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

Кафедра прикладной математики и кибернетики

Лабораторная работа № 13
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 реализуйте шаблон классов «множество». Для тестирования в качестве параметра шаблона T выберите типы: • int; • TFrac (простая дробь), разработанный вами ранее.

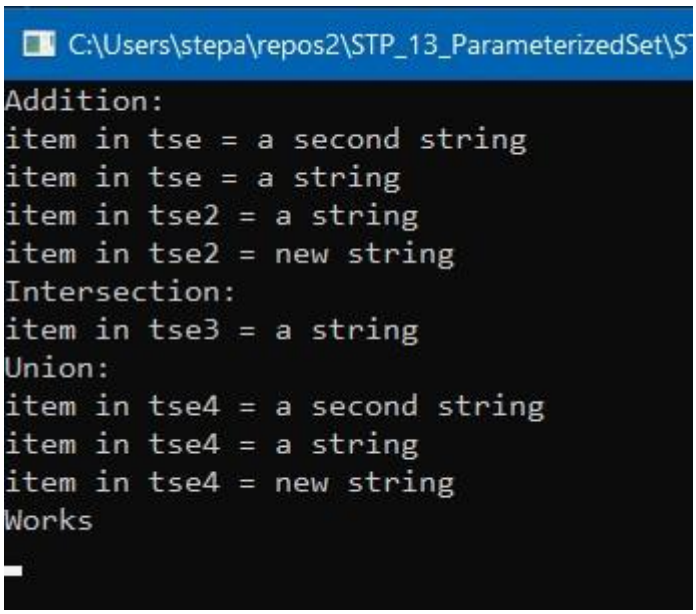
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.

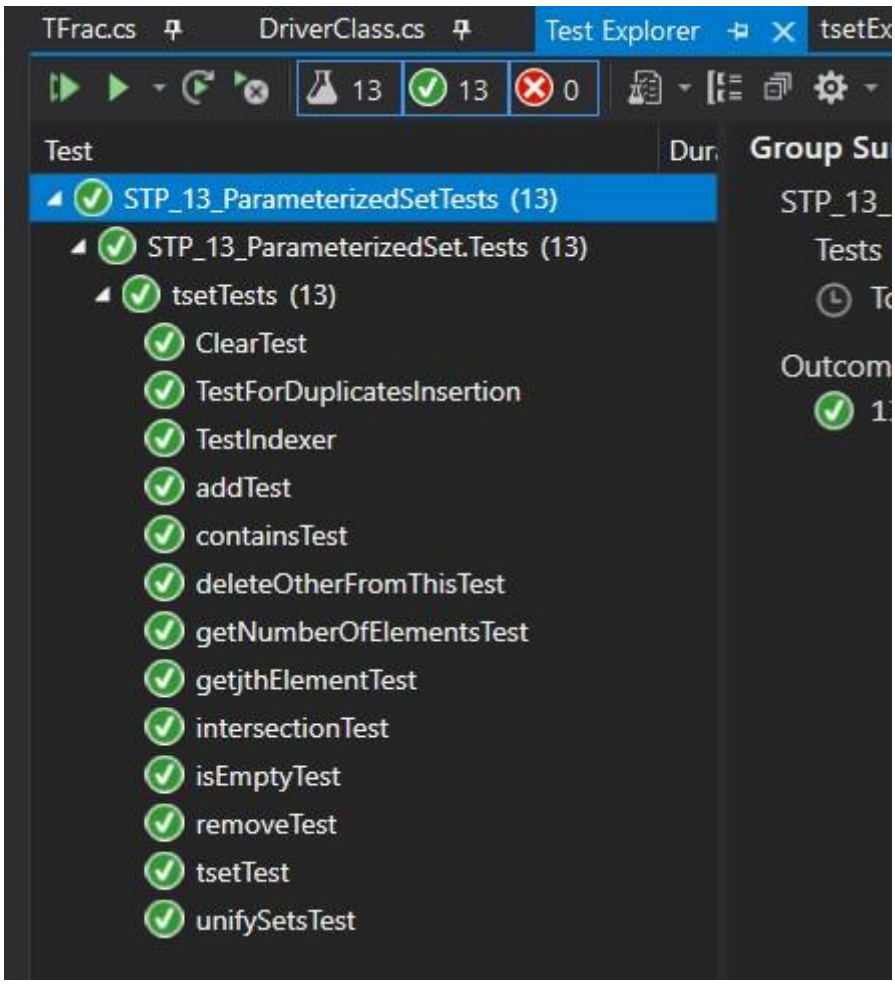
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

ТЕСТОВЫЕ НАБОРЫ ДАННЫХ

```
tset<int> ts = new tset<int>();
        ts.add(15);
        ts.add(27);
        Assert.IsTrue(ts.getNumberOfElements() == 2);
        ts.Clear();
        Assert.IsTrue(ts.getNumberOfElements() == 0);
```

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ





ВЫВОД

Научился создавать и переопределять существующие параметризованные типы. Научился создавать индексирование для целого класса. Освоил новые возможности тестирования, научился добавлять несколько проверок в одном тестовом методе. Начал изучать интерфейсы, позволяющие обходиться с несколькими параметризованными типами как с одним.

ПРИЛОЖЕНИЕ

Листинг 1. tset.cs

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

namespace STP_13_ParameterizedSet
{
    public class tset<T>
    {
        private List<T> items;

        public tset()
        {
            items = new List<T>();
        }
        private tset(IEnumerable<T> items)//IEnumerable
позволяет запускать циклы с объектами
        {
            this.items = new List<T>(items);
        }
        public void Clear()
        {
            items.Clear();
        }
        public void add(T d)
        {
            if(d.GetType().Name == "TFrac")
            {
                // if()
            }
            else
            {

            }
            if (contains(d))
            {
                return;
            }
            items.Add(d);
        }
        public void remove(T d)
        {
            items.Remove(d);
        }
        public bool isEmpty()
        {
```

```
    return items.Count == 0;
}
public bool contains(T d)
{
   return items.Contains(d);
}
public bool Contains(T item) => items.Contains(item);
public tset<T> unifySets(tset<T> other)
{
    var result = new tset<T>(this.items);
    foreach (var item in other.items)
    {
        result.add(item);
    }
    return result;
}
public tset<T> deleteOtherFromThis(tset<T> other)
{
    var result = new tset<T>(this.items);
    foreach (var item in other.items)
    {
        result.remove(item);
    }
    return result;
}
public tset<T> intersection(tset<T> other)
{
    var resultItems = new List<T>();
    foreach (var item in this.items)
    {
        if (other.contains(item))
        {
            resultItems.Add(item);
        }
    }
    var result = new tset<T>(resultItems);
    return result;
}
public int getNumberOfElements()
{
    return items.Count;
}
public T getjthElement(int j)
{
    return items[j];
}
public T this[int i]
{//индексатор
    get
    {
        if (i < 0 || i >= items.Count)
```

```
                {
                    throw new IndexOutOfRangeException();
                }
                return items[i];
            }
        }
    }
}
```

Листинг 2. tsetExtendingSet.cs

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

namespace STP_13_ParameterizedSet
{
   public class tsetExtendingSet<T> : SortedSet<T>
    {

    }
}
```

Листинг 3. tsetTests.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using STP_13_ParameterizedSet;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using STP_13_ParameterizedSet;
namespace STP_13_ParameterizedSet.Tests
{
    [TestClass()]
    public class tsetTests
    {
        [TestMethod()]
        public void tsetTest()
        {
            tset<int> ts = new tset<int>();
            Assert.IsTrue(ts.isEmpty());
        }

        [TestMethod()]
        public void ClearTest()
        {
            tset<int> ts = new tset<int>();
            ts.add(15);
```

```
    ts.add(27);
    Assert.IsTrue(ts.getNumberOfElements() == 2);
    ts.Clear();
    Assert.IsTrue(ts.getNumberOfElements() == 0);
}

[TestMethod()]
public void addTest()
{
    var s = new tset<string>();
    s.add("qwerty");
    Assert.IsTrue(s.getNumberOfElements() == 1);
    Assert.IsTrue(s.contains("qwerty"));

    s.add("500");
    Assert.IsTrue(s.getNumberOfElements() == 2);
    Assert.IsTrue(s.contains("500"));
}

[TestMethod()]
public void removeTest()
{
    var s = new tset<string>();
    s.add("a string");
    Assert.IsTrue(s.Contains("a string"));
    s.remove(("a string"));
    Assert.IsFalse(s.Contains("a string"));
}

[TestMethod]
public void TestForDuplicatesInsertion()
{
    var s = new tset<string>();
    s.add("hi there");
    Assert.AreEqual(1, s.getNumberOfElements());

    s.add("hi there");
    Assert.AreEqual(1, s.getNumberOfElements());
}

[TestMethod()]
public void isEmptyTest()
{
    var s = new tset<string>();
    s.add("hi there");
    s.Clear();
    Assert.IsTrue(s.isEmpty());
}

[TestMethod()]
public void containsTest()
```

```
{
    var s = new tset<string>();
    s.add("hi there");
    Assert.IsTrue(s.contains("hi there"));
    s.Clear();
    Assert.IsFalse(s.contains("hi there"));
}

[TestMethod()]
public void unifySetsTest()
{
    tset<int> ts = new tset<int>();
    ts.add(10);
    ts.add(19);
    tset<int> ts2 = new tset<int>();
    ts2.add(207);
    ts2.add(307);
    var ts3 = ts.unifySets(ts2);
    Assert.IsTrue(ts3.getNumberOfElements() == 4);
    Assert.IsTrue(ts3.contains(207));

}

[TestMethod()]
public void deleteOtherFromThisTest()
{
    tset<int> ts = new tset<int>();
    ts.add(10);
    ts.add(19);
    ts.add(100);
    ts.add(190);
    tset<int> ts2 = new tset<int>();
    ts2.add(19);
    ts2.add(307);
    tset<int> ts3 = ts.deleteOtherFromThis(ts2);
    Assert.AreEqual(ts3.getNumberOfElements(), 3);
    Assert.IsTrue(ts3.contains(190));
    Assert.IsFalse(ts3.contains(19));
}

[TestMethod()]
public void intersectionTest()
{
    tset<int> ts = new tset<int>();
    ts.add(10);
    ts.add(19);
    ts.add(100);
    ts.add(190);
    tset<int> ts2 = new tset<int>();
    ts2.add(19);
    ts2.add(307);
```

```
            ts2.add(100);
            ts2.add(407);
            tset<int> ts3 = ts.intersection(ts2);
            Assert.AreEqual(ts3.getNumberOfElements(), 2);
            Assert.IsTrue(ts3.contains(100));
            Assert.IsFalse(ts3.contains(10));
        }

        [TestMethod()]
        public void getNumberOfElementsTest()
        {
            tset<int> ts2 = new tset<int>();
            ts2.add(19);
            ts2.add(307);
            ts2.add(100);
            Assert.AreEqual(ts2.getNumberOfElements(), 3);
        }

        [TestMethod()]
        public void getjthElementTest()
        {
            tset<int> ts2 = new tset<int>();
            ts2.add(19);
            ts2.add(307);
            ts2.add(100);
            ts2.add(307);
            Assert.AreEqual(ts2.getjthElement(1), 307);
        }
        [TestMethod]
        public void TestIndexer()
        {
            var s = new tset<int>();
            s.add(5);
            s.add(4);
            s.add(8);
            Assert.AreEqual(5, s[0]);
            Assert.AreEqual(4, s[1]);
            Assert.AreEqual(8, s[2]);
            int y;
            s.remove(4);
            Assert.AreEqual(8, s[1]);
            Assert.ThrowsException<IndexOutOfRangeException>(()
=> y = s[5]);
            Assert.ThrowsException<IndexOutOfRangeException>(()
=> _ = s[6]);
        }
    }
}
```

Листинг 4. DriverClass.cs

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

namespace STP_13_ParameterizedSet
{
    class DriverClass
    {
        static void Main(string[] args)
        {
            tsetExtendingSet<string> tse = new
tsetExtendingSet<string>();
            tse.Add("a string");
            tse.Add("a second string");
            Console.WriteLine("Addition:");
            foreach (var item in tse)
            {
                Console.WriteLine("item in tse = " +
item.ToString());
            }
            var tse2 = new tsetExtendingSet<string>();
            tse2.Add("a string");
            tse2.Add("new string");
            foreach (var item in tse2)
            {
                Console.WriteLine("item in tse2 = " +
item.ToString());
            }

            var tse3 = tse.Intersect(tse2);
            Console.WriteLine("Intersection:");
            foreach (var item in tse3)
            {
                Console.WriteLine("item in tse3 = " +
item.ToString());
            }
            var tse4 = tse.Union(tse2);
            Console.WriteLine("Union:");
            foreach (var item in tse4)
            {
                Console.WriteLine("item in tse4 = " +
item.ToString());
            }
```

```
            Console.WriteLine("Works");
            Console.ReadLine();
        }
    }
}
```