МИНИСТРЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ

РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИИ

Отчет

О выполнении лабораторной работы № 1

«Управляемая тестами разработка классов.»

По дисциплине «Программирование на языке С#»

Вариант № 39

Выполнил ст. гр. ІТШІ-19-2: Принял:

Меденицкий Алексей Бибичков И. Е.

2020

**Цель работы**

Изучение особенностей разработки классов в среде Visual Studio 2019 на языке C# с использованием техники TDD (Test Driven Development).

**Задание №39 «Стек чисел»**

Данные класса: указатель на головку стека в динамическом списке элементов стека.

Функции класса: считывание без извлечения элемента стека, считывание с извлечением элемента стека, запись элемента в стек.

Включить в интерфейс класса, помимо функций, указанных в задании, следующие функции:

• все типы конструкторов

• функции доступа и инициализации;

• функцию сравнения объектов на равенство;

• функции ввода c консоли и вывода на консоль.

Указания к выполнению заданий этой группы: обязательно включить в класс конструктор, использующий в качестве одного из параметров одномерный или двумерный массив чисел, конструктор преобразования реализовывать не нужно. Рекомендуется реализовать вспомогательную функцию доступа к значениям, содержащимся в списке, по их координатам и использовать эту функцию для реализации прочих функций класса.

**Реализация класса с «пустыми» функциями.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace My

{

public class Stack

{

private class Node

{

public float data;

public Node next;

}

private Node begin;

public Stack() { }

public Stack(float[] arr) { }

public Stack(Stack s) { }

public void Push(float value) { throw new Exception(); }

public float Pop() { throw new Exception(); }

public float Peek() { throw new Exception(); }

public override bool Equals(object obj) { throw new Exception(); }

public void Print() { throw new Exception(); }

public void Input() { throw new Exception(); }

static void Main(string[] args)

{

Stack s = new Stack();

}

}

}

**Программная реализация тестов**

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using My;

namespace MyTest {

[TestClass]

public class StackTest {

[TestMethod]

public void Push\_Number\_Equals\_Pop () {

// Arrange

float number = 1.5f;

float expected = 1.5f;

Stack s = new Stack();

// Act

s.Push(number);

float actual = s.Pop();

// Assert

Assert.AreEqual(expected, actual, 0.001, "Push Pop values doesn't match");

}

[TestMethod]

public void Push\_Number\_Equals\_Peek () {

// Arrange

float number = 1.5f;

float expected = 1.5f;

Stack s = new Stack();

// Act

s.Push(number);

float actual = s.Peek();

// Assert

Assert.AreEqual(expected, actual, 0.0001, "Push Peek values doesn't match");

}

[TestMethod]

public void Pop\_Empty\_Stack\_Should\_Throw\_NullReferenceException() {

// Arrange

Stack s = new Stack();

//Act and Assert

Assert.ThrowsException<System.NullReferenceException>(() => s.Pop());

}

[TestMethod]

public void Peek\_Empty\_Stack\_Should\_Throw\_NullReferenceException() {

// Arrange

Stack s = new Stack();

//Act and Assert

Assert.ThrowsException<System.NullReferenceException>(() => s.Peek());

}

[TestMethod]

public void Pop\_Full\_Stack\_Should\_Remove\_Element() {

// Arrange

float expected = 1.5f;

Stack s = new Stack();

s.Push(3.4f);

s.Push(1.5f);

s.Push(1.2f);

// Act

s.Pop();

float actual = s.Pop();

// Assert

Assert.AreEqual(expected, actual, 0.0001, "Pop doesn't remove element");

}

[TestMethod]

public void Peek\_Full\_Stack\_Should\_Not\_Remove\_Element() {

// Arrange

float expected = 1.5f;

Stack s = new Stack();

s.Push(3.4f);

s.Push(1.2f);

s.Push(1.5f);

// Act

s.Peek();

float actual = s.Peek();

// Assert

Assert.AreEqual(expected, actual, 0.0001, "Peek removes element");

}

[TestMethod]

public void Equals\_Empty\_Stacks\_Return\_True () {

// Arrange

bool expected = true;

Stack a = new Stack();

Stack b = new Stack();

// Act

bool actual = a.Equals(b);

// Assert

Assert.AreEqual(expected, actual, "Stacks are not equal");

}

[TestMethod]

public void Equals\_Stacks\_With\_Same\_Data\_Return\_True () {

// Arrange

bool expected = true;

Stack a = new Stack();

a.Push(1.4f);

a.Push(2.4f);

a.Push(3.4f);

Stack b = new Stack();

b.Push(1.4f);

b.Push(2.4f);

b.Push(3.4f);

// Act

bool actual = a.Equals(b);

// Assert

Assert.AreEqual(expected, actual, "Stacks are not equal");

}

[TestMethod]

public void Equals\_Stacks\_With\_Different\_Data\_Same\_Size\_Return\_False () {

// Arrange

bool expected = false;

Stack a = new Stack();

a.Push(1.4f);

a.Push(2.4f);

a.Push(4.4f);

Stack b = new Stack();

b.Push(1.4f);

b.Push(2.4f);

b.Push(3.4f);

// Act

bool actual = a.Equals(b);

// Assert

Assert.AreEqual(expected, actual, "Stacks are equal");

}

[TestMethod]

public void Equals\_Stacks\_With\_Different\_Data\_Different\_Size\_Return\_False () {

// Arrange

bool expected = false;

Stack a = new Stack();

a.Push(1.4f);

a.Push(2.4f);

Stack b = new Stack();

b.Push(1.4f);

b.Push(2.4f);

b.Push(3.4f);

// Act

bool actual = a.Equals(b);

// Assert

Assert.AreEqual(expected, actual, "Stacks are equal");

}

[TestMethod]

public void Stack\_Full\_Array\_Init\_Data\_Should\_Equal\_Array\_Data () {

// Arrange

bool expected = true;

float[] arr = { 1, 2, 3, 4, 5 };

Stack s = new Stack(arr);

// Act

bool actual = true;

for (int i = arr.Length-1; i >= 0; --i)

if (arr[i] != s.Pop())

actual = false;

// Assert

Assert.AreEqual(expected, actual, "Stack data not equals array data");

}

[TestMethod]

public void Stack\_Empty\_Array\_Init\_Should\_Equal\_Empty\_Stack () {

// Arrange

bool expected = true;

float[] arr = {};

Stack a = new Stack(arr);

Stack b = new Stack();

// Act

bool actual = a.Equals(b);

// Assert

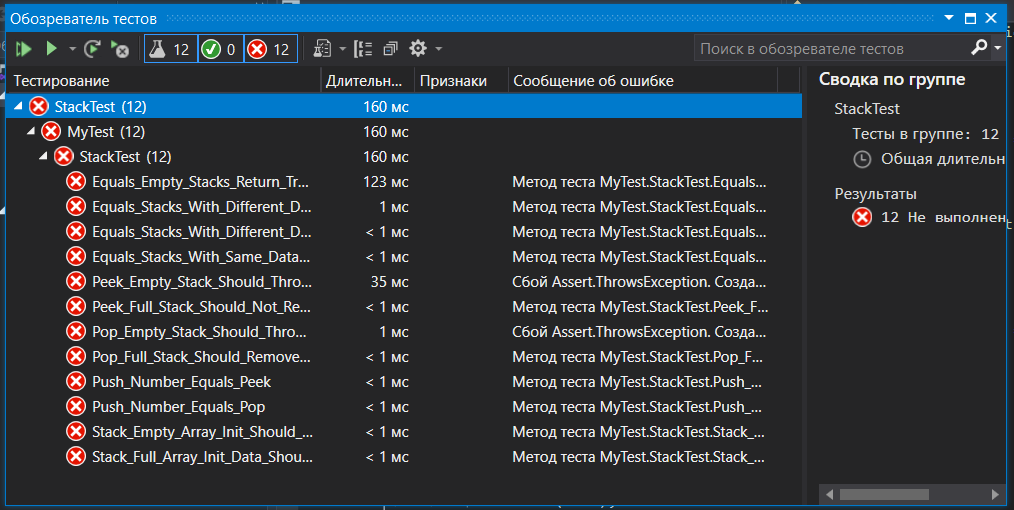
Assert.AreEqual(expected, actual, "Stacks are not equal");

}

}

}

**Скрины «красного» прохождения тестов.**



**Финальный интерфейс класса**

namespace My {

public class Stack {

private class Node {

public float data;

public Node next;

}

private Node begin;

Node Begin { get; set; }

public Stack();

public Stack(float[] arr);

public Stack(Stack s);

public void Push(float value);

public float Pop();

public float Peek();

public override bool Equals(object obj);

public void Print();

public void Input();

}

}

**Финальная реализация класса**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections;

namespace My {

public class Stack {

private class Node {

public float data;

public Node next;

}

private Node begin;

Node Begin { get; set; }

public Stack() { }

public Stack(float[] arr) {

foreach (var item in arr) {

Node temp = new Node();

temp.data = item;

temp.next = begin;

begin = temp;

}

}

public Stack(Stack s) {

Node snode = s.begin;

while (snode != null) {

Node temp = new Node();

temp.data = snode.data;

temp.next = begin;

begin = temp;

snode = snode.next;

}

}

public void Push(float value) {

Node temp = new Node();

temp.data = value;

temp.next = begin;

begin = temp;

}

public float Pop() {

if (begin == null)

throw new NullReferenceException();

float res = begin.data;

begin = begin.next;

return res;

}

public float Peek() {

if (begin == null)

throw new NullReferenceException();

return begin.data;

}

public override bool Equals(object obj) {

if (obj == null || GetType() != obj.GetType())

return false;

Stack s = (Stack)obj;

Node tnode = begin;

Node snode = s.begin;

while (tnode != null && snode != null) {

if (tnode.data != snode.data)

return false;

tnode = tnode.next;

snode = snode.next;

}

if (tnode != null || snode != null)

return false;

return true;

}

public void Print() {

Node tnode = begin;

while (tnode != null) {

Console.WriteLine("{0} ", tnode.data);

tnode = tnode.next;

}

}

public void Input() {

begin = null;

int N = int.Parse(Console.ReadLine());

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

{

Node temp = new Node();

temp.data = float.Parse(Console.ReadLine());

temp.next = begin;

begin = temp;

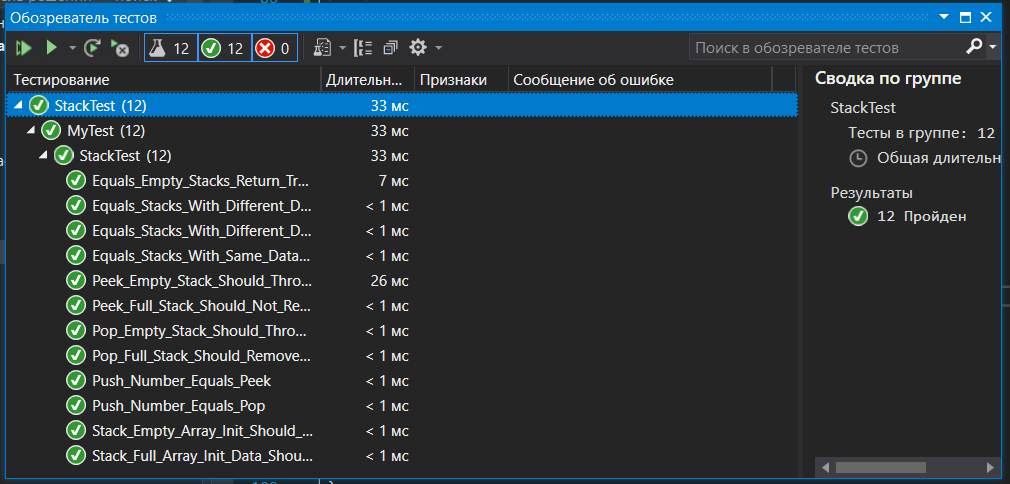
}

}

}

}

**Скрины «зеленого» прохождения тестов**

****

**Выводы по работе**

В результате выполнения лабораторной работы были изучены особенности разработки классов в среде Visual Studio 2019 на языке C# с использованием техники TDD (Test Driven Development). Был реализован набор тестов для проверки стека а также класс Stack для их прохождения.