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 C#»
Вариант № 39

Выполнил ст. гр. ІТШІ-19-2:
Меденицкий Алексей

Принял:
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 с 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
    {

       public class Node {
            public float Data { get; set; }
            public Node Next { get; set; }
        }

        public Node Begin { get; set; }

        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

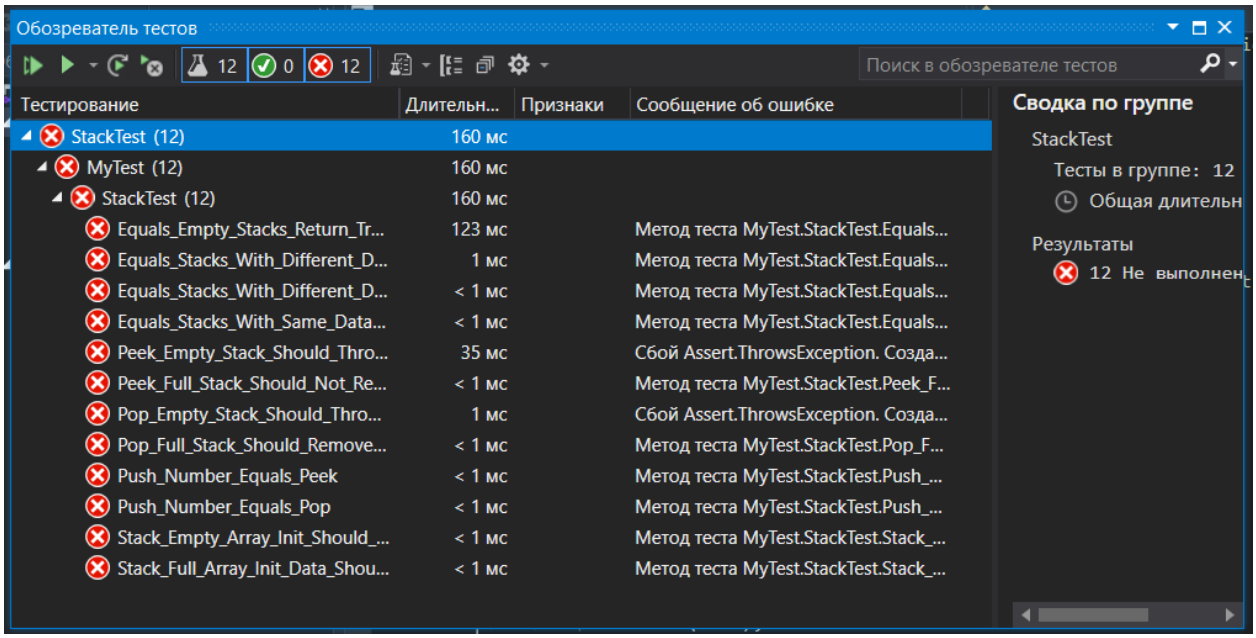
```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using My;

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 {

        public class Node {
            public float Data { get; set; }
            public Node Next { get; set; }
        }

        public 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);
```

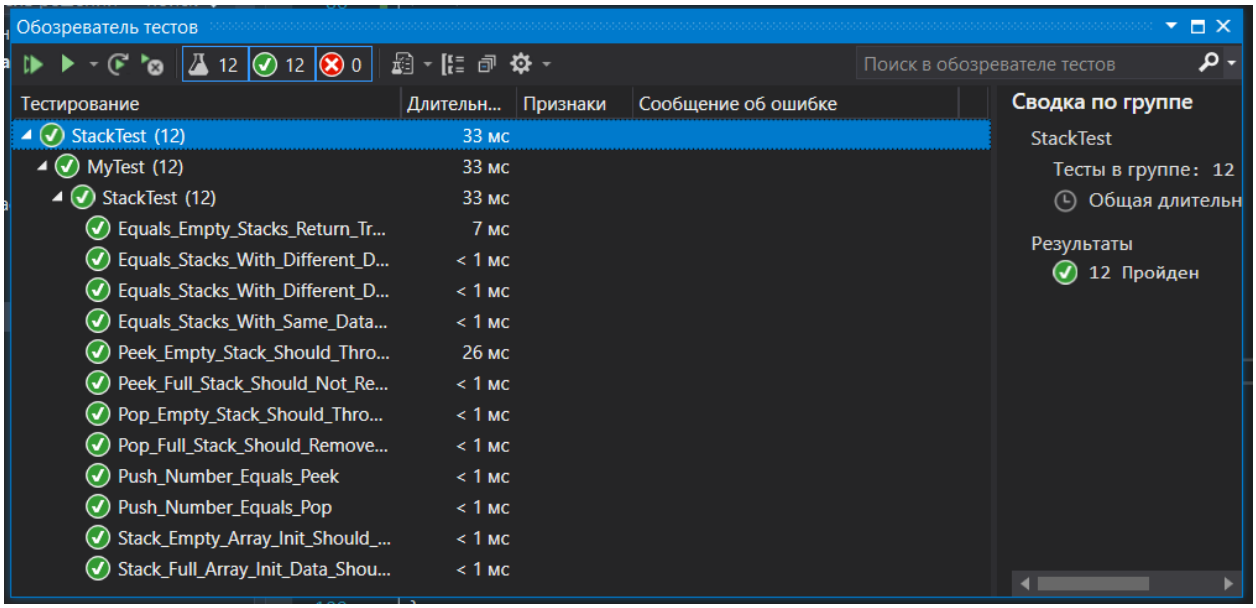
```
        public void Print();
        public void Input();

    }
}
```

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 {

        public class Node {
            public float Data { get; set; }
            public Node Next { get; set; }
        }

        public 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;
            }
        }

        static void Main(string[] args)
        {
            float[] arr = { 1, 2, 3 };
            Stack s = new Stack(arr);
            s.Print();
            s.Input();
            s.Print();
            Console.Read();
        }

    }
}
```

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.