1. **Полиморфизм**

Задание 1. Полиморфизм. Виртуальные методы.

Составить программу с одним родительским классом и двумя потомками. Потомки должны содержать виртуальные функции. Создать виртуальную функцию выдачи результатов расчета методов на экран монитора с указанием названий и полей, и их значений соответствующего объекта. Составить тестирующую программу с выдачей протокола на экран монитора. При этом создать объекты базового и производных типов, используя полиморфный контейнер - массив ссылок базового класса на объекты базового и производных классов (количество объектов >=5).



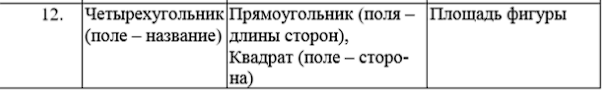


Рисунок 1.1 – Задание

Источник: собственная разработка

Листинг программы:

namespace Task\_1

{

class Program

{

public static void Main(string[] args)

{

try

{

ForSwitch();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static void ForSwitch()

{

try

{

for (int i = 0; i < 5; i++)

{

Console.Write("Введите для вычисления площади прямоугольника-1, квадрата-2: ");

switch (int.Parse(Console.ReadLine()))

{

case 1:

Rectangle();

break;

case 2:

Square();

break;

default:

Console.WriteLine("Нет такого пункта");

break;

}

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static void NameFigure()

{

Quadrilateral q = new Quadrilateral();

q.Input();

Console.WriteLine(q.GetInfo());

}

public static void Rectangle()

{

try

{

NameFigure();

Console.WriteLine("Вычисление площади прямоугольника");

Rectangle r = new Rectangle();

r.Input();

r.SquareRectangle();

Console.WriteLine(r.GetInfo());

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static void Square()

{

try

{

NameFigure();

Console.WriteLine("Вычисление площади квадрата");

Square s = new Square();

s.Input();

s.SquareSquare();

Console.WriteLine(s.GetInfo());

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

}

class Quadrilateral

{

public string name;

public Quadrilateral()

{

this.name = null;

}

public virtual void Input()

{

Console.Write("Введите название фигуры: ");

name = Console.ReadLine();

}

public virtual string GetInfo()

{

return $"Название: {name} ";

}

}

class Rectangle : Quadrilateral

{

public double sizeA;

public double sizeB;

public double S;

public Rectangle() : base()

{

this.sizeA = 0;

this.sizeB = 0;

}

public override void Input()

{

Console.Write("Введите длинну прямоугольника: ");

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

Console.Write("Введите высоту прямоугольника: ");

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

}

public virtual void SquareRectangle()

{

S = sizeA \* sizeB;

}

public override string GetInfo()

{

return $"{base.GetInfo()}Площадь прямоугольника: {S} ";

}

}

class Square : Quadrilateral

{

public double sizeA;

public double S;

public Square() : base()

{

this.sizeA = 0;

}

public override void Input()

{

Console.Write("Введите сторону квадрата: ");

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

}

public virtual void SquareSquare()

{

S = Math.Pow(sizeA, 2);

}

public override string GetInfo()

{

return $"{base.GetInfo()}Площадь квадрата: {S} ";

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1 фигура 3 4  2 квадр 3  2 новая 22  1 прямоугольник 13 2  1 новый прямой 3 4 | 12  9  484  26  12 |

Анализ результатов:

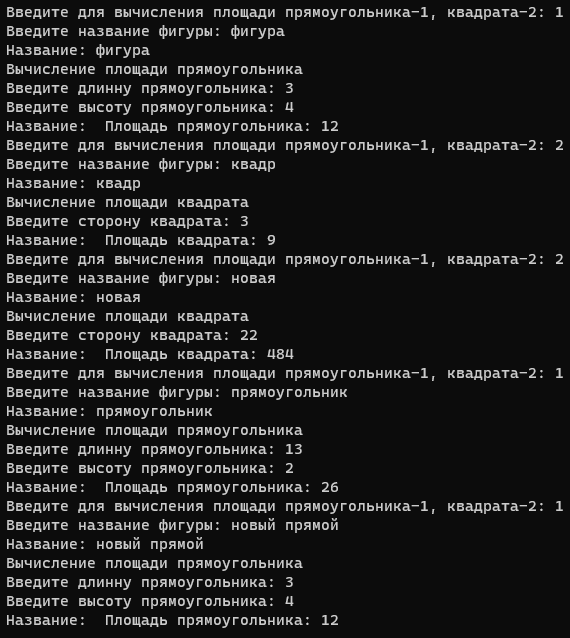


Рисунок 1.2 – Результат работы программы

Источник: собственная разработка

Задание 2. Абстрактные классы (дополнительно).

Составить программу с абстрактным родительским классом и двумя объектами - потомками. Для этого модифицировать задание 2. Составить тестирующую программу с выдачей протокола на экран монитора. В ней нужно реализовать циклический вывод параметров объектов, используя полиморфный контейнер - массив объектов базового класса (количество объектов >=5). Найти максимальную из площадей.

Листинг программы:

using System.Xml.Linq;

namespace Task\_2

{

class Program

{

public static void Main(string[] args)

{

try

{

MaxSquare(ForSwitch());

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

public static Quadrilateral[] ForSwitch()

{

try

{

int k = 5;

Quadrilateral[] quadrilaterals = new Quadrilateral[k];

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

{

Console.Write("Введите для вычисления площади прямоугольника-1, квадрата-2: ");

switch (int.Parse(Console.ReadLine()))

{

case 1:

quadrilaterals[i] = Rectangle();

break;

case 2:

quadrilaterals[i] = Square();

break;

default:

Console.WriteLine("Нет такого пункта");

break;

}

}

return quadrilaterals;

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return null;

}

}

public static void MaxSquare(Quadrilateral[] q)

{

double max = q[0].Square();

for (int i = 0; i < q.Length; i++)

{

if (max < q[i].Square())

{

max = q[i].Square();

}

}

Console.WriteLine($"Максимальная площадь: {max}");

}

public static string NameFigure()

{

Console.Write("Введите название фигуры: ");

return Console.ReadLine();

}

public static Rectangle Rectangle()

{

try

{

Console.WriteLine("Вычисление площади прямоугольника");

string name = NameFigure();

Console.Write("Введите длинну прямоугольника: ");

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

Console.Write("Введите высоту прямоугольника: ");

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

Rectangle r = new Rectangle(name, sizeA, sizeB);

r.Square();

Console.WriteLine(r.GetInfo());

return r;

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return null;

}

}

public static Squareq Square()

{

try

{

Console.WriteLine("Вычисление площади квадрата");

string name = NameFigure();

Console.Write("Введите сторону квадрата: ");

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

Squareq s = new Squareq(name, sizeA);

Console.WriteLine(s.GetInfo());

return s;

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

return null;

}

}

}

abstract class Quadrilateral

{

public string name;

public Quadrilateral(string name)

{

this.name = name;

}

public abstract double Square();

public virtual string GetInfo()

{

return $"Название: {name} ";

}

}

class Rectangle : Quadrilateral

{

public double sizeA;

public double sizeB;

public Rectangle(string name, double sizeA, double sizeB) : base(name)

{

this.sizeA = sizeA;

this.sizeB = sizeB;

}

public override double Square()

{

return sizeA \* sizeB;

}

public override string GetInfo()

{

return $"{base.GetInfo()}Площадь прямоугольника: {Square()} ";

}

}

class Squareq : Quadrilateral

{

public double sizeA;

public Squareq(string name, double sizeA) : base(name)

{

this.sizeA = sizeA;

}

public override double Square()

{

return Math.Pow(sizeA, 2);

}

public override string GetInfo()

{

return $"{base.GetInfo()}Площадь квадрата: {Square()} ";

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1 прям1 45 65  1 прям2 33 2  2 квадр1 102  2 квадр2 2  1 прям3 3 6 | 2925  66  10404  4  18  10404 |

Анализ результатов:

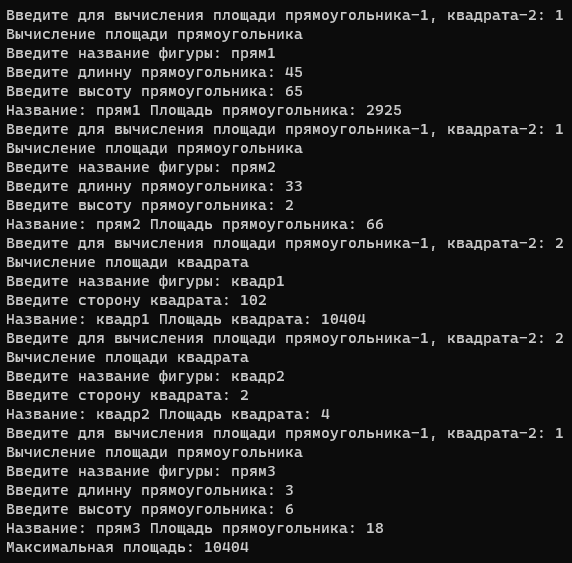


Рисунок 2.1 – Результат работы программы

Источник: собственная разработка