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

Кафедра прикладной математики и кибернетики

Лабораторная работа № 11
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 «Процессор», используя шаблон классов C++.

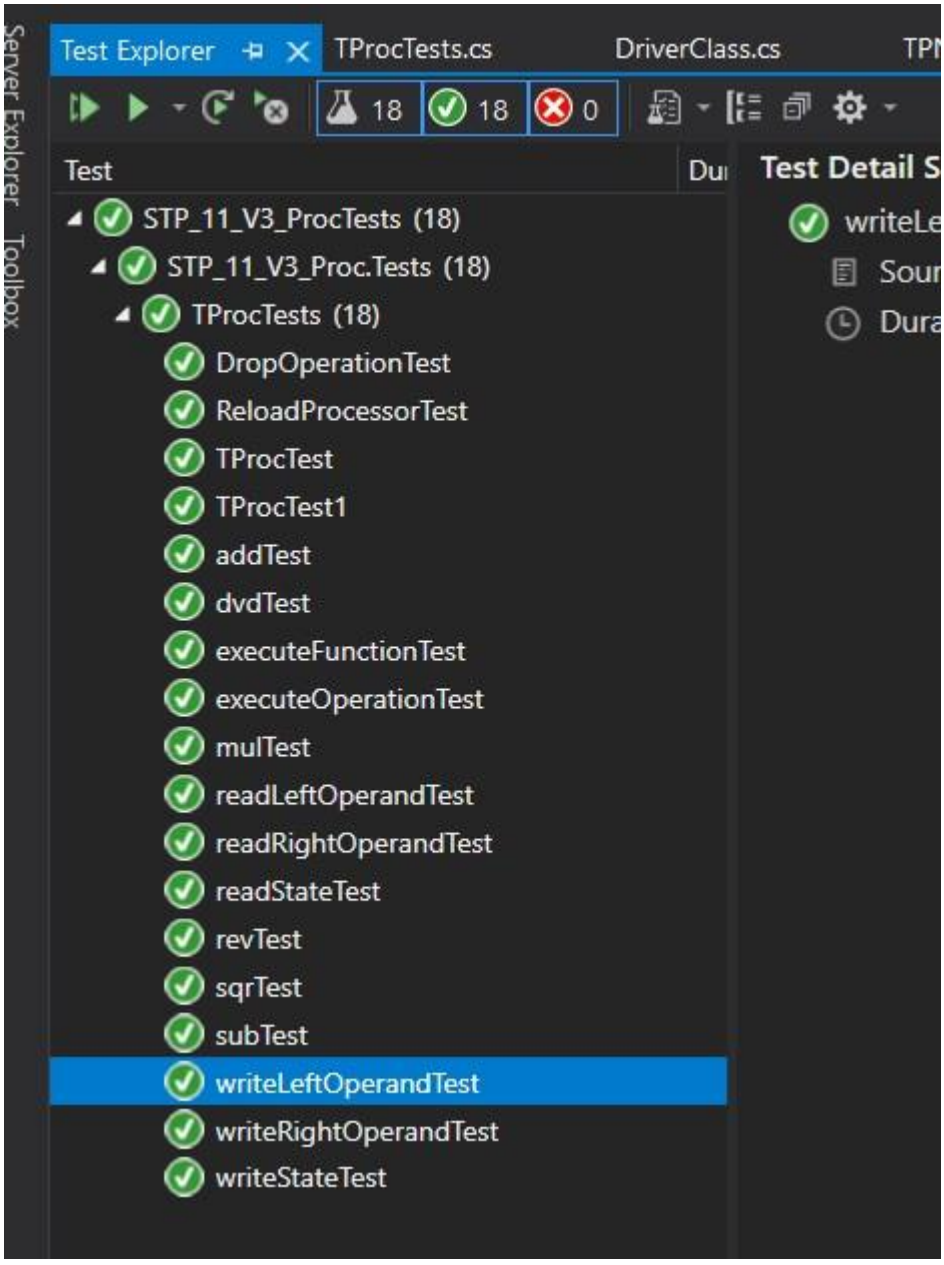
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования.

3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

ТЕСТОВЫЕ НАБОРЫ ДАННЫХ

```
TProc<TComplex> tp = new TProc<TComplex>(new TComplex(1, 7),
new TComplex(1, 6));
        tp.executeFunction("sqr");
        Assert.AreEqual(tp.readLeftOperand().ToString(), "-48+i*14");
```

ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ



ВЫВОД

Углубился в знания о шаблонах классов, о параметризованных типах в языке Сишарп. Научился использовать интерфейсы, параметризованные методы. Реализовал схему работы процессора.

ПРИЛОЖЕНИЕ

Листинг 1. TProc.cs

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

namespace STP_11_V3_Proc
{
    public class TProc<T> where T : InterfaceForNumbers<T>,
new()
    {
      private  string processorState;
        T Lop_Res;//Эти два надо инициализировать значениями по
умолчанию в конструкторе по умолчнаию
        T Rop;
        public TProc()
        {
            Lop_Res = new T();
            Rop = new T();
            processorState = "None";
        }
        public TProc(T a, T b)
        {
            if (a == null || b == null) throw new NullPointer();
            Lop_Res = a;
            Rop = b;
            processorState = "None";
        }
        public void ReloadProcessor()
        {
            Lop_Res = new T();
            Rop = new T();
            processorState = "None";
        }
        public void DropOperation()
        {
            processorState = "None";
        }
        public void executeOperation()
        {
            switch (processorState)
            {
                case "None": break;
                case "add": Lop_Res = Lop_Res.add(Lop_Res, Rop);
break;
                case "sub": Lop_Res = Lop_Res.sub(Lop_Res, Rop);
break;
```

```
                case "mul": Lop_Res = Lop_Res.mul(Lop_Res, Rop);
break;
                case "dvd": Lop_Res = Lop_Res.dvd(Lop_Res, Rop);
break;
                default:
                    throw new WrongInput();
                    break;
            }
        }
        public void executeFunction(string func)
        {
            switch (func)
            {
                case "None": break;
                case "rev": Lop_Res = Lop_Res.rev(Lop_Res);
break;
                case "sqr": Lop_Res = Lop_Res.sqr(Lop_Res);
break;
                default: throw new WrongInput();
                    break;
            }
        }
        public T readLeftOperand()
        {
            return (T)Lop_Res.Clone();
        }
        public void writeLeftOperand(T Operand)
        {
            Lop_Res = (T) Operand.Clone();
        }
        public T readRightOperand()
        {
            return (T)Rop.Clone();
        }
        public void writeRightOperand(T Operand)
        {
            Rop = (T)Operand.Clone();
        }
        public string readState()
        {
            return processorState;
        }
        public void writeState(string newState)
        {
            processorState = newState;
        }

        public T add(T a, T b)
        {
            T t = a.add(a, b);
```

```
        return t;
    }
    public T mul(T a, T b)
    {
        T t = a.mul(a, b);
        return t;
    }
    public T sub(T a, T b)
    {
        T t = a.sub(a, b);

        return t;
    }
    public T dvd(T a, T b)
    {
        T t = a.dvd(a, b);
        return t;
    }
    public T none(T a, T b)
    {
        T t = a.add(a, b);

        return t;
    }
    public T rev(T a)
    {
        T t = a.rev(a);
        return t;
    }
    public T sqr(T a)
    {
        T t = a.sqr(a);

        return t;
    }
    public class WrongInput : Exception
    {
        public WrongInput()
        {
            Console.WriteLine("wrong input");
        }
    }
    public class NullPointer : Exception
    {
        public NullPointer()
        {
            Console.WriteLine("wrong link");
        }
    }

}
```

```
}
```

Листинг 2. TProcTests.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using STP_11_V3_Proc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace STP_11_V3_Proc.Tests
{
    [TestClass()]
    public class TProcTests
    {
        [TestMethod()]
        public void TProcTest()
        {
            TProc<TFrac> tp = new TProc<TFrac>();
            Assert.AreEqual(tp.readLeftOperand().ToString(),
"0");
        }

        [TestMethod()]
        public void TProcTest1()
        {
            TProc<TFrac> tp = new TProc<TFrac>(new TFrac(1, 7),
new TFrac(1, 6));
            Assert.AreEqual(tp.readRightOperand().ToString(),
"1/6");
        }

        [TestMethod()]
        public void ReloadProcessorTest()
        {
            TProc<TFrac> tp = new TProc<TFrac>(new TFrac(1, 7),
new TFrac(1, 6));
            tp.ReloadProcessor();

            Assert.AreEqual(tp.readState(), "None");
        }

        [TestMethod()]
        public void DropOperationTest()
        {
            TProc<TFrac> tp = new TProc<TFrac>(new TFrac(1, 7),
new TFrac(1, 6));
            tp.DropOperation();
```

```
            Assert.AreEqual(tp.readState(), "None");
        }

        [TestMethod()]
        public void executeOperationTest()
        {
            TProc<TFrac> tp = new TProc<TFrac>(new TFrac(1, 7),
new TFrac(1, 6));
            tp.writeState("mul");
            tp.executeOperation();
            Assert.AreEqual(tp.readLeftOperand().ToString(),
"1/42");
        }

        [TestMethod()]
        public void executeFunctionTest()
        {
            TProc<TComplex> tp = new TProc<TComplex>(new
TComplex(1, 7), new TComplex(1, 6));
            tp.executeFunction("sqr");
            Assert.AreEqual(tp.readLeftOperand().ToString(), "-
48+i*14");
        }

        [TestMethod()]
        public void readLeftOperandTest()
        {
            TProc<TComplex> tp = new TProc<TComplex>(new
TComplex(1, 7), new TComplex(1, 6));
            Assert.AreEqual(tp.readLeftOperand().ToString(),
"1+i*7");
        }

        [TestMethod()]
        public void writeLeftOperandTest()
        {
            TProc<TPNumber> tp = new TProc<TPNumber>(new
TPNumber(11, 10, 1), new TPNumber(15, 10, 1));
            TPNumber tpn = new TPNumber("AB,0", 16, 1);
            tp.writeLeftOperand(tpn);

            Assert.AreEqual(tp.readLeftOperand().ToString(),
"AB,0");
        }

        [TestMethod()]
        public void readRightOperandTest()
        {
            TProc<TComplex> tp = new TProc<TComplex>(new
TComplex(1, 7), new TComplex(1, 6));
```

```
            Assert.AreEqual(tp.readRightOperand().ToString(),
"1+i*6");
        }

        [TestMethod()]
        public void writeRightOperandTest()
        {
            TProc<TPNumber> tp = new TProc<TPNumber>(new
TPNumber(11, 10, 1), new TPNumber(15, 10, 1));
            TPNumber tpn = new TPNumber("AB,0", 16, 1);
            tp.writeRightOperand(tpn);

            Assert.AreEqual(tp.readRightOperand().ToString(),
"AB,0");

        }

        [TestMethod()]
        public void readStateTest()
        {
            TProc<TPNumber> tp = new TProc<TPNumber>();
            Assert.AreEqual(tp.readState(), "None");
        }

        [TestMethod()]
        public void writeStateTest()
        {
            TProc<TPNumber> tp = new TProc<TPNumber>();
            tp.writeState("dvd");
            Assert.AreEqual(tp.readState(), "dvd");
        }

        [TestMethod()]
        public void addTest()
        {
            TProc<TFrac> tp = new TProc<TFrac>(new TFrac(1, 7),
new TFrac(1, 6));
            tp.writeState("add");
            tp.executeOperation();
            Assert.AreEqual(tp.readLeftOperand().ToString(),
"13/42");
        }

        [TestMethod()]
        public void mulTest()
        {
            TProc<TComplex> tp = new TProc<TComplex>(new
TComplex(1, 7), new TComplex(1, 6));
            tp.writeState("mul");
            tp.executeOperation();
```

```
            Assert.AreEqual(tp.readLeftOperand().ToString(), "-
41+i*13");
        }

        [TestMethod()]
        public void subTest()
        {
            TProc<TComplex> tp = new TProc<TComplex>(new
TComplex(1, 7), new TComplex(1, 6));
            tp.writeState("sub");
            tp.executeOperation();
            Assert.AreEqual(tp.readLeftOperand().ToString(),
"0+i*1");
        }

        [TestMethod()]
        public void dvdTest()
        {
            TProc<TFrac> tp = new TProc<TFrac>(new TFrac(1, 7),
new TFrac(1, 6));
            tp.writeState("dvd");
            tp.executeOperation();
            Assert.AreEqual(tp.readLeftOperand().ToString(),
"6/7");
        }



        [TestMethod()]
        public void revTest()
        {
            TProc<TFrac> tp = new TProc<TFrac>(new TFrac(2, 7),
new TFrac(1, 6));
            tp.executeFunction("rev");
            Assert.AreEqual(tp.readLeftOperand().ToString(),
"7/2");
        }

        [TestMethod()]
        public void sqrTest()
        {
            TProc<TFrac> tp = new TProc<TFrac>(new TFrac(2, 7),
new TFrac(1, 6));
            tp.executeFunction("sqr");
            Assert.AreEqual(tp.readLeftOperand().ToString(),
"4/49");
        }
    }
} }
```

Листинг 3. InterfaceForNumbers.cs

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

namespace STP_11_V3_Proc
{
    public interface InterfaceForNumbers<T> where T : new()
    {
        T add(T a, T b);
        T mul(T a, T b);
        T sub(T a, T b);
        T dvd(T a, T b);
        T rev(T a);
        T sqr(T a);
        object Clone();
    }
}
```