**Лабораторная работа №12.1**

**Постановка задачи:**

**Задача 1:** Задан некоторый класс согласно индивидуальному заданию. Реализовать для данного класса конструктор с параметрами и без. Предусмотреть ввод данных пользователем с клавиатуры. Создать для полученного класса интерфейс «Площадь», «Периметр» (длину окружности считать за периметр или можно создать отдельный интерфейс) и дополнительные («Объем», если ваша фигура трехмерная, и для поиска сторон, диагоналей и др.), внутри которого определены соответствующие методы..

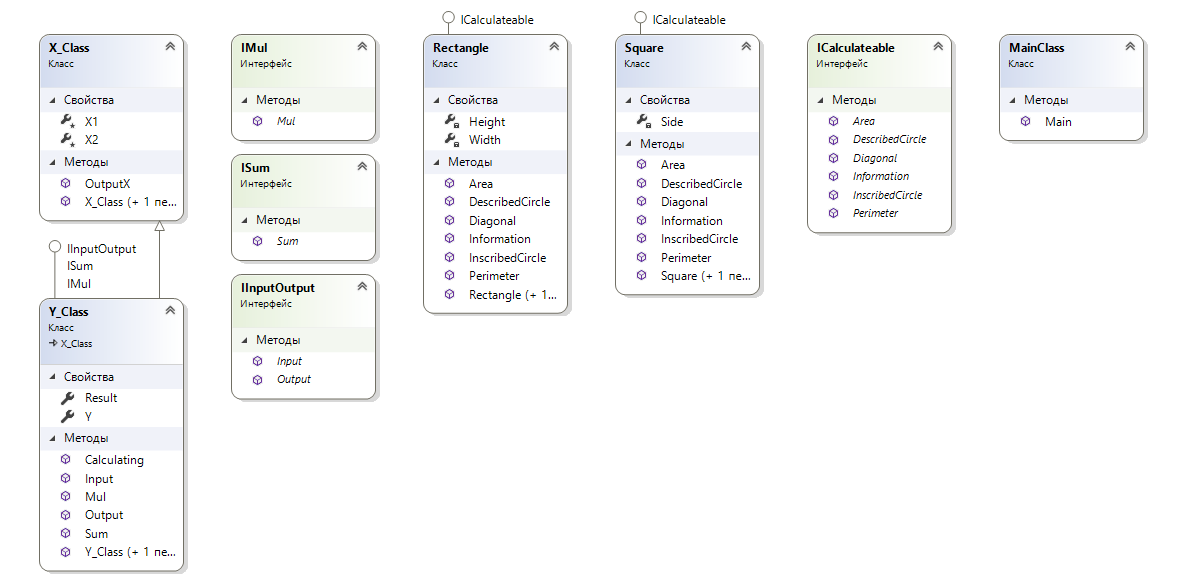
|  |  |  |
| --- | --- | --- |
| 9. | Квадрат | Периметр, площадь, диагональ, радиус окружности вписанной в квадрат, радиус окружности описанной около квадрата |

**Задача 2:** Реализовать иерархию классов Х->Y. В классе Х присутствуют данные x1, x2. В классе Y – данное у. Определить конструкторы с параметрами и без параметров для обоих классов. Создать интерфейс для ввода-вывода данных. В производном классе присутствует метод, определяющий действия из индивидуального задания.

|  |  |
| --- | --- |
| 9. | Значение х1\*х2+у |

**Задача 3:** Для задания номер 2 реализовать множественное наследование интерфейсов, т.е. для выражения (а+b)\*с написать интерфейс ISum и IMult, которые будут наследовать друг друга, а реализовываться уже в одном классе Х с полями х1, х2, у. Здесь же присутствует метод для расчета данных. Создать интерфейс для ввода-вывода данных.

**Диаграмма:**



**Код:**

using System;

namespace Test\_2

{

interface ICalculateable

{

double Perimeter();

double Area();

double Diagonal();

double DescribedCircle();

double InscribedCircle();

void Information();

}

interface IInputOutput

{

void Input(double y, double x1, double x2);

void Output();

}

interface ISum

{

void Sum();

}

interface IMul

{

void Mul();

}

class Square : ICalculateable

{

private double Side { get; set; }

public Square(double side)

{

Side = side;

}

public Square()

{

Side = 5;

}

public double Perimeter()

{

return Side \* 4;

}

public double Area()

{

return Side \* Side;

}

public double Diagonal()

{

return Side \* Math.Sqrt(2);

}

public double DescribedCircle()

{

return Side / Math.Sqrt(2);

}

public double InscribedCircle()

{

return Side / 2;

}

public void Information()

{

Console.WriteLine("Параметры заданного квадрата:");

Console.WriteLine($"1. Периметр: {Perimeter()}\n2. Площадь: {Area()}\n3. Диагональ: {Diagonal()}\n4. Радиус описанной окружности: {DescribedCircle()}\n5. Радиус вписанной окружности: {InscribedCircle()}");

}

}

class Rectangle : ICalculateable

{

private double Width { get; set; }

private double Height { get; set; }

public Rectangle(double width, double height)

{

Width = width;

Height = Width;

}

public Rectangle()

{

Width = 5;

Height = 10;

}

public double Area()

{

return Width \* Height;

}

public double DescribedCircle()

{

return Math.Sqrt(Width \* Width + Height \* Height) / 2;

}

public double Diagonal()

{

return Math.Sqrt((Width \* Width) + (Height \* Height));

}

public double InscribedCircle()

{

if (Width > Height)

{

return Height / 2;

}

else

{

return Width / 2;

}

}

public double Perimeter()

{

return Width \* 2 + Height \* 2;

}

public void Information()

{

Console.WriteLine("Параметры заданного квадрата:");

Console.WriteLine($"1. Периметр: {Perimeter()}\n2. Площадь: {Area()}\n3. Диагональ: {Diagonal()}\n4. Радиус описанной окружности: {DescribedCircle()}\n5. Радиус вписанной окружности: {InscribedCircle()}");

}

}

class X\_Class

{

protected double X1 { get; set; }

protected double X2 { get; set; }

public X\_Class(double x1, double x2)

{

X1 = x1;

X2 = x2;

}

public X\_Class()

{

X1 = 5;

X2 = 10;

}

public void OutputX()

{

Console.WriteLine($"X1: {X1}\nX2: {X2}");

}

}

class Y\_Class : X\_Class, IInputOutput, ISum, IMul

{

public double Y { get; set; }

public double Result { get; set; }

public Y\_Class(double y, double x1, double x2) : base(x1, x2)

{

Result = 1;

Input(y, x1, x2);

}

public Y\_Class()

{

Result = 1;

Y = 8;

X1 = 5;

X2 = 10;

}

public void Input(double y, double x1, double x2)

{

X1 = x1;

X2 = x2;

Y = y;

}

public void Output()

{

OutputX();

Console.WriteLine($"Y: {Y}");

Console.WriteLine($"Ответ: {Calculating()}");

}

public void Mul()

{

Result = X1 \* X2;

}

public void Sum()

{

Result += Y;

}

public double Calculating()

{

Mul();

Sum();

return Result;

}

}

class MainClass

{

public static void Main(string[] args)

{

try

{

Console.WriteLine("----------------");

Console.WriteLine("Длина стороны квадрата: ");

double a = double.Parse(Console.ReadLine());

while (a < 0)

{

Console.WriteLine("Длина стороны квадрата: ");

a = double.Parse(Console.ReadLine());

}

double b;

Square sq = new Square(a);

sq.Information();

Console.WriteLine("----------------");

Console.WriteLine("Длина сторон прямоугольника: ");

a = double.Parse(Console.ReadLine());

b = double.Parse(Console.ReadLine());

while ((a < 0) || (b < 0))

{

Console.WriteLine("Длина сторон прямоугольника: ");

a = double.Parse(Console.ReadLine());

b = double.Parse(Console.ReadLine());

}

Rectangle rect = new Rectangle(a, b);

rect.Information();

Console.WriteLine("----------------");

double x1, x2;

Console.Write("Введите X1:");

x1 = double.Parse(Console.ReadLine());

Console.Write("Введите X2:");

x2 = double.Parse(Console.ReadLine());

Console.Write("Введите Y:");

double y = double.Parse(Console.ReadLine());

X\_Class x = new X\_Class(x1, x2);

Y\_Class \_y = new Y\_Class(y, x1, x2);

\_y.Output();

Console.ReadKey();

}

catch (Exception)

{

Console.WriteLine($"Произошла ошибка");

Square sq = new Square();

sq.Information();

Rectangle rect = new Rectangle();

rect.Information();

Y\_Class y = new Y\_Class();

y.Output();

Console.ReadKey();

}

}

}

}

**Скриншоты:**

