Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа № 10
по дисциплине «Современные технологии программирования»

Выполнил:
студент группы ИП-712
Алексеев Степан
Владимирович
ФИО студента

Работу проверил:
ассистент кафедры Агалаков А.А.
ФИО преподавателя

Новосибирск 2020 г.

Оглавление

ЗАДАНИЕ

1. В соответствии с приведенной ниже спецификацией реализовать параметризованный абстрактный тип данных «память», для хранения одного числа – объекта типа T, используя шаблон классов C++.

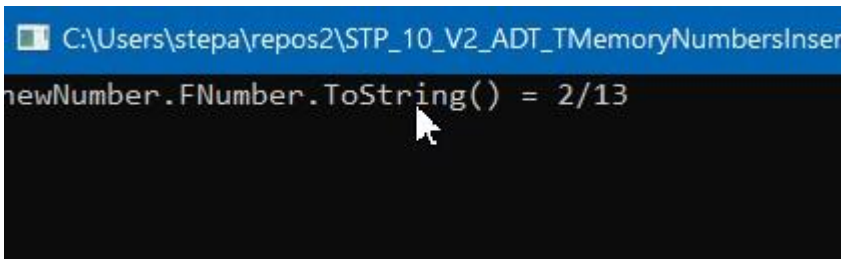
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.

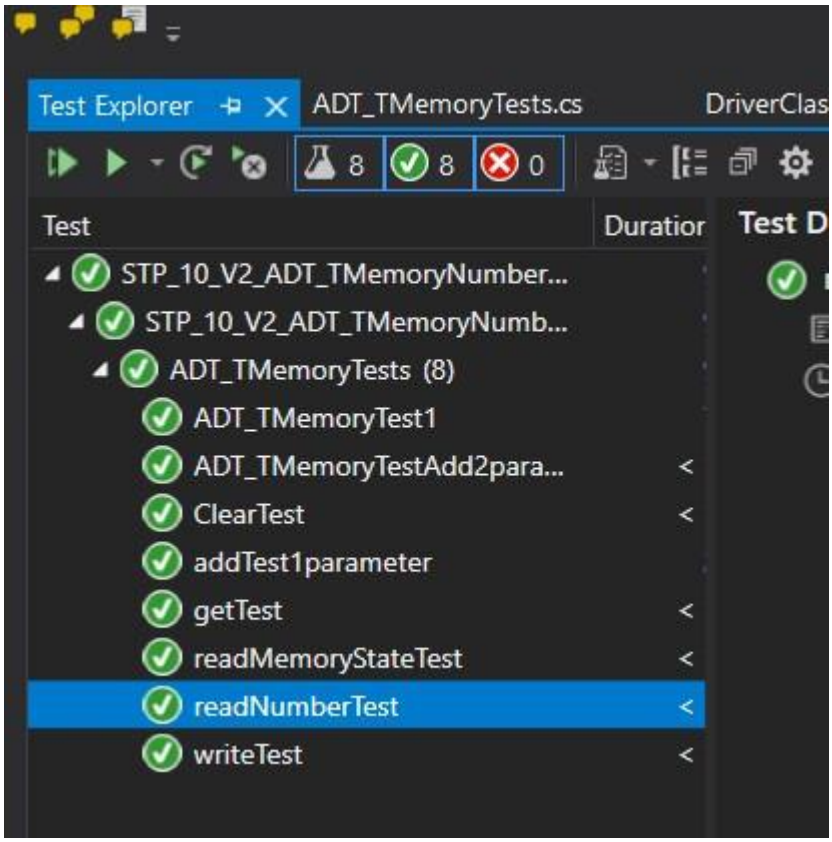
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

ТЕСТОВЫЕ НАБОРЫ ДАННЫХ

```
ADT_TMemory<TFrac> newNumber = new
ADT_TMemory<TFrac>(new TFrac(1, 13));
TFrac tf = newNumber.readNumber();
Assert.AreEqual(tf.ToString(), "1/13");
```

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ





ВЫВОД

Научился работать с обобщениями(дженериками или шаблонами) на языке C#. Научился использовать интерфейсы для классов, в т.ч. параметризованные интерфейсы. На опыте опробовал использование шаблонов классов и убедился в их работоспособности.

ПРИЛОЖЕНИЕ

Листинг 1. ADT_TMemory.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace STP_10_V2_ADT_TMemoryNumbersInsertedLikeFiles
{
    public class ADT_TMemory<T> where T :
InterfaceForNumbers<T>, new()
    {
        public T FNumber;
        string FState;//Memory state

        public ADT_TMemory()
        {
            //ADT_TMemory<TFrac> newNumber = new
ADT_TMemory<TFrac>();
            //newNumber.FNumber = new TFrac();
            FState = "_Off";
        }
        public ADT_TMemory(T t)
        {
            if (t == null) throw new NullPointerException();
            FNumber = t;
            FState = "_On";
        }
        public void write(T e)
        {
            if (e == null) throw new NullPointerException();
            FNumber = e;
            FState = "_On";
        }
        public T get()
        {
            FState = "_On";
            // T t = new T();
            return FNumber;
        }
        public void add(T e)
        {
            if (e == null) throw new NullPointerException();
            FNumber = FNumber.add(e);
            FState = "_On";
        }
        public T add(T a, T b)
        {
```

```
            if (a == null || a == null) throw new
NullPointerException();
            FNumber = FNumber.add(a, b);
            FState = "_On";
            return FNumber;
        }
        public void Clear()
        {
            FNumber = new T();
            FState = "_Off";
        }
        public string readMemoryState()
        {
            return FState;
        }
        public T readNumber()
        {
            return FNumber;
        }
        public class NullPointerException : Exception
        {
            public NullPointerException()
            {
                Console.WriteLine("wrong input");
            }
        }
    }
}}
```

Листинг 2. InterfaceForNumbers.cs

```
//using STP_04_ADT_TFrac;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace STP_10_V2_ADT_TMemoryNumbersInsertedLikeFiles
{
    public interface InterfaceForNumbers<T> where T : new()
    {
        T add(T e);
        T add(T a, T b);
    }
}
```

Листинг 3. DriverClass_STP_10_V2_.cs

```
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace STP_10_V2_ADT_TMemoryNumbersInsertedLikeFiles
{
    class DriverClass_STP_10_V2_
    {
        static void Main(string[] args)
        {
            ADT_TMemory<TFrac> newNumber = new
ADT_TMemory<TFrac>(new TFrac(1, 13));
            ADT_TMemory<TFrac> newNumber2 = new
ADT_TMemory<TFrac>(new TFrac(1, 13));
           // TFrac tf = newNumber.add(newNumber.FNumber,
newNumber2.FNumber);
            newNumber.FNumber = newNumber.add(newNumber.FNumber,
newNumber2.FNumber);
            Console.WriteLine("newNumber.FNumber.ToString() = "
+ newNumber.FNumber.ToString());
           // Console.WriteLine("hello");
            Console.ReadLine();
        }
    }
}
```

Листинг 4. ADT_TmemoryTests.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using STP_10_V2_ADT_TMemoryNumbersInsertedLikeFiles;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using STP_10_V2_ADT_TMemoryNumbersInsertedLikeFiles;
namespace STP_10_V2_ADT_TMemoryNumbersInsertedLikeFiles.Tests
{
    [TestClass()]
    public class ADT_TMemoryTests
    {
        [TestMethod()]
        public void ADT_TMemoryTestAdd2parameters()
        {
            ADT_TMemory<TFrac> newNumber = new
ADT_TMemory<TFrac>(new TFrac(1, 13));
            ADT_TMemory<TFrac> newNumber2 = new
ADT_TMemory<TFrac>(new TFrac(1, 13));
            // TFrac tf = newNumber.add(newNumber.FNumber,
newNumber2.FNumber);
```

```
            newNumber.FNumber = newNumber.add(newNumber.FNumber,
newNumber2.FNumber);
            // Console.WriteLine("newNumber.FNumber.ToString() =
" + newNumber.FNumber.ToString());
            Assert.AreEqual("2/13",
newNumber.FNumber.ToString());
        }

        [TestMethod()]
        public void ADT_TMemoryTest1()
        {
            ADT_TMemory<TComplex> newNumber = new
ADT_TMemory<TComplex>(new TComplex(1, 13));
            ADT_TMemory<TComplex> newNumber2 = new
ADT_TMemory<TComplex>(new TComplex(1, 13));
            newNumber.FNumber = newNumber.add(newNumber.FNumber,
newNumber2.FNumber);
            Assert.AreEqual("2+i*26",
newNumber.FNumber.ToString());
        }

        [TestMethod()]
        public void writeTest()
        {
            ADT_TMemory<TComplex> newNumber = new
ADT_TMemory<TComplex>(new TComplex(1, 13));
            ADT_TMemory<TComplex> newNumber2 = new
ADT_TMemory<TComplex>(new TComplex(24, -2));
            newNumber.write(newNumber2.FNumber);
            Assert.AreEqual("24-i*2",
newNumber.FNumber.ToString());
        }

        [TestMethod()]
        public void getTest()
        {
            ADT_TMemory<TComplex> newNumber = new
ADT_TMemory<TComplex>(new TComplex(1, -13));
            ADT_TMemory<TComplex> newNumber2 = new
ADT_TMemory<TComplex>(new TComplex());
            newNumber2.FNumber = newNumber.get();
            Assert.AreEqual("1-i*13",
newNumber2.FNumber.ToString());
        }

        [TestMethod()]
        public void addTest1parameter()
        {
            ADT_TMemory<TPNumber> newNumber = new
ADT_TMemory<TPNumber>(new TPNumber("1,0", 16, 5));
```

```
            ADT_TMemory<TPNumber> newNumber2 = new
ADT_TMemory<TPNumber>(new TPNumber("9,0", 16, 5));
            newNumber.add(newNumber2.FNumber);
            Assert.AreEqual("A,00000",
newNumber.FNumber.ToString());
        }


        [TestMethod()]
        public void ClearTest()
        {
            ADT_TMemory<TFrac> newNumber = new
ADT_TMemory<TFrac>(new TFrac(1, 13));
            newNumber.Clear();
            Assert.AreEqual("0", newNumber.FNumber.ToString());
        }

        [TestMethod()]
        public void readMemoryStateTest()
        {
            ADT_TMemory<TFrac> newNumber = new
ADT_TMemory<TFrac>(new TFrac(1, 13));
            string str = newNumber.readMemoryState();
            Assert.AreEqual("_On", str);
        }

        [TestMethod()]
        public void readNumberTest()
        {
            ADT_TMemory<TFrac> newNumber = new
ADT_TMemory<TFrac>(new TFrac(1, 13));
            TFrac tf = newNumber.readNumber();
            Assert.AreEqual(tf.ToString(), "1/13");
        }
    }
}
```