**Лабораторная работа №11.1**

**Постановка задачи:**

**Задача 1:** Задан абстрактный класс «Геометрическая фигура», внутри которого определены абстрактные методы. Производным классом определить класс из варианта индивидуального задания. Реализовать для этих классов конструкторы с параметрами и без и метод для расчёта данных, определенных в индивидуальном задании. Предусмотреть ввод данных пользователем с клавиатуры.

|  |  |  |
| --- | --- | --- |
| 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, определить методы ввода-вывода данных. В производном классе присутствует метод, определяющий действия из индивидуального задания.

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

**Задача 3:** Реализовать иерархию классов. Класс «Фигура» - абстрактный. Первый производный класс из него – класс из задания номер 1. Второй производный класс – родственный первому, т.е. если первый класс равносторонний треугольник, то ему родственный, к примеру, прямоугольный треугольник. В каждом производном классе должны присутствовать: конструктор инициализации, методы ввода-вывода данных, методы из индивидуального задания номер 1.

**Код:**

using System;

namespace Test\_2

{

abstract class Figure

{

public abstract double Perimeter();

public abstract double Area();

public abstract double Diagonal();

public abstract double DescribedCircle();

public abstract double InscribedCircle();

public abstract void Information();

}

class Square : Figure

{

private double Side { get; set; }

public Square(double side)

{

Side = side;

}

public Square()

{

Side = 5;

}

public override double Perimeter()

{

return Side \* 4;

}

public override double Area()

{

return Side \* Side;

}

public override double Diagonal()

{

return Side \* Math.Sqrt(2);

}

public override double DescribedCircle()

{

return Side / Math.Sqrt(2);

}

public override double InscribedCircle()

{

return Side / 2;

}

public override 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 : Figure

{

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 override double Area()

{

return Width \* Height;

}

public override double DescribedCircle()

{

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

}

public override double Diagonal()

{

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

}

public override double InscribedCircle()

{

if (Width > Height)

{

return Height / 2;

}

else

{

return Width / 2;

}

}

public override double Perimeter()

{

return Width \* 2 + Height \* 2;

}

public override 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

{

protected double Y { get; set; }

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

{

Y = y;

}

public Y\_Class()

{

Y = 8;

X1 = 5;

X2 = 10;

}

public void Info()

{

OutputX();

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

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

}

public double Calculating()

{

return X1 \* X2 + Y;

}

}

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.Info();

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.Info();

Console.ReadKey();

}

}

}

}

**Скриншоты:**

