Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

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

Кафедра прикладной математики и кибернетики

Лабораторная работа №11 «Память»

> Выполнил: Студент группы ИП-013 Копытина Т.А. Работу проверил: ассистент кафедры ПМиК Агалаков А.А.

Содержание

1.	Зада	ние	. 3
2.	Исхо	одный код программы	. 4
	2.1.	Код программы	. 4
	2.2.	Код тестов.	. 6
3.	. Результаты модульных тестов		. 8
		од	

1. 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.

Данные

Память (тип ТМетогу, в дальнейшем - память) - это память для хранения «числа» объекта типа Т в поле FNumber, и значения «состояние памяти» в поле FState. Объект память - изменяемый. Он имеет два состояния, обозначаемых значениями: «Включена» (_On), «Выключена» (_Off). Её изменяют операции: Записать (Store), Добавить (Add), Очистить (Clear).

2. Исходный код программы 2.1. Код программы

TMemory.cs

```
using System;
using System.Collections.Generic;
using System.Text;
namespace lab11
    public abstract class TMemory<T> where T : new()
        public T data;
        private bool fState = false;
        public TMemory()
            data = new T();
            fState = false;
        }
        public void WriteMemory(T number)
            data = number;
            fState = true;
        }
        public T Get()
            fState = true;
            return data;
        }
        public void Add(T addComplex)
            data = (dynamic)addComplex + (dynamic)data;
            fState = true;
        }
        public void Clear()
            data = new T();
            fState = false;
        public bool ReadState()
            return fState;
        public T ReadNumber()
            return data;
        }
    }
}
```

```
Memory.cs
using System;
using System.Collections.Generic;
using System.Text;
namespace lab11
{
    public class Memory<T> : TMemory<T> where T : new()
        public Memory() : base()
        {
        }
    }
}
```

2.2.Код тестов

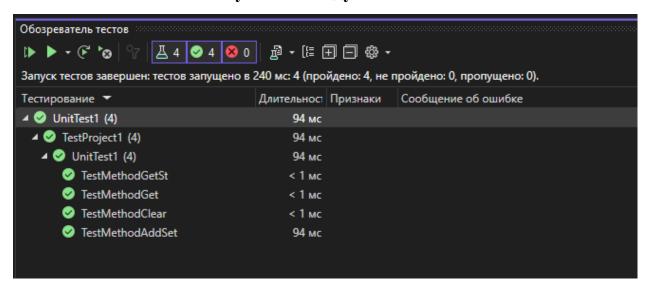
UnitTest1.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using lab11;
using lab5;
namespace TestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethodAddSet()
        {
            lab11.Memory<Frac> f = new lab11.Memory<Frac>();
            f.WriteMemory(new Frac(1, 5));
            var otvet = new Frac(1, 5);
            Assert.AreEqual(otvet.Denominator, f.ReadNumber().Denominator);
            Assert.AreEqual(otvet.Numerator, f.ReadNumber().Numerator);
        }
        [TestMethod]
        public void TestMethodGetSt()
        {
            lab11.Memory<Frac> f = new lab11.Memory<Frac>();
            Assert.IsFalse(f.ReadState());
            f.WriteMemory(new Frac(1, 5));
            Assert.IsTrue(f.ReadState());
        }
```

```
public void TestMethodClear()
        {
            lab11.Memory<Frac> f = new lab11.Memory<Frac>();
            f.WriteMemory(new Frac(5, 6));
            var otvet = new Frac(5, 6);
            Assert.AreEqual(otvet.Denominator, f.ReadNumber().Denominator);
            Assert.AreEqual(otvet.Numerator, f.ReadNumber().Numerator);
            f.Clear();
            otvet = new Frac();
            Assert.AreEqual(otvet.Denominator, f.ReadNumber().Denominator);
            Assert.AreEqual(otvet.Numerator, f.ReadNumber().Numerator);
        }
        [TestMethod]
        public void TestMethodGet()
        {
            lab11.Memory<Frac> f = new lab11.Memory<Frac>();
            f.WriteMemory(new Frac(5, 6));
            var otvet = new Frac(5, 6);
            Assert.AreEqual(otvet.Denominator, f.Get().Denominator);
            Assert.AreEqual(otvet.Numerator, f.Get().Numerator);
        }
   }
}
```

[TestMethod]

3. Результаты модульных тестов



4. Вывод

По итогам данной лабораторной работе были сформированы практические навыки реализации абстрактных типов данных в соответствии с заданной спецификацией с помощью классов С# и их модульного тестирования.