**Лабораторная работа № 2(часть I и II)**

**Широков Ф. А. з22928/2**

**Тема:** Определение компонентов ПО. Выделение структурных единиц. Разработка спецификаций структурных единиц класса. Организация иерархии наследования.

**Вариант 10**: создать базовый класс Rectangle (прямоугольник), содержащий поля: длина, ширина. Создать производные классы Parallelepiped (параллелепипед) и Pyramid (пирамида), содержащие дополнительное поле: высота. Описать методы для вывода на экран сведений об объекте, получения и установки значений полей, методы вычисления площади поверхности и объёма фигур

**Цель работы:** получить практические навыки разработки иерархии классов, методов, свойств классов.

*Указания:*

Требуется описать базовый класс и 1-3 наследованных класса. Каждый разрабатываемый класс должен находиться в отдельном модуле (файле) в составе проекта. Для добавления нового модуля в проект выберите пункт меню «Проект» → «Добавить класс» и укажите название класса.

Класс должен содержать: поля, конструкторы, свойства, методы (или перегруженные методы).

*Разработка структур данных должна производиться по следующему плану:*

1. Объявите базовый класс и опишите его поля. **Поля** должны быть скрыты (уровень доступа private или protected).
2. Опишите конструктор базового класса (конструкторов может быть и несколько, если требуется создавать объекты разными способами).
3. Опишите методы базового класса. **Методы,** которые представляют интерфейсную часть класса, должны быть открыты (уровень доступа public).
4. Продумайте, какие методы должны быть виртуальными и будут переопределены в дочерних классах. Опишите эти методы, пометив их словом virtual.
5. По условию задачи необходимо будет получать или задавать значения некоторых полей. Опишите **свойства**, связанные с этими полями, доступные для чтения (get) и (или) для установки (set). Свойства, также как и методы, будут иметь публичный доступ. Это позволит получать или устанавливать значение скрытого поля из основной программы (из класса Program).
6. Объявите дочерний класс и опишите его дополнительные поля. Дочерний класс наследует поля и методы базового класса.
7. Опишите конструктор дочернего класса (т.к. у каждого класса должен быть свой конструктор).
8. Опишите новые и переопределенные методы дочернего класса. Переопределённые методы пометьте словом override.
9. Для проверки функционала описанных классов **в главной функции** создайте по одному объекту каждого класса и покажите вызов методов. Убедитесь, что переопределенные методы вызываются правильно, в соответствие с типом объекта, к которому применяются.

**Выполнение:**

1. Создаем базовый класс Rectangle.

class Rectangle // базовый класс

{

private double length;

private double width;

public Rectangle() // конструктор по умолчанию

{

length = 5.0;

width = 4.0;

}

public Rectangle(double length, double width)

{

this.length = length;

this.width = width;

}

public double getLength() // геттеры

{

return this.length;

}

public double getWidth()

{

return this.width;

}

public void setLength(double length) // сеттеры

{

this.length = length;

}

public void setWidth(double width)

{

this.width = width;

}

public override string ToString() // вывод в консоль

{

return "It is Rectangle object, length = " + this.length + " , width = " + this.width;

}

public virtual double getArea() // расчет площади

{

return this.length \* this.width;

}

public virtual double getPerimeter() //расчет периметра

{

return 2 \* (this.length + this.width);

}

2. От него образуем два класса наследника, Parallelepiped (параллелепипед) и Pyramid (пирамида):

class Parallelepiped : Rectangle

{

private double height;

public Parallelepiped() : base() // конструтор по умолчанию, с вызовом конструктора базового класса и инициализацией нового поля

{

height = 7.0;

}

public Parallelepiped(double length, double wigth, double height) : base(length, wigth)

{

this.height = height;

}

public double Height //геттер и сеттер в более компактном виде

{

get { return height; }

set { height = value;}

}

public override string ToString()

{

return "It is Parallelepiped object, length = " + this.getLength() + " , width = " + this.getWidth() + ", height = " + this.Height;

}

public override double getArea() // переопределенный метод getArea()

{

return 2 \* base.getArea() + (base.getPerimeter() \* this.height);

}

public double getVolume()

{

return base.getArea() \* this.height;

}

}

class Pyramid : Rectangle

{

private double height;

private double height\_of\_side;

public Pyramid() : base()

{

height = 8.0;

height\_of\_side = height + 3.0;

}

public Pyramid(double length, double wigth, double height) : base(length, wigth)

{

this.height = height;

height\_of\_side = height + 3.0;

}

public double Height

{

get { return height; }

set { height = value; }

}

public double SideHeight

{

get { return height\_of\_side; }

}

public override string ToString()

{

return "It is Pyramid object, length = " + this.getLength() + " , width = " + this.getWidth() + ", height = " + this.Height + ", the side's height = " + SideHeight;

}

public override double getArea()

{

return 4 \* (0.5 \* (base.getArea() \* this.height\_of\_side));

}

public double getVolume()

{

return 0.3333 \* (base.getArea() \* this.height);

}

}

3. Код главной функции, с примером создания объектов с использованием конструктора по умолчанию и конструктора с данными пользователя. Далее показан вызов методов у объектов.

class Program

{

static void Main(string[] args)

{

Rectangle rectangle1 = new Rectangle();

Console.WriteLine(rectangle1.ToString());

Console.WriteLine(rectangle1.getArea());

Rectangle rectangle2 = new Rectangle(12.0, 9.0);

Console.WriteLine(rectangle2.ToString());

Console.WriteLine(rectangle2.getArea());

Parallelepiped parallelepiped1 = new Parallelepiped();

Console.WriteLine(parallelepiped1.ToString());

Console.WriteLine(parallelepiped1.getArea());

Console.WriteLine(parallelepiped1.getVolume());

Parallelepiped parallelepiped2 = new Parallelepiped(8.0, 5.9, 7.0);

Console.WriteLine(parallelepiped2.ToString());

Console.WriteLine(parallelepiped2.getArea());

Console.WriteLine(parallelepiped2.getVolume());

Pyramid pyramid1 = new Pyramid();

Console.WriteLine(pyramid1.ToString());

Console.WriteLine(pyramid1.getArea());

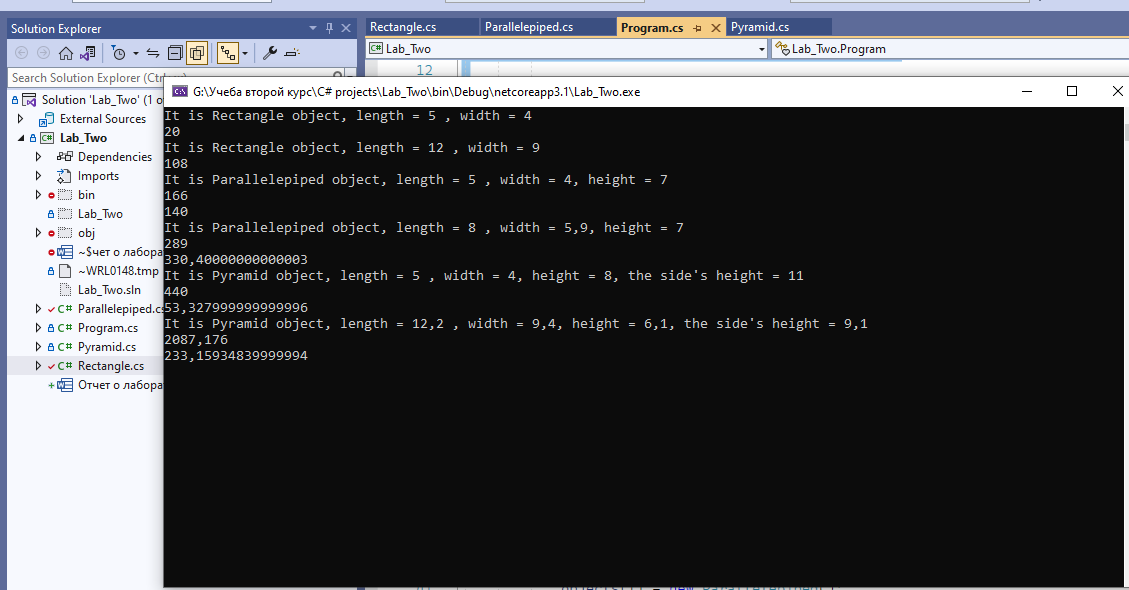
Console.WriteLine(pyramid1.getVolume());

Pyramid pyramid2 = new Pyramid(12.2, 9.4, 6.1);

Console.WriteLine(pyramid2.ToString());

Console.WriteLine(pyramid2.getArea());

Console.WriteLine(pyramid2.getVolume());



4. Для демонстрации переопределения методов в классах наследниках, создадим массив объектов Rectangle, присвоим некоторым объектам ссылки на объекты дочерних классов и вызовем один и тот же метод у всех объектов.

Rectangle[] objects = new Rectangle[6];

objects[0] = new Rectangle();

objects[1] = new Parallelepiped();

objects[2] = new Pyramid();

objects[3] = new Rectangle(Convert.ToDouble(Console.ReadLine()), Convert.ToDouble(Console.ReadLine()));

objects[4] = new Parallelepiped(Convert.ToDouble(Console.ReadLine()), Convert.ToDouble(Console.ReadLine()), Convert.ToDouble(Console.ReadLine()));

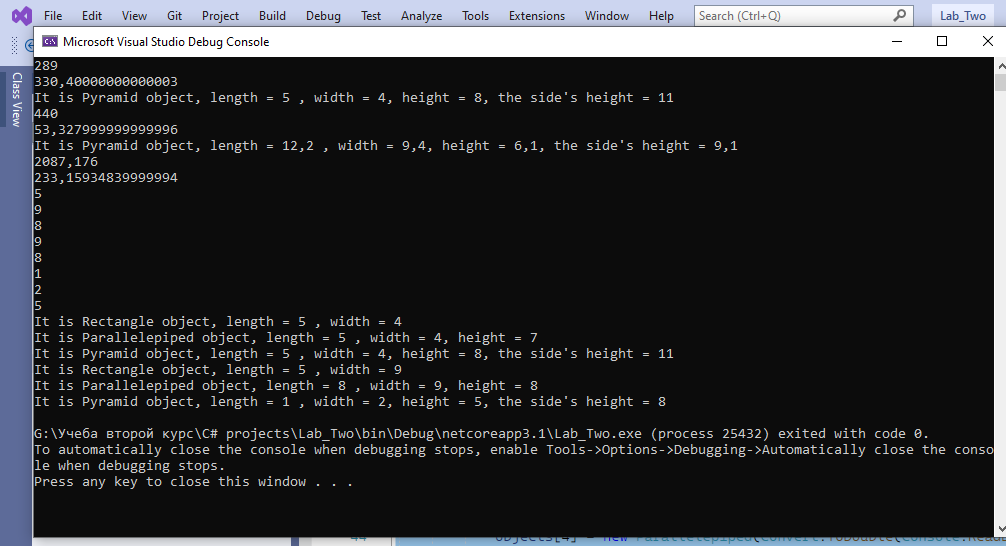
objects[5] = new Pyramid(Convert.ToDouble(Console.ReadLine()), Convert.ToDouble(Console.ReadLine()), Convert.ToDouble(Console.ReadLine()));

foreach (Rectangle shape in objects)

{

Console.WriteLine(shape.ToString());

}



Как видно, метод toString(), который переопределен у всех дочерних классов, правильно указал тип объекта и его поля. Ниже представлена диаграмма классов.

