Пример простой иерархии классов

```
  public void ShowDim()
  {
    Console.WriteLine("Ширина и высота равны " Width + " и " + Height
  }
}

// Класс Triangle, производный от класса TwoDShape.
class Triangle : TwoDShape {
  public string Style; // тип треугольника

  // Возвратить площадь треугольника.
  public double Area() {
    return Width * Height / 2;
  }

  // Показать тип треугольника.
  public void ShowStyle() {
    Console.WriteLine("Треугольник " + Style);
  }
}
```

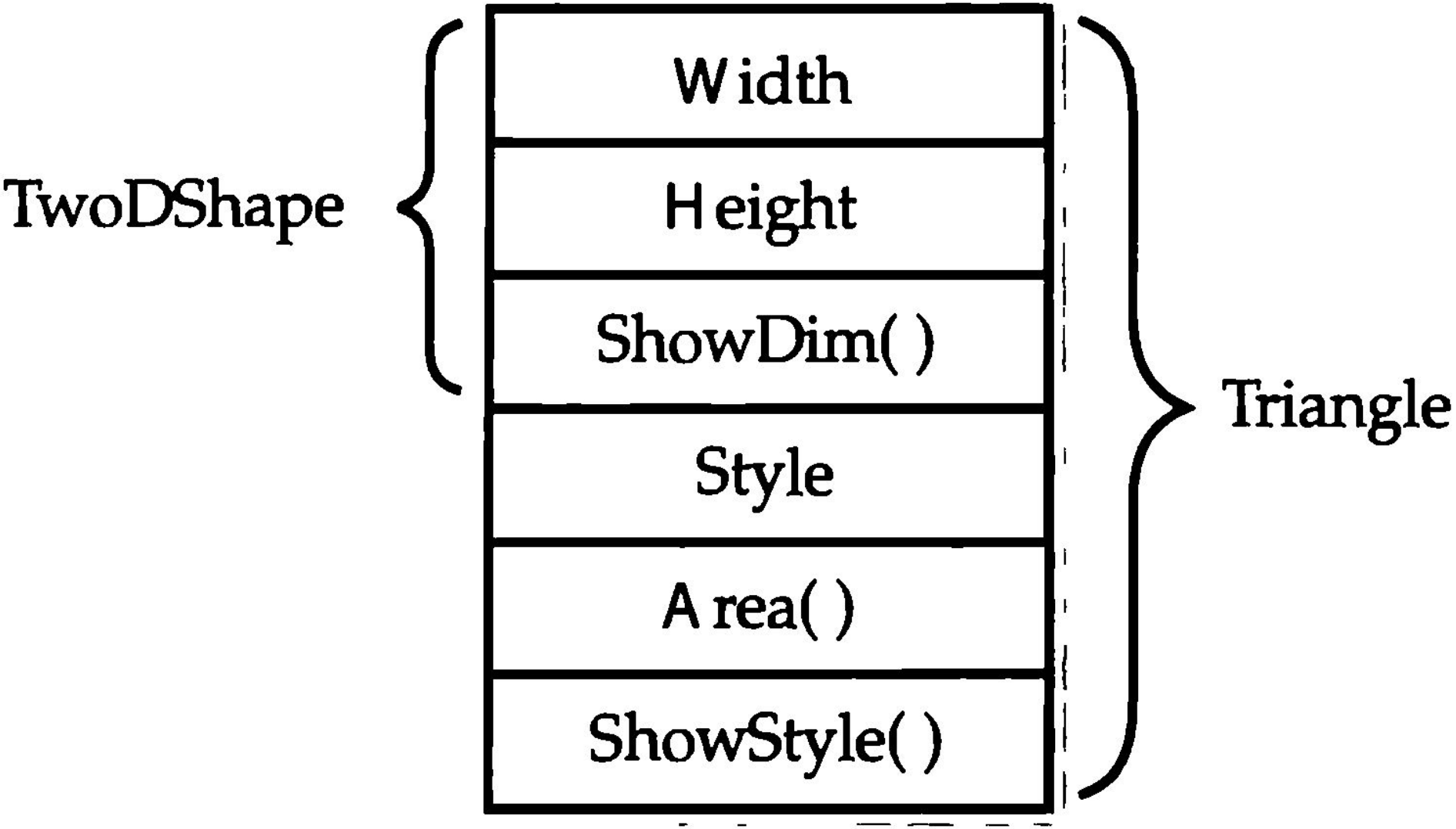
```
class Shapes
{

static void Main()
  {
    Triangle t1 = new Triangle();
    Triangle t2 = new Triangle();

    t1.Width = 4.0;
    t1.Height = 4.0;
    t1.Style = "равнобедренный";

    t2.Width = 8.0;
    t2.Height = 12.0;
    t2.Style = "прямоугольный";
```

Схема представления класса Triangle



Общая форма объявления класса, наследующего от базового класса

class *имя_производного_класса* :
имя_базового_класса
{
// тело класса
}

Еще один класс, производный от класса TwoDShape и инкапсулирующий прямоугольники

```
// Класс для прямоугольников,  производный от класса TwoDShape.
class Rectangle  :  TwoDShape
{
  // Возвратить логическое значение true, если
  // прямоугольник является квадратом.
  public bool IsSquare()
  {
    if(Width == Height)  return true;
    return false;
  }

  // Возвратить площадь прямоугольника.
  public double Area()
  {
    return Width * Height;
  }
}
```

Ошибка! Доступ к закрытым членам класса не наследуется.

```
class TwoDShape
{
  double Width;  // теперь это закрытая переменная
  double Height; // теперь это закрытая переменная

  public void ShowDim()
  {
    Console.WriteLine("Ширина и высота равны " + Width + " и " + Height);
  }
}

class Triangle : TwoDShape // Класс Triangle производный от класса TwoDShape.

{
  public string Style; // тип треугольника


  public double Area() // Возвратить площадь треугольника.
  {
  return Width * Height / 2; // Ошибка. Доступ к закрытому члену класса
запрещен
  }


  public void ShowStyle() // Показать тип треугольника.
{     Console.WriteLine("Треугольник " + Style);   }}
```

Вызов конструкторов базового класса

конструктор_производного_класса
(*список_параметров*):base
(*список_аргументов*)

{

// тело конструктора

}

Конструктор только в производном классе

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

  public double Height
  {
     get { return height; }
     set { height = value < 0 ? -value : value; }
  }

  public void ShowDim()
  {
    Console.WriteLine("Ширина и длина равны " + Width + " и " + Height);
  }
}



class Triangle : TwoDShape
{
  string Style;

  // Конструктор.
  public Triangle(string s, double w, double h)
  {
    Width = w;
    Height = h;
    Style = s;
  }

  // Возвратить площадь треугольника.
  public double Area()
  {
```

Конструкторы и в базовом и в производном классах

```
  }
  // Свойства ширины и высоты объекта.
  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 void ShowDim()
  {
    Console.WriteLine(Width+" "+Height);
  } }
 class Triangle : TwoDShape
{
  string Style;

  // Вызвать конструктор базового      класса.
  public Triangle(string s, double w,          double h) : base(w, h)
  {
    Style = s;
  }

  // Возвратить площадь треугольника.
  public double Area()
  {
    return Width * Height / 2;
```

Сокрытие имен

```
class A
{
  public  int  i = 0;
}

// Создать производный класс.
class B : A
{
  new  int  i; // этот член скрывает
               член i из класса A

  public B(int b)
  {
    i = b;    // член i в классе B
  }

  public void Show()
  {
    Console.WriteLine("Член i в производном классе: " + i);
  }
}
```

Применение ключевого слова base для доступа к скрытому имени

Форма применения ключевого слова base в общем виде:

base.*член*

где *член* может обозначать метод или переменную экземпляра

Применение ключевого слова base для доступа к скрытому имени

```
class A
{   public int i = 0; }

// Создать производный класс.
class B : A
{  new int i; // этот член скрывает член i из класса A

  public B(int a, int b)
  {
    base.i = a; // Здесь обнаруживается скрытый член из класса A
    i = b; // член i из класса B
  }

  public void Show()
  {
    // Здесь выводится член i из класса A.
    Console.WriteLine("Член i в базовом классе: " +base.i);
    // А здесь выводится член i из класса B.
    Console.WriteLine("Член i в производном классе: " +i);
  } }
class UncoverName
{
 static void Main()
 {    B ob = new B(1, 2);    ob.Show();  } }
```

Вызов скрытого метода

```
  public void Show()
  {
    Console.WriteLine("Член i в базовом классе:
                                  " + i);
  }
}

// Создать производный класс.
class B : A
{
  new int i; // этот член скрывает член i из класса A

  public B(int a, int b)
  {
    base.i = a; // здесь обнаруживается скрытый член из класса A
    i = b; // член i из класса B
  }


// Здесь скрывается метод Show() из класса A.
        new public void Show()
          {
            base.Show(); // здесь вызывается метод                Show() из класса A

            // далее выводится член i из класса B
            Console.WriteLine("Член i в производном                    классе: " + i);
          }
        }
```

Ссылки на базовый класс и объекты производных классов

```
class X
{
  int a;

  public X(int i) { a = i; }
}

class Y
{
  int a;

  public Y(int i) { a = i; }
}

class IncompatibleRef
{
  static void Main()
  {
    X x = new X(10);
    X x2;
    Y y = new Y(5);

    x2 = x; // верно, поскольку оба объекта относятся к одному
            // и тому же типу
    X2 = y; // ошибка, поскольку это объекты разного типа. Программа
// компилироваться не будет!!!
  }
}
```

Передать ссылку на объект производного класса переменной ссылки на объект базового класса

```
{
  public int a;

  public X(int i)
  { a = i; }
}

class Y : X
{
  public int b;

  public Y(int i, int j) : base(j)
  {b = i; }
}

class BaseRef
{
  static void Main()
  {
    X x = new X(10);
    X x2;
    Y y = new Y(5, 6);

x2 = x; // верно, поскольку оба объекта относятся к одному типу

    Console.WriteLine("x2.a: " + x2.a);

x2 = y; // тоже верно, поскольку класс Y       производный от класса X

Console.WriteLine("x2.a: " + x2.a);

    // ссылкам на объекты класса X известно только о членах класса X
```

```
  {  get { return width; }
     set { width = value < 0 ? -value : value;}
  }
 public double Height
  {  get { return height; }
     set { height = value < 0 ? -value : value; }}
   public void ShowDim()
  {  Console.WriteLine("Ширина и высота равны " + Width + " и " + Height);
  }}

class Triangle : TwoDShape
{  string Style;

  public Triangle()
  { Style = "null";}

  public Triangle(string s, double w, double h) : base(w, h)  {  Style = s; }

  // Сконструировать копию объекта типа Triangle.
  public Triangle(Triangle ob) : base(ob)
  {Style = ob.Style;}
   // Возвратить площадь треугольника.
  public double Area()
  {  return Width * Height / 2; }

  // Показать тип треугольника.
  public void ShowStyle()
  {   Console.WriteLine("Треугольник " + Style); }
}

class Shapes7 {
  static void Main()
  {
    Triangle t1 = new Triangle("прямоугольный",             8.0, 12.0);

    // Создать копию объекта t1.
    Triangle t2 = new Triangle(t1);
```