init();

}

firstCollection.Add(addElem);

firstCollectionS.Add(addElem.ToString());

secondCollection.Add(addElem.BasePrint\_edition, addElem);

secondCollectionS.Add(addElem.BasePrint\_edition.ToString(), addElem);

}

}

public void DellItem()

{

Console.WriteLine("\nВведите название журнала, который вы хотите удалить");

string Name = Console.ReadLine();

Program.CheckInt(out int PagesP, "\nВведите кол-во страниц журнала, который вы хотите удалить", 0);

Console.WriteLine("\nВведите период журнала, который вы хотите удалить");

string Period = Console.ReadLine();

Journal addElem = new Journal(Name, PagesP, Period);

if (firstCollection.Contains(addElem))

{

firstCollection.Remove(addElem);

firstCollectionS.Remove(addElem.ToString());

secondCollection.Remove(addElem.BasePrint\_edition);

secondCollectionS.Remove(addElem.BasePrint\_edition.ToString());

Console.WriteLine("\nЭлемент удалён\n");

}

else

Console.WriteLine("\nВ коллекциях нет такого элемента\n");

}

}

}

**ПРИЛОЖЕНИЕ Д**

**Класс Program**

using System;

using Лаба\_10;

using System.Collections.Generic;

using System.Diagnostics;

namespace \_11\_lab

{

public class Program

{

static void Main(string[] args)

{

TestCollections t5 = new();

t5.ShowSecondCollectionS();

t5.AddItem();

Console.WriteLine();

t5.ShowSecondCollectionS();

t5.DellItem();

t5.ShowSecondCollectionS();

Stopwatch time = new();

Journal first = new Journal(t5.firstCollection[0].Name, t5.firstCollection[0].Pages\_num, t5.firstCollection[0].Period);

Journal middle = new Journal(t5.firstCollection[499].Name, t5.firstCollection[499].Pages\_num, t5.firstCollection[499].Period);

Journal end = new Journal(t5.firstCollection[999].Name, t5.firstCollection[999].Pages\_num, t5.firstCollection[999].Period);

Journal notInCollecton = new Journal("Elem", 0, "Period");

time.Start();

t5.firstCollection.Contains(first);

time.Stop();

int firstColConFirst = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.firstCollection.Contains(middle);

time.Stop();

int firstColConMiddle = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.firstCollection.Contains(end);

time.Stop();

int firstColConEnd = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.firstCollection.Contains(notInCollecton);

time.Stop();

int firstColConNotInCol = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.firstCollectionS.Contains(first.ToString());

time.Stop();

int firstColConSFirst = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.firstCollectionS.Contains(middle.ToString());

time.Stop();

int firstColConSMiddle = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.firstCollectionS.Contains(end.ToString());

time.Stop();

int firstColConSEnd = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.firstCollectionS.Contains(notInCollecton.ToString());

time.Stop();

int firstColConSNotInCol = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollectionS.ContainsKey(first.BasePrint\_edition.ToString());

time.Stop();

int secondColConSFirst = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollectionS.ContainsKey(middle.BasePrint\_edition.ToString());

time.Stop();

int secondColConSMiddle = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollectionS.ContainsKey(end.BasePrint\_edition.ToString());

time.Stop();

int secondColConSEnd = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollectionS.ContainsKey(notInCollecton.BasePrint\_edition.ToString());

time.Stop();

int secondColConSNotInCol = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollection.ContainsKey(first.BasePrint\_edition);

time.Stop();

int secondColConFirst = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollection.ContainsKey(middle.BasePrint\_edition);

time.Stop();

int secondColConMiddle = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollection.ContainsKey(end.BasePrint\_edition);

time.Stop();

int secondColConEnd = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollection.ContainsKey(notInCollecton.BasePrint\_edition);

time.Stop();

int secondColConNotInCol = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollection.ContainsValue(first);

time.Stop();

int secondColConFirstValue = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollection.ContainsValue(middle);

time.Stop();

int secondColConMiddleValue = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollection.ContainsValue(end);

time.Stop();

int secondColConEndValue = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

time.Start();

t5.secondCollection.ContainsValue(notInCollecton);

time.Stop();

int secondColConNotInColValue = Convert.ToInt32(time.ElapsedTicks);

time.Reset();

Console.WriteLine($"Время поиска первого элемента в коллекции List<Journal>: {firstColConFirst} тиков\n" +

$"Время поиска среднего элемента в коллекции List<Journal>: {firstColConMiddle} тиков\n" +

$"Время поиска последнего элемента в коллекции List<Journal>: {firstColConEnd} тиков\n" +

$"Время поиска несуществующего элемента в коллекции List<Journal>: {firstColConNotInCol} тиков\n\n" +

$"Время поиска первого элемента в коллекции List<String>: {firstColConSFirst} тиков\n" +

$"Время поиска среднего элемента в коллекции List<String>: {firstColConSMiddle} тиков\n" +

$"Время поиска последнего элемента в коллекции List<String>: {firstColConSEnd} тиков\n" +

$"Время поиска несуществующего элемента в коллекции List<String>: {firstColConSNotInCol} тиков\n\n" +

$"Время поиска элемента по первому ключу в коллекции Dictionary<Print\_edition, Journal>: {secondColConFirst} тиков\n" +

$"Время поиска элемента по среднему ключу в коллекции Dictionary<Print\_edition, Journal>: {secondColConMiddle} тиков\n" +

$"Время поиска элемента по последнему ключу в коллекции Dictionary<Print\_edition, Journal>: {secondColConEnd} тиков\n" +

$"Время поиска элемента по несуществующему ключу в коллекции Dictionary<Print\_edition, Journal>: {secondColConNotInCol} тиков\n\n" +

$"Время поиска элемента по первому ключу в коллекции Dictionary<string, Journal>: {secondColConSFirst} тиков\n" +

$"Время поиска элемента по среднему ключу в коллекции Dictionary<string, Journal>: {secondColConSMiddle} тиков\n" +

$"Время поиска элемента по последнему ключу в коллекции Dictionary<string, Journal>: {secondColConSEnd} тиков\n" +

$"Время поиска элемента по несуществующему ключу в коллекции Dictionary<string, Journal>: {secondColConSNotInCol} тиков\n\n" +

$"Время поиска значения первого элемента в коллекции Dictionary<Print\_edition, Journal>: {secondColConFirstValue} тиков\n" +

$"Время поиска значения среднего элемента в коллекции Dictionary<Print\_edition, Journal>: {secondColConMiddleValue} тиков\n" +

$"Время поиска значения последнего элемента в коллекции Dictionary<Print\_edition, Journal>: {secondColConEndValue} тиков\n" +

$"Время поиска значения несуществующего элемента в коллекции Dictionary<Print\_edition, Journal>: {secondColConNotInColValue} тиков\n");

}

}

}