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 №13 (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](#_Toc106123747)

2 [Диаграмма классов 6](#_Toc106123748)

3 [Код программы 7](#_Toc106123749)

3.1 [Код класса JournalJ 7](#_Toc106123750)

3.2 [Код класса CollectionHandlerEventArgs 7](#_Toc106123751)

3.3 [Код класса NewStack<T> 7](#_Toc106123752)

3.4 [Код класса Program 7](#_Toc106123753)

4 [Код Unit-тестов 8](#_Toc106123754)

5 [Анализ покрытия кода тестами 9](#_Toc106123755)

[ПРИЛОЖЕНИЕ А *Код класса JournalJ* 10](#_Toc106123756)

[ПРИЛОЖЕНИЕ Б *Код класса CollectionHandlerEventArgs* 11](#_Toc106123757)

[ПРИЛОЖЕНИЕ В *Код класса NewStack<T>* 12](#_Toc106123758)

[ПРИЛОЖЕНИЕ Г *Код класса Program* 16](#_Toc106123759)

[ПРИЛОЖЕНИЕ Д *Код Unit-тестов* 18](#_Toc106123760)

1 Постановка задачи

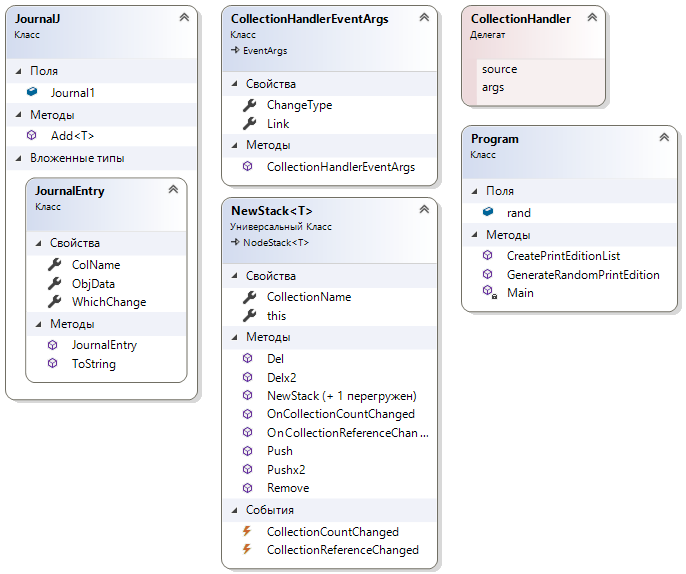
Задача (с уточнениями к 16 варианту) формулируется следующим образом:

1. Создать иерархию классов. Для каждого класса реализовать конструктор без параметров, с параметрами, свойства для доступа к полям объектов, метод для автоматического формирования объектов. Перегрузить метод ToString() для формирования строки со значениями всех полей класса (библиотека из лаб. 10).
2. Определить класс **MyNewCollection<T> (NewStack<T> - вариант 16)** производный от класса MyCollection<T> (NodeStack<T>). Класс MyСollection<T> (NodeStack<T>) взять из лабораторной работы №12 (лучше, как библиотеку). В классе должны быть реализованы в классе методы для заполнения коллекции (элементы коллекции формируются автоматически), добавления элементов коллекции, удаления элементов коллекции, очистки коллекции, реализован итератор для доступа к элементам коллекции, реализовано свойство Length (только для чтения), содержащее текущее количество элементов коллекции.
3. В класс MyNewCollection (NewStack) добавить события, которые извещают об изменениях в коллекции. Коллекция изменяется:
   1. При удалении/добавлении элементов
   2. При изменении одной из входящих в коллекцию ссылок, например, когда одной из ссылок присваивается новое значение.

В этом случае в соответствующих методах или свойствах класса генерируются события.

1. В класс MyNewCollection (NewStack) добавить:
   1. Открытое автореализуемое свойство типа string с названием коллекции.
   2. Метод void Add(T obj) для добавления элемента в коллекцию.
   3. Метод void AddDefault() для добавления случайного элемента в коллекцию.
   4. Метод bool Remove (int j) для удаления элемента с номером j. Если в списке нет элемента с номером j, метод возвращает значение false.
   5. Индексатор (с методами get и set) с целочисленным индексом для доступа к элементу с заданным номером.
2. Для событий, извещающих об изменениях в коллекции, определяется делегат CollectionHandler: void **CollectionHandler (object source, CollectionHandlerEventArgs args).**
3. Для передачи информации о событии определить класс **CollectionHandlerEventArgs**, производный от класса System.EventArgs, который содержит:
   1. Открытое автореализуемое свойство типа string с информацией о типе изменений в коллекции.
   2. Открытое автореализуемое свойство для ссылки на объект, с которым связаны изменения.
   3. Конструкторы для инициализации класса.
4. В класс MyNewCollection (NewStack) добавить два события типа CollectionHandle:
   1. **CollectionCountChanged**, которое происходит при добавлении нового элемента в коллекцию или при удалении элемента из коллекции. Через объект CollectionHandlerEventArgs cобытие передает строку с информацией о том, что в коллекцию был добавлен новый элемент или из нее был удален элемент, ссылку на добавленный или удаленный элемент.
   2. **CollectionReferenceChanged**, которое происходит, когда одной из ссылок, входящих в коллекцию, присваивается новое значение. Через объект CollectionHandlerEventArgs событие передает строку с информацией о том, что был заменен элемент в коллекции, и ссылку на новый элемент.
5. Событие CollectionCountChanged бросают следующие методы класса MyNewCollection (NewStack):
   1. AddDefaults() – добавить случайный объект.
   2. Add (Т obj ) – добавить объект.
   3. Remove (int index) – удалить элемент по индексу.
6. Событие CollectionReferenceChanged бросает метод set индексатора, определенного в классе MyNewCollection (NewStack).
7. Информация об изменениях коллекции записывается в класс **Journal (JournalJ),** который хранит информацию в списке объектов типа **JournalEntry**. Каждый объект типа JournalEntry содержит информацию об отдельном изменении, которое произошло в коллекции. JournalEntry содержит:
   1. Открытое автореализуемое свойство типа string с названием коллекции, в которой произошло событие.
   2. Открытое автореализуемое свойство типа string с информацией о типе изменений в коллекции.
   3. Открытое автореализуемое свойство типа string c данными объекта, с которым связаны изменения в коллекции.
   4. Конструктор для инициализации полей класса.
   5. Перегруженную версию метода string ToString().
8. Написать демонстрационную программу, в которой:
   1. Создать две коллекции MyNewCollection (NewStack).
   2. Создать два объекта типа Journal (JournalJ), **один объект Journal (JournalJ) подписать на события CollectionCountChanged и CollectionReferenceChanged из первой коллекции, другой объект Journal (JournalJ) подписать на события CollectionReferenceChanged из обеих коллекций.**
9. Внести изменения в коллекции MyNewCollection (NewStack):
   1. Добавить элементы в коллекции.
   2. Удалить некоторые элементы из коллекций.
   3. Присвоить некоторым элементам коллекций новые значения.
10. Вывести данные обоих объектов Journal (JournalJ).

2 Диаграмма классов

******Диаграмма классов представлена на рисунке 1.

***Рисунок 1 – Диаграмма классов***

3 Код программы

Код класса JournalJ

См. [ПРИЛОЖЕНИЕ А](#ПриложениеА).

Код класса CollectionHandlerEventArgs

См. [ПРИЛОЖЕНИЕ Б](#ПриложениеБ).

Код класса NewStack<T>

См. [ПРИЛОЖЕНИЕ В](#ПриложениеВ).

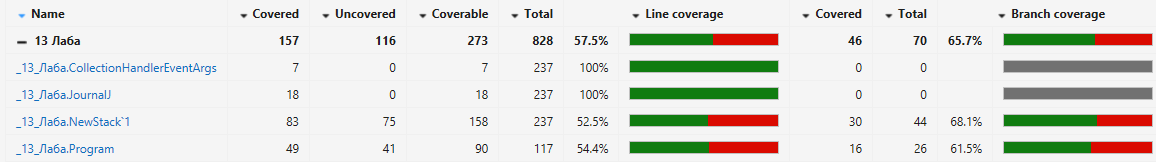
Код класса Program

См. [ПРИЛОЖЕНИЕ Г](#ПриложениеГ).

4 Код Unit-тестов

См. [ПРИЛОЖЕНИЕ Д](#ПриложениеД).

5 Анализ покрытия кода тестами

******Покрытие кода тестами представлено на рисунке 2.

***Рисунок 2 – Покрытие кода тестами***

ПРИЛОЖЕНИЕ А

**Код класса JournalJ**

public class JournalJ //Класс, в котором содержится информация об изменениях в коллекции

{

public List<JournalEntry> Journal1 = new(); //Список для записи изменений

public class JournalEntry //Класс, созданный в качестве целового для записи в список изменений

{

public string ColName { get; set; }

public string WhichChange { get; set; }

public string ObjData { get; set; }

public JournalEntry(string Name = "", string Change = "", string Data = "")

{

ColName = Name;

WhichChange = Change;

ObjData = Data;

}

public override string ToString()

{

return "Название коллекции, в которой произошло изменение: " + ColName + "\n" +

"Тип изменения: " + WhichChange + "\n" + "Данные объекта, с которым связаны изменение:\n" +

ObjData + "\n";

}

}

public void Add<T>(object source, CollectionHandlerEventArgs args) //Метод-обработчик события

{

Journal1.Add(new JournalEntry(((NewStack<T>)source).CollectionName, args.ChangeType, args.Link)); //Добавляет запись в журнал об изменении в объекте

}

}

ПРИЛОЖЕНИЕ Б

**Код класса CollectionHandlerEventArgs**

public class CollectionHandlerEventArgs: EventArgs //Класс для передачи информации о событии

{

public string ChangeType { get; set; } //Тип изменений в коллекции

public string Link { get; set; } //Изменённый объект

public CollectionHandlerEventArgs(string x = "", string link = "")

{

ChangeType = x;

Link = link;

}

}

ПРИЛОЖЕНИЕ В

**Код класса NewStack<T>**

public delegate void CollectionHandler(object source, CollectionHandlerEventArgs args); //Непосредственно делегат

public class NewStack<T>: NodeStack<T>

{

public event CollectionHandler CollectionCountChanged; //Событие, использующееся при удалении-добавлении объекта

public event CollectionHandler CollectionReferenceChanged; //Событие, использующееся при изменении ссылки на объект

public void OnCollectionCountChanged(object source, CollectionHandlerEventArgs e) //Метод, вызывающий событие CollectionCountChanged в случае, если на событие есть подписанные методы-обработчики

{

CollectionCountChanged?.Invoke(source, e);

}

public void OnCollectionReferenceChanged(object source, CollectionHandlerEventArgs e) //Метод, вызывающий событие CollecitonReferenceChanged в случае, если на событии есть подписанные методы-обработчики

{

CollectionReferenceChanged?.Invoke(source, e);

}

public string CollectionName { get; set; }

public NewStack(): base() { }

public NewStack(List<T> p, string Name): base(p)

{

CollectionName = Name;

}

public T this[int index] //Реализация индексатора

{

get

{

if (index >= 0 && index < count)

{

if (index == 0)

return head.Data;

Node<T> temp = head;

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

temp = temp.Pred;

return temp.Data;

}

else //Если выбран недопустимый индекс

{

Console.WriteLine("Вы вышли за пределы стека\n");

return default;

}

}

set

{

if (index >= 0 && index < count)

{

if (index == 0)

{

head.Data = value;

OnCollectionReferenceChanged(this, new CollectionHandlerEventArgs("Был заменён элемент в коллекции", head.Data.ToString())); //После изменения ссылки вызывается событие CollectionReferenceChanged

}

Node<T> temp = head;

for (int i = 0; i < index; i++) //Переход к элементу стека с индексом index

temp = temp.Pred;

temp.Data = value; //Присваивание выбранного значению элемента стека с индексом index

OnCollectionReferenceChanged(this, new CollectionHandlerEventArgs("Был заменён элемент в коллекции", temp.Data.ToString())); //После изменения ссылки вызывается событие CollectionReferenceChanged

}

else

Console.WriteLine("Вы вышли за пределы стека\n");

}

}

new public void Push(T item) //Метод из NodeStack (12 лабораторная работа) с добавление событий внутри метода при добавлении объекта

{

Node<T> node = new Node<T>(item);

if (head.Data == null)

{

head = new Node<T>(item);

count += 1;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Добавление элемента в коллекцию", node.Data.ToString())); //После добавления элемента вызывается событие CollectionCountChanged

return;

}

node.Pred = head;

head.Next = node;

head = node;

Node<T> smt = new Node<T>();

smt = head;

while (smt.Pred != null)

smt = smt.Pred;

head.First = smt;

count++;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Добавление элемента в коллекцию", node.Data.ToString())); //После добавления элемента вызывается событие CollectionCountChanged

}

new public void Pushx2(T item, T item1) //Метод из NodeStack (12 лабораторная работа) с добавление событий внутри метода при добавлении объекта

{

if (head.Data == null)

{

Node<T> el1 = new Node<T>(item1);

head = el1;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Добавление элемента в коллекцию", el1.Data.ToString())); //После добавления элемента вызывается событие CollectionCountChanged

Node<T> el2 = new Node<T>(item);

head.First = el2;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Добавление элемента в коллекцию", el2.Data.ToString())); //После добавления элемента вызывается событие CollectionCountChanged

head.First.Next = head;

head.Pred = head.First;

count += 2;

return;

}

Node<T> itemN = new(item);

Node<T> itemN1 = new(item1);

head.Next = itemN1;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Добавление элемента в коллекцию", itemN1.Data.ToString())); //После добавления элемента вызывается событие CollectionCountChanged

head.Next.Pred = head;

Node<T> smt = head.First;

head = head.Next;

smt.Pred = itemN;

smt.Pred.Next = smt;

smt = smt.Pred;

head.First = smt;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Добавление элемента в коллекцию", itemN.Data.ToString())); //После добавления элемента вызывается событие CollectionCountChanged

count += 2;

}

new public bool Del() //Метод из NodeStack (12 лабораторная работа) с добавление событий внутри метода при удалении объекта

{

if (IsEmpty)

{

Console.WriteLine("Стек пуст");

return false;

}

if (head.Pred == null)

{

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Удаление элемента из коллекции", head.Data.ToString())); //Перед удалением элемента вызывается событие CollectionCountChanged

head.Data = default;

}

else

{

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Удаление элемента из коллекции", head.Data.ToString())); //Перед удалением элемента вызывается событие CollectionCountChanged

head.Pred.First = head.First;

head = head.Pred;

head.Next = null;

}

count -= 1;

return true;

}

new public bool Delx2() //Метод из NodeStack (12 лабораторная работа) с добавление событий внутри метода при удалении объектов

{

if (IsEmpty)

{

Console.WriteLine("Стек пуст");

return false;

}

if (Count == 1)

{

Console.WriteLine("В стеке есть только 1 элемент. Удаление 2 элементов невозможно\n");

return false;

}

else if (Count == 2)

{

head = head.Pred;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Удаление элемента из коллекции", head.Next.Data.ToString())); //Перед удалением элемента вызывается событие CollectionCountChanged

head.Next = null;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Удаление элемента из коллекции", head.Data.ToString())); //Перед удалением элемента вызывается событие CollectionCountChanged

head.Data = default;

count -= 2;

return true;

}

else

{

head.Pred.First = head.First;

head = head.Pred;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Удаление элемента из коллекции", head.Next.Data.ToString())); //Перед удалением элемента вызывается событие CollectionCountChanged

head.Next = null;

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Удаление элемента из коллекции", head.First.Data.ToString())); //Перед удалением элемента вызывается событие CollectionCountChanged

head.First = head.First.Next;

head.First.Pred = null;

count -= 2;

return true;

}

}

public bool Remove (int j) //Добавленный метод для удаления по индексу в стеке

{

if (j > count - 1 || j < 0) //Если нет элемента с таким индексом, возврат false

return false;

Node<T> temp = head;

if (j == 0) //Особая обработка для 0 индекса

{

head.Pred.First = head.First; //Смещаем ссылку на первый элемент на 1 элемент назад

head = head.Pred; //Смещаем верхушки стека на 1 элемент назад

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Удаление элемента из коллекции", head.Next.Data.ToString())); //Перед удалением элемента вызывается событие CollectionCountChanged

head.Next = null; //Присваиваем следующему элементу значение null (непосредственно удаление элемента)

count -= 1;

return true;

}

for (int i = 0; i < j; i ++) //Движение к элементу с заданным индексом

{

temp = temp.Pred;

}

OnCollectionCountChanged(this, new CollectionHandlerEventArgs("Удаление элемента из коллекции", temp.Data.ToString())); //Перед удалением элемента вызывается событие CollectionCountChanged

temp.Data = default; //Присваивание информационному полю заданного объекта значение default

temp.Next.Pred = temp.Pred; //Ссылке следующего объекта присваиваем ссылку на предыдущий

temp.Pred = temp.Next; //Ссылке предыдущего объекта присваиваем ссылку на следующий

count -= 1;

return true;

}

}

ПРИЛОЖЕНИЕ Г

**Код класса Program**

public class Program

{

public static Random rand = new();

public static List<Print\_edition> CreatePrintEditionList(int size = 10)

{

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

Print\_edition data = new();

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

{

data = new();

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

if (number == 1)

{

data = new Print\_edition();

data.Init();

}

if (number == 2)

{

data = new Book();

data.Init();

}

if (number == 3)

{

data = new Textbook();

data.Init();

}

if (number == 4)

{

data = new Journal();

data.Init();

}

temp.Add(data);

}

return temp;

}

public static Print\_edition GenerateRandomPrintEdition()

{

Print\_edition data = new();

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

if (number == 1)

{

data = new Print\_edition();

data.Init();

}

if (number == 2)

{

data = new Book();

data.Init();

}

if (number == 3)

{

data = new Textbook();

data.Init();

}

if (number == 4)

{

data = new Journal();

data.Init();

}

return data;

}

static void Main(string[] args)

{

//Создание коллекций

NewStack<Print\_edition> f = new(CreatePrintEditionList(10), "Коллекция 1");

NewStack<Print\_edition> f1 = new(CreatePrintEditionList(10), "Коллекция 2");

//Создание объектов класса Journal для записи изменений, произошедших в коллекции

JournalJ j = new();

JournalJ j1 = new();

//Подписывание методов-обработчиков для событий (метод добавляет запись об изменении в соответствующие коллекции)

f.CollectionCountChanged += j.Add<Print\_edition>;

f.CollectionReferenceChanged += j.Add<Print\_edition>;

f.CollectionReferenceChanged += j1.Add<Print\_edition>;

f1.CollectionReferenceChanged += j1.Add<Print\_edition>;

//Производим изменения (добавление, удаление, присваивание новых значений) в коллекции

f.Pushx2(GenerateRandomPrintEdition(), GenerateRandomPrintEdition());

f.Del();

f[4] = GenerateRandomPrintEdition();

f1.Delx2();

f1.Delx2();

f1[4] = GenerateRandomPrintEdition();

f1.Push(GenerateRandomPrintEdition());

//Вывод журналов событий

bool isPrint = false;

foreach (JournalJ.JournalEntry x in j.Journal1)

{

isPrint = true;

Console.WriteLine(x);

}

if (!isPrint)

Console.WriteLine("Журнал 1 пуст");

isPrint = false;

foreach (JournalJ.JournalEntry x in j1.Journal1)

{

isPrint = true;

Console.WriteLine(x);

}

if (!isPrint)

Console.WriteLine("Журнал 2 пуст");

}

}

ПРИЛОЖЕНИЕ Д

**Код Unit-тестов**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using \_13\_Лаба;

using EditionLib;

namespace \_13Test

{

[TestClass]

public class CollectionHandlerEventArgsTestClass //Тестирование класса CollectionHandlerEventArgs

{

[TestMethod]

public void CollectionHandlerEventArgsNullConstructorTest() //Тестирование конструктора с пустыми параметрами

{

CollectionHandlerEventArgs f = new();

Assert.AreEqual("", f.ChangeType);

Assert.AreEqual("", f.Link);

}

[TestMethod]

public void CollectionHandlerEventArgsNotNullConstructorTest() //Тестирование конструктора с непустыми параметрами

{

CollectionHandlerEventArgs f = new("Добавление", "Печатное издание 1");

Assert.AreEqual("Добавление", f.ChangeType);

Assert.AreEqual("Печатное издание 1", f.Link);

}

}

[TestClass]

public class JournalEntryTestClass //Тестирование класса JournalEntry

{

[TestMethod]

public void JournalEntryNullConstructorTest() //Тестирование конструктора с пустыми параметрами

{

JournalJ.JournalEntry j = new();

Assert.AreEqual("", j.ColName);

Assert.AreEqual("", j.ObjData);

Assert.AreEqual("", j.WhichChange);

}

[TestMethod]

public void JournalEntryNotNullConstructorTest() //Тестирование конструктора с непустыми параметрами

{

JournalJ.JournalEntry j = new("Коллекция 1", "Изменение объекта", "Печатное издание 1");

Assert.AreEqual("Коллекция 1", j.ColName);

Assert.AreEqual("Печатное издание 1", j.ObjData);

Assert.AreEqual("Изменение объекта", j.WhichChange);

}

[TestMethod]

public void JournalEntryToStringTest() //Тестирование метода переопределённого метода ToString()

{

JournalJ.JournalEntry j = new("Коллекция 1", "Изменение объекта", "Печатное издание 1");

string tostring = j.ToString();

Assert.AreEqual("Название коллекции, в которой произошло изменение: Коллекция 1\nТип изменения: " +

"Изменение объекта\nДанные объекта, с которым связаны изменение:\nПечатное издание 1\n", tostring);

}

}

[TestClass]

public class JournalJTestClass //Тестирование класса JournalJ

{

[TestMethod]

public void AddInListTest() //Тестирование добавление событий в журнал класса JournalJ

{

NewStack<Print\_edition> f = new();

JournalJ j = new();

f.CollectionCountChanged += j.Add<Print\_edition>;

f.Push(Program.GenerateRandomPrintEdition());

Assert.AreEqual(1, j.Journal1.Count);

}

}

[TestClass]

public class NewStackTestClass //Тестирование класса NewStack

{

[TestClass]

public class IndexerTestClass //Тестирование индексатора

{

[TestMethod]

public void GetTest() //Тестирование блока Get

{

NewStack<Print\_edition> f = new();

Print\_edition add = new Print\_edition(1000, "Печатное издание 1");

f.Push(new Print\_edition(1000, "Печатное издание 1"));

Assert.AreEqual(add, f[0]); //Возврат элемента без перебора стека (верхушка)

Print\_edition add1 = new Print\_edition(400, "Печатное издание 2");

f.Push(add1);

Assert.AreEqual(add, f[1]); //Возврат элемента с перебором стека через цикл

Assert.AreEqual(default, f[-345]); //Тестирование выбора недопустимого индекса

}

[TestMethod]

public void SetTest() //Тестирование блока Set

{

NewStack<Print\_edition> f = new();

Print\_edition add = new Print\_edition(1000, "Печатное издание 1");

Print\_edition change = new Print\_edition(43, "Печатное издание 2");

f.Push(new Print\_edition(1000, "Печатное издание 1"));

f[0] = change; //Установка значения без перебора стека (для верхушки)

Assert.AreEqual(change, f[0]);

Print\_edition add1 = new Print\_edition(10, "Печатное издание 3");

f.Push(add1);

f[1] = add1; //Установка значения с перебором стека через цикл

Assert.AreEqual(add1, f[1]);

f[-34] = new Print\_edition(4, "Печатное издание 4"); //Тестирование установки недопустимого значения

}

}

[TestClass]

public class RemoveMethodTestClass //Тестирование метода Remove

{

[TestMethod]

public void BadIndexTest() //Тестирование удаления элемента с невозможным индексом

{

NewStack<Print\_edition> f = new(Program.CreatePrintEditionList(10), "Коллекция 1");

Assert.AreEqual(false, f.Remove(-4));

Assert.AreEqual(10, f.Count);

}

[TestMethod]

public void ZeroIndexTest() //Тестирование удаление элемента с нулевым индексом

{

NewStack<Print\_edition> f = new(Program.CreatePrintEditionList(10), "Коллекция 1");

Assert.AreEqual(true, f.Remove(0));

Assert.AreEqual(9, f.Count);

}

[TestMethod]

public void MiddleIndexTest() //Тестирование удаления элемента с серединным индексом

{

NewStack<Print\_edition> f = new(Program.CreatePrintEditionList(10), "Коллекция 1");

Assert.AreEqual(true, f.Remove(4));

Assert.AreEqual(9, f.Count);

}

}

[TestMethod]

public void ConstructorTest() //Тестирование конструкторов класса NewStack

{

NewStack<Print\_edition> f1 = new(Program.CreatePrintEditionList(10), "Коллекция 1"); //Тестирование конструктора со списком элементов типа PrintEdition (добавилось поле CollectionName)

Assert.AreEqual("Коллекция 1", f1.CollectionName);

}

[TestMethod]

public void NullDelegateTest() //Тестирование методов OnCollectionCountChanged и OnCollectionReferenceChanged

{

NewStack<Print\_edition> f = new(Program.CreatePrintEditionList(10), "Коллекция 1");

f.Push(Program.GenerateRandomPrintEdition());

f[4] = Program.GenerateRandomPrintEdition();

//Поскольку при отсутствии методов-обработчиков событий добавление и изменение объекта не вызывает ошибку, то метод, проверяющий наличие подписчиков у события, работает

}

}

}