**17 СБОРКИ, БИБЛИОТЕКИ, АТРИБУТЫ, ДИРЕКТИВЫ**

Задание №1. Треугольник (методы ввода сторон, проверки на существование, вычисления периметра, вычисления площади, определения вида треугольника (разносторонний, равнобедренный, равносторонний)); Прямоугольник (методы ввода сторон, вычисления периметра, вычисления площади).

Листинг программы:

namespace FigureLib

{

public abstract class Figure

{

public abstract string Area();

public abstract string Perimeter();

public abstract string ShapeName();

public abstract string GetFigureType();

// public void ShowInfo()

// {

// Console.WriteLine(

// $"Название фигуры: {ShapeName()}\n" +

// $"Площадь: {Area()}\n" +

// $"Периметр: {Perimeter()}\n" +

// $"Тип: {GetFigureType()}"

// );

// Console.WriteLine();

// }

}

public class Triangle : Figure

{

double sideA, sideB, sideC; // Стороны треугольника

// Конструктор

public Triangle(double triangleSideA, double triangleSideB, double triangleSideC)

{

if (!isExist(triangleSideA, triangleSideB, triangleSideC))

{

throw new Exception($"Error");

}

SideA = triangleSideA;

SideB = triangleSideB;

SideC = triangleSideC;

}

// Свойство, проверяем значение на отрицательность.

// Если значение отрицательное меняем его на аналогичное положительное.

public double SideA

{

get { return sideA; }

set { sideA = value < 0 ? -value : value; }

}

public double SideB

{

get { return sideB; }

set { sideB = value < 0 ? -value : value; }

}

public double SideC

{

get { return sideC; }

set { sideC = value < 0 ? -value : value; }

}

public bool isExist(double sideA, double sideB, double sideC)

{

if (sideA > 0 && sideB > 0 && sideC > 0 && (sideA + sideB > sideC && sideA + sideC > sideB && sideB + sideC > sideA))

{

return true;

}

return false;

}

// Метод для вычисления площади треугольника

public override string Area()

{

double semPer = (sideA + sideB + sideC) / 2;

return Math.Sqrt(semPer \* (semPer - sideA) \* (semPer - sideB) \* (semPer - sideC)).ToString();

}

// Метод для вычисления периметра треугольника

public override string Perimeter()

{

return (sideA + sideB + sideC).ToString();

}

public override string GetFigureType()

{

if (sideA == SideB && sideA == sideC) return "равносторонний";

if (sideA == SideB || sideA == sideC || SideB == sideC) return "равнобедренный";

return "треугольник, как треугольник";

}

// Метод возвращающий наименование фигуры

public override string ShapeName()

{

return "Треугольник";

}

}

}

namespace ClassLibrary1

{

public abstract class RectangleFigure

{

public abstract string Area();

public abstract string Perimeter();

public abstract string ShapeName();

public abstract string GetFigureType();

//public void ShowInfo()

//{

// Console.WriteLine(

// $"Название фигуры: {ShapeName()}\n" +

// $"Площадь: {Area()}\n" +

// $"Периметр: {Perimeter()}\n" +

// $"Тип: {GetFigureType()}"

// );

// Console.WriteLine();

//}

}

public class Rectangle : RectangleFigure

{

double width; // Ширина прямоугольника

double height; // Высота прямоугольника

// Конструктор

public Rectangle(double rectangleWidth, double rectangleHeight)

{

Width = rectangleWidth;

Height = rectangleHeight;

}

// Свойство, проверяем значение на отрицательность.

public double Width

{

get { return width; }

set { width = value < 0 ? -value : value; }

}

public double Height

{

get { return height; }

set { height = value < 0 ? -value : value; }

}

//public override bool isExist()

//{

// if (width > 0 && height > 0)

// {

// return true;

// }

// return false;

//}

// Метод для вычисления площади прямоугольника

public override string Area()

{

return (width \* height).ToString();

}

// Метод для вычисления периметра прямоугольника

public override string Perimeter()

{

return (width \* 2 + height \* 2).ToString();

}

public override string GetFigureType()

{

if (width == height) return "Квадрат";

return "Не квадратный прямоугольник";

}

// Метод возвращающий наименование фигуры

public override string ShapeName()

{

return "Прямоугольник";

}

}

}

using FigureLib;

using ClassLibrary1;

using Parking;

try

{

Figure figure1 = new Triangle(10, 10, 10);

Console.WriteLine("Фигура треугольник создана");

//figure1.ShowInfo();

RectangleFigure figure3 = new Rectangle(5, 5);

Console.WriteLine("Фигура прямоугольник создана");

//figure3.ShowInfo();

}

catch (Exception)

{

Console.WriteLine("Error");

}

Console.ReadLine();

Таблица 17.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Фигура треугольник создана  Фигура прямоугольник создана |

Анализ результатов представлен на рисунке 17.1.



Рисунок 17.1 – Результат работы программы

Источник: собственная разработка

Задание №2. Создать свою библиотеку на индивидуальную тему и продемонстрировать ее функциональность.

Листинг программы:

namespace Parking

{

public class ParkingAuto

{

public string nameOfAuto;

public decimal priceOfAuto;

public string typeOfAuto;

public ParkingAuto(string nameOfAuto, decimal priceOfAuto, string typeOfAuto)

{

this.nameOfAuto = nameOfAuto;

this.priceOfAuto = priceOfAuto;

this.typeOfAuto = typeOfAuto;

}

public string BuyAuto(decimal priceOfAuto)

{

if (priceOfAuto < this.priceOfAuto)

{

return "Недостаточно средств";

}

else

{

return $"Вы купили машину {nameOfAuto}";

}

}

public string ShowInfo()

{

return $"Модель - {nameOfAuto}; Цена - {priceOfAuto}; Тип авто - {typeOfAuto};";

}

}

}

using FigureLib;

using ClassLibrary1;

using Parking;

try

{

ParkingAuto auto = new ParkingAuto("Ламборджини", 1000, "Легковой");

Console.WriteLine(auto.ShowInfo());

Console.WriteLine("Введите количество денег для оплаты:");

decimal price = decimal.Parse(Console.ReadLine());

Console.WriteLine(auto.BuyAuto(price));

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

Console.ReadLine();

Таблица 17.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1000 | Вы купили машину Ламборджини |

Анализ результатов представлен на рисунке 17.2.

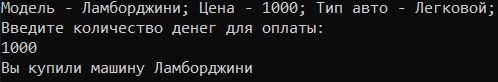


Рисунок 17.2 – Результат работы программы

Источник: собственная разработка