Лабораторная работа №6

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И МЕХАТРОНИКИ

Кафедра информационных технологий и мехатроники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проведению лабораторных работ по дисциплине
«Объектно-ориентированное программирование»
для студентов специальности 6.050101 “Компьютерные науки”

Разработчик - доцент кафедры информационных технологий и мехатроники
кандидат технических наук, старший научный сотрудник
Тимонин Владимир Алексеевич

Харків 2015

Лабораторная работа №6
Исследование возможностей интегрированной среды разработки Visual C# для создания приложений с классами. Виртуальные методы.

Цель работы – исследовать возможности интегрированной среды разработки Visual Studio 2010 и получить практические навыки по созданию приложений, использующие виртуальные методы.

1. Теоретические сведения

Виртуальным называется такой метод, который объявляется как virtual в базовом классе. Виртуальный метод отличается тем, что он может быть переопределен в одном или нескольких производных классах. Следовательно, у каждого производного класса может быть свой вариант виртуального метода. Кроме того, виртуальные методы интересны тем, что именно происходит при их вызове по ссылке на базовый класс. В этом случае средствами языка C# определяется именно тот вариант виртуального метода, который следует вызывать, исходя из типа объекта, к которому происходит обращение по ссылке, причем это делается во время выполнения. Поэтому при ссылке на разные типы объектов выполняются разные варианты виртуального метода. Иными словами, вариант выполняемого виртуального метода выбирается по типу объекта, а не по типу ссылки на этот объект. Так, если базовый класс содержит виртуальный метод и от него получены производные классы, то при обращении к разным типам объектов по ссылке на базовый класс выполняются разные варианты этого виртуального метода.

Переопределение метода служит основанием для воплощения одного из самых эффективных в C# принципов: динамической диспетчеризации методов, которая представляет собой механизм разрешения вызова во время выполнения, а не компиляции.

Значение динамической диспетчеризации методов состоит в том, что именно благодаря ей в C# реализуется динамический полиморфизм.

1.1. Объявление виртуальных методов

Метод объявляется как виртуальный в базовом классе с помощью ключевого слова **virtual**, указываемого перед его именем. Когда же виртуальный метод переопределяется в производном классе, то для этого используется модификатор **override**. А сам процесс повторного определения виртуального метода в производном классе называется переопределением метода. При переопределении имя, возвращаемый тип и сигнатура переопределяющего метода должны быть точно такими же, как и у того виртуального метода, который переопределяется. Кроме того, виртуальный метод не может быть объявлен как static или abstract.

Пример 1. Нижеприведенный код программы «Пример №1» демонстрирует применение виртуального метода и его переопределение. Результат выполнения кода представлен на рис. 1.

```
class Base
  {
    // Создание виртуального метода в базовом классе
    public virtual void Trigon()
    {
      Console.WriteLine("  Синус 30 градусов равен: {0}", Math.Sin(Math.PI*30/180));
    }
  }
  class Derivedl : Base
  {
```

```
    // Переопределение метода Trigon() в производном классе
    public override void Trigon()
    {
        Console.WriteLine("  Удвоенный синус 30 градусов равен: {0}",
                                                    2 * Math.Sin(Math.PI * 30 / 180));
    }
}
class Derived2 : Base
{
    // Переопределение метода Trigon() в еще одном производном классе
    public override void Trigon()
    {
        Console.WriteLine("  Утроенный синус 30 градусов равен: {0}",
                                                     3 * Math.Sin(Math.PI * 30 / 180));
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.Title = "Пример №1";
        Console.BackgroundColor = ConsoleColor.White;
        Console.Clear();
        Console.ForegroundColor = ConsoleColor.Black;
        Console.WriteLine();
        Base baseOb = new Base();
        Derivedl dObl = new Derivedl();
        Derived2 dOb2 = new Derived2();
        Base baseRef; // ссылка на базовый класс
        baseRef = baseOb;
        baseRef.Trigon();
        baseRef = dObl;
        baseRef.Trigon();
        baseRef = dOb2;
        baseRef.Trigon();
        Console.Read();
    }
}
```

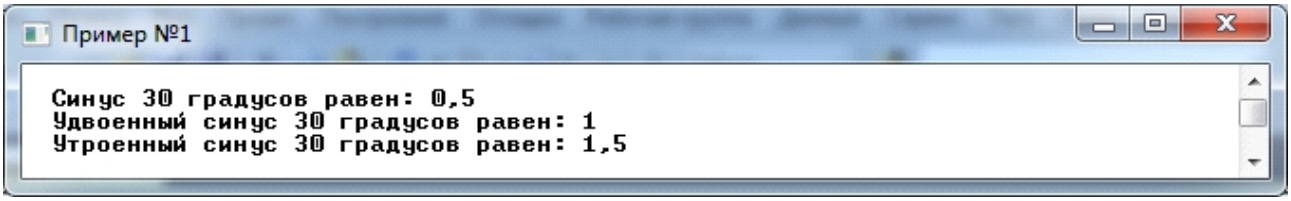

Рис. 1. Результат выполнения кода примера №1.

В коде из приведенного выше примера создаются базовый класс Base и два производных от него класса — Derivedl и Derived2. В классе Base объявляется виртуальный метод Trigon(), который переопределяется в обоих производных классах. Затем в методе Main() объявляются объекты типа Base, Derivedl и Derived2. Кроме того, объявляется переменная baseRef ссылочного типа Base. Далее ссылка на каждый тип объекта присваивается переменной baseRef и затем используется для вызова

метода Trigon(). Вариант выполняемого метода Trigon() определяется по типу объекта, к которому происходит обращение по ссылке во время вызова этого метода, а не по типу класса переменной baseRef.

Переопределять виртуальный метод совсем не обязательно, в этом случае виртуальный метод используется из базового класса.

Если при наличии многоуровневой иерархии виртуальный метод не переопределяется в производном классе, то выполняется ближайший его вариант, обнаруживаемый вверх по иерархии.

1.2. Применение виртуальных методов

Для того чтобы стали понятнее преимущества виртуальных методов, применим их в классе TwoDShape. В предыдущих примерах в каждом классе, производном от класса TwoDShape, определялся метод Area (). Но, по-видимому, метод Area () лучше было бы сделать виртуальным в классе TwoDShape и тем самым предоставить возможность переопределить его в каждом производном классе с учетом особенностей расчета площади той двумерной формы, которую инкапсулирует этот класс. Именно это и сделано в приведенном ниже примере программы. Ради удобства демонстрации классов в этой программе введено также свойство name в классе TwoDShape.

Пример 2. Нижеприведенный код программы «Пример №2» демонстрирует применение виртуальных методов. В качестве базового класса выбрана плоская геометрическая фигура. Результат выполнения программы представлен на рис. 2.

```
class TwoDShape
  {
    double pri_width;
    double pri_height;
    public TwoDShape()
    {
      Width = Height = 0.0;
      name = "null";
    }
    public TwoDShape(double x, string n)
    {
      Width = Height = x;
      name = n;
    }
    public TwoDShape(double w, double h, string n)
    {
      Width = w;
      Height = h;
      name = n;
    }
    // Конструирование копии объекта TwoDShape
    public TwoDShape(TwoDShape ob)
    {
      Width = ob.Width;
      Height = ob.Height;
      name = ob.name;
    }
    // Свойства
    public double Width
    {
      get { return pri_width; }
```

```
        set { pri_width = value < 0 ? -value : value; }
    }
    public double Height
    {
        get { return pri_height; }
        set { pri_height = value < 0 ? -value : value; }
    }
    public string name { get; set; }
    public void ShowDim()
    {
        Console.WriteLine("  Ширина и высота равны " + Width + " и " + Height);
    }
    public virtual double Area()
    {
        Console.WriteLine("  Метод Area() должен быть переопределен");
        return 0.0;
    }
}
// Класс для треугольников, производный от класса TwoDShape
class Triangle : TwoDShape
{
    string Style;
    public Triangle() { Style = "null"; }
    public Triangle(string s, double w, double h) :  base(w, h, "треугольник")
    {
        Style = s;
    }
    // Конструирование равнобедренного треугольника
    public Triangle(double x) : base(x, "треугольник")
    {
        Style = "равнобедренный";
    }
    // Конструирование копии объекта типа Triangle
    public Triangle(Triangle ob) : base(ob)
    {
        Style = ob.Style;
    }
    // Переопределение метода Area() для класса Triangle
    public override double Area()
    {
        return Width * Height / 2;
    }
    public void ShowStyle()
    {
        Console.WriteLine("Треугольник " + Style);
    }
}
// Класс для прямоугольников, производный от класса TwoDShape
class Rectangle : TwoDShape
{
    public Rectangle(double w, double h) :  base(w, h, "прямоугольник"){ }
    // Конструирование квадрата
```

```
    public Rectangle(double x) :  base(x, "прямоугольник") { }
    // Конструирование копии объекта типа Rectangle
    public Rectangle(Rectangle ob) : base(ob) { }
    public bool IsSquare()
    {
        if (Width == Height) return true;
        return false;
    }
    // Переопределение метода Area() для класса Rectangle
    public override double Area()
    {
        return Width * Height;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.Title = "Пример №2";
        Console.BackgroundColor = ConsoleColor.White;
        Console.Clear();
        Console.ForegroundColor = ConsoleColor.Black;
        Console.WriteLine();
        TwoDShape[] shapes = new TwoDShape[5];
        shapes[0] = new Triangle("прямоугольный", 8.0, 12.0);
        shapes[1] = new Rectangle(10);
        shapes[2] = new Rectangle(10, 4);
        shapes[3] = new Triangle(7.0);
        shapes[4] = new TwoDShape(10, 20, "общая форма");
        for(int i=0; i < shapes.Length; i++)
        {
            Console.WriteLine("  Объект — " + shapes[i].name);
            Console.WriteLine("  Площадь равна " + shapes[i].Area());
            Console.WriteLine();
        }
        Console.Read();
    }
}
```

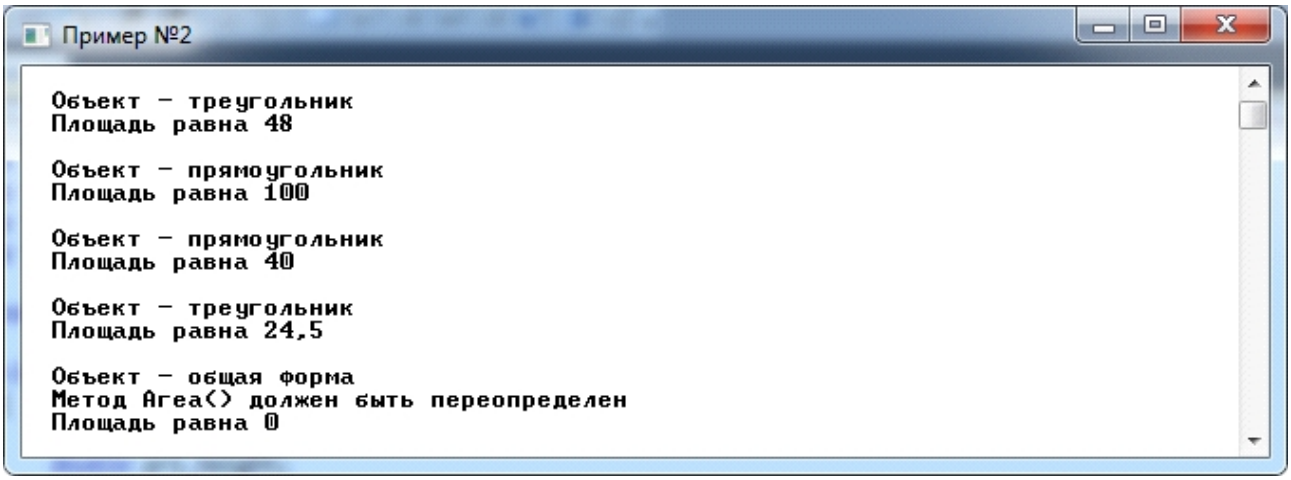

Рис. 2. Результат выполнения кода примера №2.

В классе TwoDShape метод Area() реализован в виде заполнителя, который сообщает о том, что пользователь данного метода должен переопределить его в производном классе. Каждое переопределение метода Area() предоставляет конкретную его реализацию, соответствующую типу объекта, инкапсулируемого в производном классе. Так, если реализовать класс для эллипсов, то метод Area() должен вычислять площадь эллипса.

2. Рабочее задание

Задание 1. Руководствуясь теоретическим материалом раздела 1 изучить возможности языка C# по созданию приложений, использующие виртуальные методы класса, и выполнить примеры, описанные в этом разделе.

Задание 2. Разработать приложение «Вес геометрической фигуры», с помощью которого можно рассчитать веса геометрических фигур (цилиндра, пирамиды и усеченного конуса), изготовленных из различного материала.

Все фигуры изготавливаются из заготовки, имеющей форму прямоугольного параллелепипеда с размерами **a**, **b**, **c** (длина, ширина и высота соответственно). Исходные данные для расчета представлены в таблицах 1 и 2.

Количество фигур задается с клавиатуры.

В качестве базового класса использовать класс «Прямоугольный параллелепипед», разработанный в приложении «Расчет объема пирамиды» (см. лабораторную работу «Исследование возможностей интегрированной среды разработки Visual C# для создания приложений с производными классами»).

Таблица 1. Формулы расчета объемов фигур

Фигура	Формула
Цилиндр	$V = \pi r^2 h$, (r - радиус цилиндра, h - высота цилиндра)
Пирамида	$V = \frac{1}{3} Sh$, (S - площадь основания пирамиды, h - высота пирамиды)
Усеченный конус	$V = \frac{1}{3}\pi h(r1^2 + r1 * r2 + r2^2)$, ($r_1$ - радиус нижнего основания усеченного конуса; r_2 - радиус верхнего основания усеченного конуса; h - высота усеченного конуса)

Таблица 2. Удельный вес цветных металлов

Наименование цветного металла	Удельный вес, г/см^3
Цинк	7,13
Алюминий	2,69808
Свинец	11,337
Олово	7,29
Медь	8,96
Титан	4,505

Никель	8,91
Магний	1,74
Ванадий	6,11
Вольфрам	19,3
Хром	7,19
Молибден	10,22
Серебро	10,5
Тантал	16,65
Золото	19,32
Платина	21,45

3. Контрольные вопросы

1. Что такое виртуальный метод и его отличие от обычных методов?
2. Как объявляются виртуальные методы?
3. Как переопределяются виртуальные методы?

Литература

1. Голощапов А.Л. Microsoft Visual Studio 2010. – СПб.:БХВ-Петербург, 2011. – 544 с.: ил.
2. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ. - М.: «Русская Редакция», 2002.- 576 с.: ил.
3. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ. - М.: «Русская Редакция», 2002.- 624 с.: ил.
4. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. Пер. с англ. - М.: Издательский дом "Вильямс", 2011. — 1392 с.: ил.
5. Фленов М.Е. Библия C#. - СПб.: БХВ-Петербург, 2011.– 560с.: ил.
6. Шилдт Г. C# Учебный курс . – СПб.: Питер, Издательская группа BHV, 2003. – 512 с.: ил.