**TASK 1**

**Calculator**

namespace Task1

{

public class Calculator : ICalculator

{

public double Multiply(double lhs, double rhs)

{

return lhs \* rhs;

}

public double Divide(double lhs, double rhs)

{

if (rhs == 0)

{

return double.NaN;

}

return lhs / rhs;

}

}

}

**Calc Test ------**

**using Xunit;**

**namespace Task1**

**{**

**public class CalculatorTests**

**{**

**[Fact]**

**public void TestMultiply()**

**{**

**var c = new Calculator();**

**var result = c.Multiply(3, 2);**

**Assert.Equal(result, 6);**

**}**

**[Fact]**

**public void TestDivide()**

**{**

**var c = new Calculator();**

**var result = c.Divide(6, 3);**

**Assert.Equal(result, 2);**

**}**

**[Fact]**

**public void TestDivideByZero()**

**{**

**var c = new Calculator();**

**var result = c.Divide(6, 0);**

**Assert.Equal(result, double.NaN);**

**}**

**}**

**}**

**Icalculator -----**

**namespace Task1**

**{**

**interface ICalculator**

**{**

**double Multiply(double lhs, double rhs);**

**double Divide(double lhs, double rhs);**

**}**

**}**

**PROXY-------------**

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Threading.Tasks;**

**using Xunit;**

**namespace Task1**

**{**

**public class Proxy : ICalculator**

**{**

**enum OperationType**

**{**

**Multiplication, Division**

**}**

**struct Calculation**

**{**

**public OperationType Type;**

**public double Lhs;**

**public double Rhs;**

**public double Result;**

**}**

**private Queue<Calculation> \_queue = new Queue<Calculation>(10);**

**private Calculator \_calculator = new Calculator();**

**public Proxy()**

**{**

**}**

**public double Multiply(double lhs, double rhs)**

**{**

**var result = FromCache(OperationType.Multiplication, lhs, rhs, (a, b) => \_calculator.Multiply(a, b));**

**return result;**

**}**

**public double Divide(double lhs, double rhs)**

**{**

**var result = FromCache(OperationType.Division, lhs, rhs, (a, b) => \_calculator.Divide(a, b));**

**return result;**

**}**

**private double FromCache(OperationType op, double lhs, double rhs, Func<double, double, double> fn)**

**{**

**foreach (var r in \_queue)**

**{**

**if (r.Type == op && (r.Lhs == lhs && r.Rhs == rhs))**

**{**

**return r.Result;**

**}**

**}**

**var result = fn(lhs, rhs);**

**\_queue.Enqueue(new Calculation**

**{**

**Type = op,**

**Lhs = lhs,**

**Rhs = rhs,**

**Result = result,**

**});**

**return result;**

**}**

**[Fact]**

**public void TestProxy()**

**{**

**var result = Multiply(3, 2);**

**var head = \_queue.First();**

**Assert.Equal(OperationType.Multiplication, head.Type);**

**Assert.Equal(3, head.Lhs);**

**Assert.Equal(2, head.Rhs);**

**Assert.Equal(6, head.Result);**

**head.Result = 42;**

**\_queue.Clear();**

**\_queue.Enqueue(head);**

**result = Multiply(3, 2);**

**Assert.Equal(42, result);**

**}**

**}**

**}**