Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации Сибирский Государственный Университет
Телекоммуникаций и Информатики СибГУТИ

Кафедра прикладной математики и кибернетики

Лабораторная работа №5

по дисциплине “Современные технологии программирования”

Класс простая дробь

Выполнил:
Студент группы ИП-916
Меньщиков Д.А.

Работу проверил:
Агалаков А. А.

Новосибирск, 2022

Задание:

1. Реализовать абстрактный тип данных «простая дробь», используя класс C++ в соответствии с приведенной ниже спецификацией.
2. Протестировать каждую операцию, определенную на типе данных, используя средства модульного тестирования Visual Studio. Тестирование осуществляйте по критерию команд C0 .
3. Если необходимо, предусмотрите возбуждение исключительных ситуаций.

Выполнение лабораторной работы:

Код программы:

```
namespace lab5
{
    public class SimpleFractionClass
    {
        private int _nominator = 0;
        private int _denominator = 0;

        private void reduce()
        {
            var GCD = gcd(_nominator, _denominator);

            if (GCD != 1)
            {
                _nominator /= GCD;
                _denominator /= GCD;
            }

        }

        private static int gcd(int a, int b)
        {
            while (b != 0)
            {
                int temp = b;
                b = a % b;
                a = temp;
            }
            return a;
        }

        private static int lcm(int a, int b)
        {
            return (a / gcd(a, b)) * b;
        }

        public SimpleFractionClass(int numerator, int denominator)
        {
            this._nominator = numerator;
            this._denominator = denominator;

            reduce();
        }
```

```
49
50          public SimpleFractionClass(string fractionString)
51          {
52              int delimeterPosition = fractionString.IndexOf('/');
53
54              if (delimeterPosition < 0)
55              {
56                  throw new Exception($"Invalid string"); ;
57              }
58
59              _nominator = int.Parse(fractionString.Substring(0, delimeterPosition));
60              _denominator = int.Parse(fractionString.Substring(delimeterPosition + 1));
61          }
62
63          public SimpleFractionClass(SimpleFractionClass x)
64          {
65              _nominator = x._nominator;
66              _denominator = x._denominator;
67          }
68
69          public void Show()
70          {
71              Console.WriteLine($"Nominator {_nominator}");
72              Console.WriteLine($"Denominator {_denominator}");
73          }
74
75          public static SimpleFractionClass operator +(SimpleFractionClass a, SimpleFractionClass b)
76          {
77              var unionDenominator = lcm(a._denominator, b._denominator);
78
79              var firstNumber = a._nominator * unionDenominator / a._denominator;
80              var secondNumber = b._nominator * unionDenominator / b._denominator;
81
82              return new(firstNumber + secondNumber, unionDenominator);
83          }
```

```
84
85          public static SimpleFractionClass operator -(SimpleFractionClass a, SimpleFractionClass b)
86          {
87              var unionDenominator = lcm(a._denominator, b._denominator);
88
89              var firstNumber = a._nominator * unionDenominator / a._denominator;
90              var secondNumber = b._nominator * unionDenominator / b._denominator;
91
92              return new(firstNumber - secondNumber, unionDenominator);
93          }
94
95          public static SimpleFractionClass operator *(SimpleFractionClass a, SimpleFractionClass b)
96          {
97              return new(a._nominator * b._nominator, a._denominator * b._denominator);
98          }
99
100         public static SimpleFractionClass operator /(SimpleFractionClass a, SimpleFractionClass b)
101         {
102             var nominator = a._nominator * b._denominator;
103             var denominator = a._denominator * b._nominator;
104
105             if (denominator < 0)
106             {
107                 nominator *= -1;
108                 denominator *= -1;
109             }
110
111             return new(a._nominator * b._denominator, a._denominator * b._nominator);
112         }
113
114         public static SimpleFractionClass Pow(SimpleFractionClass a, int n = 2)
115         {
116             return new((int)Math.Pow(a._nominator, n), (int)Math.Pow(a._denominator, n));
117         }
118
119         public static SimpleFractionClass Revers(SimpleFractionClass a)
120         {
121             return new(a._denominator, a._nominator);
122         }
123
```

```
        public static SimpleFractionClass Minus(SimpleFractionClass a)
        {
            SimpleFractionClass z = new(0, 1);
            return new(z - a);
        }

        public static bool operator ==(SimpleFractionClass a, SimpleFractionClass b)
        {
            return (a._nominator == b._nominator && a._denominator == b._denominator);
        }

        public static bool operator !=(SimpleFractionClass a, SimpleFractionClass b)
        {
            return (a._nominator != b._nominator && a._denominator != b._denominator);
        }

        public int GetNominatorInt()
        {
            return _nominator;
        }

        public int GetDenominatorInt()
        {
            return _denominator;
        }

        public string GetNominatorString()
        {
            return _nominator.ToString();
        }
```

```

          public string GetDenominatorString()
          {
              return _denominator.ToString();
          }

          new public string ToString()
          {
              var nominator = GetNominatorString();
              var denominator = GetDenominatorString();

              string sout = nominator + "/" + denominator;

              return sout;
          }
      }
  }
```

Модульные тесты:

```
namespace TestSimpleFractionClass
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestConstructor()
        {
            SimpleFractionClass a = new (1, 5);
        }

        [TestMethod]
        public void TestConstructorString()
        {
            SimpleFractionClass a = new("1/5");
        }

        [TestMethod]
        public void TestOperatorPlus()
        {
            SimpleFractionClass a = new("1/5");
            SimpleFractionClass b = new(1, 3);

            SimpleFractionClass actual = new (a + b);
            SimpleFractionClass expected = new (8, 15);

            Assert.IsTrue(expected == actual);
        }

        [TestMethod]
        public void TestOperatorMinus()
        {
            SimpleFractionClass a = new("1/5");
            SimpleFractionClass b = new(1, 3);

            SimpleFractionClass actual = new(a - b);
            SimpleFractionClass expected = new(2, -15);

            Assert.IsTrue(expected == actual);
        }
```

```
43
44          [TestMethod]
45          public void TestOperatorProduct()
46          {
47              SimpleFractionClass a = new("1/5");
48              SimpleFractionClass b = new(1, 3);
49
50              SimpleFractionClass actual = new(a * b);
51              SimpleFractionClass expected = new(1, 15);
52
53              Assert.IsTrue(expected == actual);
54          }
55
56          [TestMethod]
57          public void TestOperatorDivide()
58          {
59              SimpleFractionClass a = new("1/5");
60              SimpleFractionClass b = new(1, 3);
61
62              SimpleFractionClass actual = new(a / b);
63              SimpleFractionClass expected = new(3, 5);
64
65              Assert.IsTrue(expected == actual);
66          }
67
68          [TestMethod]
69          public void TestPow()
70          {
71              SimpleFractionClass a = new(5, 18);
72
73              SimpleFractionClass actual = SimpleFractionClass.Pow(a);
74              SimpleFractionClass expected = new(25, 324);
75
76              Assert.IsTrue(expected == actual);
77          }
```

```
78
79          [TestMethod]
80          public void TestRevers()
81          {
82              SimpleFractionClass a = new(5, 18);
83
84              SimpleFractionClass actual = SimpleFractionClass.Revers(a);
85              SimpleFractionClass expected = new(18, 5);
86
87              Assert.IsTrue(expected == actual);
88          }
89
90          [TestMethod]
91          public void TestNotEqual()
92          {
93              SimpleFractionClass a = new(5, 18);
94
95              SimpleFractionClass actual = SimpleFractionClass.Revers(a);
96              SimpleFractionClass expected = new(1, 5);
97
98              Assert.IsFalse(expected != actual);
99          }
100     }
101 }
```