**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П.О.СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Дисциплина «Объектно-ориентированное проектирование и программирование»

ОТЧЕТ

Лабораторная работа № 2

Особенности разработки и верификации в объектно-ориентированном языке

Выполнил студент

Группы ИТД-21

Чайдаков И.М.

Принял преподаватель

Башаримов Ю. С.

Гомель 2024

Особенности разработки и верификации в объектно-ориентированном языке

**Цель работы:** научится работать с перегрузкой параметров и методами верификации приложений в объектно-ориентированном языке *C#*

**Задание:** Необходимо разработать класс представляющий собой определенный интеграл. Выполнить для него перегрузку операции сложения и умножения.

**Ход выполнения задания:**

**1.** Описываем класс представляющий собой определенный интеграл (см. Приложение А).

**2.** Описываем класс представляющий собой пользовательский интерфейс программы (см. Приложение А).

**3.** Запускает файл класса main и видим перед собой консольный интерфейс (Рисунок 1).

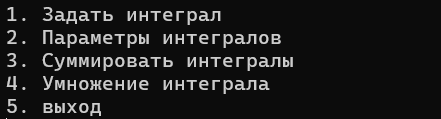


Рисунок 1 — Интерфейс программы.

**4.** Инициализируем интеграл от куба *x* на промежутке от 2 до 7. Даем ему название, интеграл инициализирован (Рисунок 2).

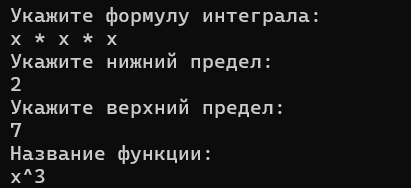


Рисунок 2 — Инициализация интеграла.

**5.** Выполняем проверку параметров созданного интеграла (Рисунок 3).

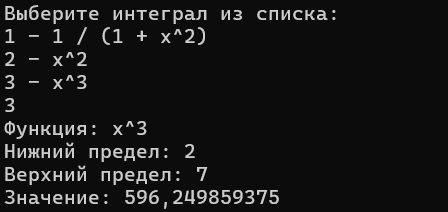


Рисунок 3 — Параметры созданного интеграла.

**6.** Попробуем сложить созданный нами интеграл с интегралов от квадрата *x* (Рисунок 4).

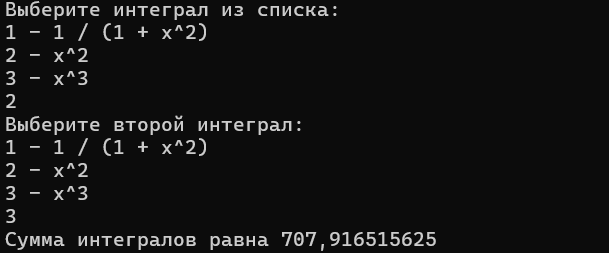


Рисунок 4 — Сложение интегралов.

**5.** Попробуем умножить созданный нами интеграл на 0.72 (Рисунок 5).

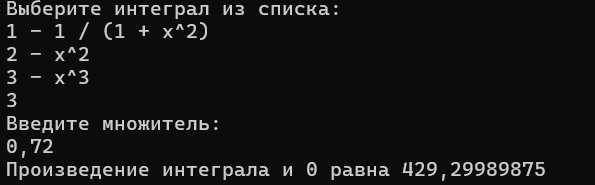


Рисунок 5 — Умножение интеграла.

**Вывод:** Перегрузка операторов позволяет выполнять операции над объектами классов что открывает доступ к большему количеству взаимодействий между объектами классов позволяя решать более сложные задачи.

**ПРИЛОЖЕНИЕ А**

**Код используемых классов**

**Integral**

namespace Library

{

    /// <summary>

    /// Представляет определенный интеграл функции.

    /// </summary>

    public class Integral

    {

        private string \_funcName;

        private Func<double, double> \_function;

        private double \_lowerLimit;

        private double \_upperLimit;

        private double? \_computedValue;

        /// <summary>

        /// Инициализирует новый экземпляр класса.

        /// </summary>

        /// <param name="function">Функция, которую нужно интегрировать.</param>

        /// <param name="lowerLimit">Нижний предел интегрирования.</param>

        /// <param name="upperLimit">Верхний предел интегрирования.</param>

        /// <param name="funcName">Название функции.</param>

        public Integral(Func<double, double> function, double lowerLimit, double upperLimit, string funcName = "unknown function")

        {

            \_funcName = funcName;

            \_function = function;

            \_lowerLimit = lowerLimit;

            \_upperLimit = upperLimit;

            \_computedValue = null;

        }

        /// <summary>

        /// Получает название функции.

        /// </summary>

        /// <returns>Название функции.</returns>

        public string GetName()

        {

            return \_funcName;

        }

        /// <summary>

        /// Вычисляет значение определенного интеграла.

        /// </summary>

        /// <returns>Значение интеграла или <c>null</c>, если интеграл не вычислен.</returns>

        public double? Calculate()

        {

            double stepSize = (\_upperLimit - \_lowerLimit) / 1000;

            double sum = 0.0;

            for (int i = 0; i < 1000; i++)

            {

                double x = \_lowerLimit + (i + 0.5) \* stepSize;

                sum += \_function(x);

            }

            \_computedValue = sum \* stepSize;

            return \_computedValue;

        }

        /// <summary>

        /// Оператор сложения для интегралов.

        /// </summary>

        /// <param name="a">Первый интеграл.</param>

        /// <param name="b">Второй интеграл.</param>

        /// <returns>Сумма значений интегралов.</returns>

        public static double? operator +(Integral a, Integral b)

        {

            return a.\_computedValue + b.\_computedValue;

        }

        /// <summary>

        /// Оператор умножения для интеграла и числа.

        /// </summary>

        /// <param name="integral">Интеграл.</param>

        /// <param name="number">Число.</param>

        /// <returns>Результат умножения значения интеграла на число.</returns>

        public static double? operator \*(Integral integral, double number)

        {

            return integral.\_computedValue \* number;

        }

        /// <summary>

        /// Выводит информацию о функции и ее значении на консоль.

        /// </summary>

        public void Print()

        {

            Console.WriteLine($"Функция: {\_funcName}");

            Console.WriteLine($"Нижний предел: {\_lowerLimit}");

            Console.WriteLine($"Верхний предел: {\_upperLimit}");

            if (\_computedValue != null)

            {

                Console.WriteLine($"Значение: {\_computedValue}\n");

            }

        }

    }

}

**ПРИЛОЖЕНИЕ С**

**Модульные тесты**

**CalcTest\_1:**

namespace Tests;

using Xunit;

using System;

using Library;

public class CalcTests\_1

{

    private readonly Calculate calc = new Calculate();

    private readonly Integral integralA = new Integral(x => x \* x, 2, 7, "x^2");

    private readonly Integral integralB = new Integral(x => x \* x \* x, 2, 7, "x^3");

    private readonly Integral integralC = new Integral(x => 1 / (1 + x \* x), 2, 7, "1 / (1 + x^2)");

    public CalcTests\_1()

    {

        integralA.Solve();

        integralB.Solve();

        integralC.Solve();

    }

    [Fact]

    public void Test\_1()

    {

        double number = -5.13;

        double result = Calculate.Sum(integralA, number);

        Assert.Equal(106.54, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_2()

    {

        double number = 7.28;

        double result = Calculate.Sum(integralB, number);

        Assert.Equal(603.53, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_3()

    {

        double number = 0;

        double result = Calculate.Sum(integralC, number);

        Assert.Equal(0.32, Math.Round(result, 2));

    }

}

**CalcTest\_2:**

namespace Tests;

using Xunit;

using System;

using Library;

public class CalcTests\_2

{

    private readonly Calculate calc = new Calculate();

    private readonly Integral integralA = new Integral(x => x \* x, 2, 7, "x^2");

    private readonly Integral integralB = new Integral(x => x \* x \* x, 2, 7, "x^3");

    private readonly Integral integralC = new Integral(x => 1 / (1 + x \* x), 2, 7, "1 / (1 + x^2)");

    public CalcTests\_2()

    {

        integralA.Solve();

        integralB.Solve();

        integralC.Solve();

    }

    [Fact]

    public void Test\_1()

    {

        double result = Calculate.Sum(integralA, integralB);

        Assert.Equal(707.92, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_2()

    {

        double result = Calculate.Sum(integralA, integralC);

        Assert.Equal(111.99, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_3()

    {

        double result = Calculate.Sum(integralB, integralC);

        Assert.Equal(596.57, Math.Round(result, 2));

    }

}

**CalcTest\_3:**

namespace Tests;

using Xunit;

using System;

using Library;

public class CalcTests\_3

{

    private readonly Calculate calc = new Calculate();

    private readonly Integral integralA = new Integral(x => x \* x, 2, 7, "x^2");

    private readonly Integral integralB = new Integral(x => x \* x \* x, 2, 7, "x^3");

    private readonly Integral integralC = new Integral(x => 1 / (1 + x \* x), 2, 7, "1 / (1 + x^2)");

    public CalcTests\_3()

    {

        integralA.Solve();

        integralB.Solve();

        integralC.Solve();

    }

    [Fact]

    public void Test\_1()

    {

        double number = 0;

        double result = Calculate.Mul(integralA, number);

        Assert.Equal(0, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_2()

    {

        double number = 0.17;

        double result = Calculate.Mul(integralB, number);

        Assert.Equal(101.36, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_3()

    {

        double number = -48.53;

        double result = Calculate.Mul(integralC, number);

        Assert.Equal(-15.61, Math.Round(result, 2));

    }

}

**CalcTest\_4:**

namespace Tests;

using Xunit;

using System;

using Library;

public class CalcTests\_4

{

    private readonly Calculate calc = new Calculate();

    private readonly Integral integralA = new Integral(x => x \* x, 2, 7, "x^2");

    private readonly Integral integralB = new Integral(x => x \* x \* x, 2, 7, "x^3");

    private readonly Integral integralC = new Integral(x => 1 / (1 + x \* x), 2, 7, "1 / (1 + x^2)");

    public CalcTests\_4()

    {

        integralA.Solve();

        integralB.Solve();

        integralC.Solve();

    }

    [Fact]

    public void Test\_1()

    {

        double result = Calculate.Mul(integralA, integralB);

        Assert.Equal(66581.23, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_2()

    {

        double result = Calculate.Mul(integralA, integralC);

        Assert.Equal(35.93, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_3()

    {

        double result = Calculate.Mul(integralB, integralC);

        Assert.Equal(191.84, Math.Round(result, 2));

    }

}

**CalcTest\_5:**

namespace Tests;

using Xunit;

using System;

using Library;

public class CalcTests\_5

{

    private readonly Calculate calc = new Calculate();

    private readonly Integral integralA = new Integral(x => x \* x, 2, 7, "x^2");

    private readonly Integral integralB = new Integral(x => x \* x \* x, 2, 7, "x^3");

    private readonly Integral integralC = new Integral(x => 1 / (1 + x \* x), 2, 7, "1 / (1 + x^2)");

    public CalcTests\_5()

    {

        integralA.Solve();

        integralB.Solve();

        integralC.Solve();

    }

    [Fact]

    public void Test\_1()

    {

        double result = Calculate.Sum(integralA, double.PositiveInfinity);

        Assert.Equal(double.PositiveInfinity, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_2()

    {

        double result = Calculate.Mul(integralA, double.NegativeInfinity);

        Assert.Equal(double.NegativeInfinity, Math.Round(result, 2));

    }

    [Fact]

    public void Test\_3()

    {

        double number = -1;

        double result = Calculate.Mul(integralB, number);

        Assert.Equal(-596.25, Math.Round(result, 2));

    }

}

**ПРИЛОЖЕНИЕ B**

**Код класса *main***

using Library;

using NCalc;

class Program

{

    static void Main()

    {

        string input;

        string name;

        int[] userChoice = new int[2];

        int count;

        double upperLimit;

        double lowerLimit;

        double number;

        Func<double, double> func;

        double? result;

        List<Integral> integrals = new List<Integral>();

        Integral integralA = new Integral(x => 1 / (1 + x \* x), 2, 7, "1 / (1 + x^2)");

        Integral integralB = new Integral(x => x \* x, 2, 7, "x^2");

        integralA.Calculate();

        integralB.Calculate();

        integrals.Add(integralA);

        integrals.Add(integralB);

        bool endIsNear = false;

        while (!endIsNear)

        {

            Console.WriteLine("1. Задать интеграл");

            Console.WriteLine("2. Параметры интегралов");

            Console.WriteLine("3. Суммировать интегралы");

            Console.WriteLine("4. Умножение интеграла");

            Console.WriteLine("5. выход");

            input = Console.ReadLine();

            if (input != null)

            {

                userChoice[0] = int.Parse(input);

                switch (userChoice[0])

                {

                    case 1:

                        {

                            Console.WriteLine("Укажите формулу интеграла: ");

                            input = Console.ReadLine();

                            func = (x) =>

                            {

                                var expr = new Expression(input);

                                expr.Parameters["x"] = x;

                                return (double)expr.Evaluate();

                            };

                            Console.WriteLine("Укажите нижний предел: ");

                            lowerLimit = Convert.ToDouble(Console.ReadLine());

                            Console.WriteLine("Укажите верхний предел: ");

                            upperLimit = Convert.ToDouble(Console.ReadLine());

                            if(upperLimit < lowerLimit)

                            {

                                Console.WriteLine("Error. Wrong data, try again.");

                                break;

                            }

                            Console.WriteLine("Название функции: ");

                            name = Console.ReadLine();

                            integrals.Add(new Integral(func, lowerLimit, upperLimit, name));

                            integrals[integrals.Count-1].Calculate();

                            break;

                        }

                    case 2:

                        {

                            Console.WriteLine("Выберите интеграл из списка:");

                            for (count = 1; count <= integrals.Count; count++)

                            {

                                Console.WriteLine($"{count} - {integrals[count-1].GetName()}");

                            }

                            userChoice[0] = int.Parse(Console.ReadLine());

                            integrals[userChoice[0]-1].Print();

                            break;

                        }

                    case 3:

                        {

                            Console.WriteLine("Выберите интеграл из списка:");

                            for (count = 1; count <= integrals.Count; count++)

                            {

                                Console.WriteLine($"{count} - {integrals[count-1].GetName()}");

                            }

                            userChoice[0] = int.Parse(Console.ReadLine());

                            if(userChoice[0] > integrals.Count)

                            {

                                Console.WriteLine("Error. Wrong input, try again.");

                                break;

                            }

                            Console.WriteLine("Выберите второй интеграл:");

                            for (count = 1; count <= integrals.Count; count++)

                            {

                                Console.WriteLine($"{count} - {integrals[count-1].GetName()}");

                            }

                            userChoice[1] = int.Parse(Console.ReadLine());

                            if(userChoice[1] > integrals.Count)

                            {

                                Console.WriteLine("Error. Wrong input, try again.");

                                break;

                            }

                            result = integrals[userChoice[0]-1] + integrals[userChoice[1]-1];

                            Console.WriteLine($"Сумма интегралов равна {result}");

                            break;

                        }

                    case 4:

                        {

                            Console.WriteLine("Выберите интеграл из списка:");

                            for (count = 1; count <= integrals.Count; count++)

                            {

                                Console.WriteLine($"{count} - {integrals[count-1].GetName()}");

                            }

                            input = Console.ReadLine();

                            userChoice[0] = int.Parse(input);

                            if(userChoice[0] > integrals.Count)

                            {

                                Console.WriteLine("Error. Wrong input, try again.");

                                break;

                            }

                            Console.WriteLine("Введите множитель: ");

                            input = Console.ReadLine();

                            if (double.TryParse(input, out number))

                            {

                                result = integrals[userChoice[0]-1] \* number;

                                Console.WriteLine($"Произведение интеграла и {userChoice[1]} равна {result}");

                            }

                            else

                            {

                                Console.WriteLine("Error. wrong input, try again.");

                            }

                            break;

                        }

                    case 5:

                        {

                            endIsNear = true;

                            break;

                        }

                    default:

                        {

                            Console.WriteLine("Error. Wrong input, try again.\n");

                            break;

                        }

                }

            }

            else

            {

                Console.WriteLine("Error. Null input, try again.\n");

            }

        }

    }

}