



# Основы C#

## Лекция 8.1

Наследование классов. Абстрактные методы и полиморфизм

# Наследование

---

наследование позволяет создавать новый класс на базе другого. Класс, на базе которого создается новый класс, называется базовым, а базирующийся новый класс – наследником или производным классом.

## Объявление наследника

---

*модификаторы* **class** имя\_производного\_класса  
: имя\_базового\_класса  
{операторы\_тела\_производного\_класса}

Конструкция :имя\_базового\_класса в стандарте C# называется спецификацией базы класса.

# Модификаторы доступа

---

Члены базового класса, имеющие статус доступа **private**, как были недоступны для производного класса.

Члены базового класса, имеющие модификатор **public**, открыты для членов и объектов производного класса.

Члены базового класса, имеющие статус доступа **protected** доступны (открыты) для членов производного класса, но в то же время были закрыты (недоступны) для объектов производного класса

# Модификаторы доступа

---

В производном классе обычно вводятся новые члены, определяющие новое поведение и дополнительные характеристики объектов производного класса. Для новых (не унаследованных) членов производных классов имена выбираются произвольно.

Если в производном классе объявлен член, имя которого совпадает с именем какого-либо члена базового класса, то для их различения в производном классе используются уточненные имена:

**this.имя\_члена\_производного\_класса**

**base.имя\_члена\_базового\_класса**

# this. и base.

---

```
class Disk // Класс круг
{
    protected double rad; // Радиус круга
    protected Disk(double ri) { rad = ri; } // Конструктор
    protected double Area    { get { return rad * rad * Math.PI; } }
}

class Ring : Disk // Класс кольцо
{
    new double rad; // Радиус внутренней окружности
    public Ring(double Ri, double ri): base(Ri) { rad = ri; } // Конструктор
    public new double Area { get { return base.Area - Math.PI * rad * rad; } }

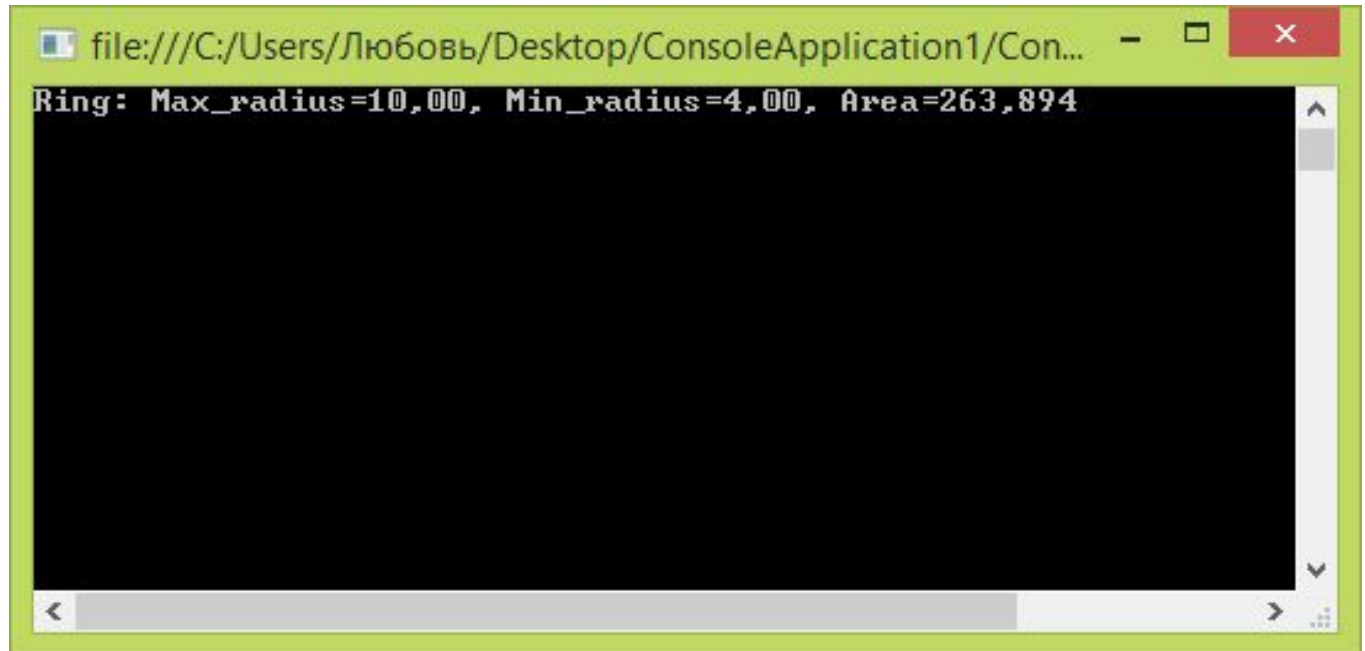
    public void print()
    {
        Console.WriteLine("Ring: Max_radius={0:f2}, " + "Min_radius={1:f2}," + «
Area={2:f3}", base.rad, rad, Area);
    }
}
```

# this. и base.

---

```
class Program
```

```
{  
    static void Main(string[] args)  
    {  
        Ring rim = new Ring(10.0, 4.0);  
        rim.print();  
    }  
}
```

A screenshot of a Windows console application window. The title bar shows the file path: file:///C:/Users/Любовь/Desktop/ConsoleApplication1/Con... The window has standard minimize, maximize, and close buttons. The console output displays the text: Ring: Max\_radius=10,00, Min\_radius=4,00, Area=263,894. The console area is black with white text. There are scrollbars on the right and bottom of the console area.

```
file:///C:/Users/Любовь/Desktop/ConsoleApplication1/Con...  
Ring: Max_radius=10,00, Min_radius=4,00, Area=263,894
```

# Конструкторы при наследовании

---

Конструкторы не наследуются.

Конструктор базового класса необходимо явно вызвать из инициализатора конструктора производного класса.

При отсутствии вызова компилятор по умолчанию дополнит объявление конструктора обращением к конструктору базового класса без параметров.



# Одноименные методы базового и производного классов

---

1. для методов возможна перегрузка (overload). Одноименные методы базового и производного классов должны иметь **разные спецификации параметров**.

2. Переопределение (экранирование или сокрытие (hiding)), методом производного класса одноименного метода базового класса (**спецификации параметров совпадают**). Для избегания используется модификатор **new**.

3. метод базового класса может быть объявлен как виртуальный (virtual), при его переопределении (overriding) в производных классах обеспечивается полиморфизм.

# Переопределение метода

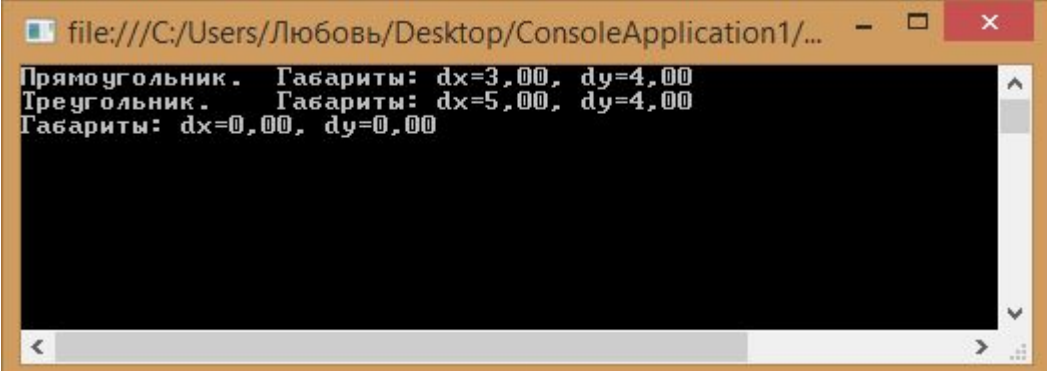
---

```
class Figure // Базовый класс
{
    protected double dx, dy; // Размеры вдоль осей
    public void print()
    { Console.WriteLine("Габариты: dx={0:f2}, dy={1:f2}", dx, dy); }
}
class Rectangle : Figure //Производный класс – прямоугольник:
{
    public Rectangle(double xi, double yi)
    { dx = xi; dy = yi; }
    public new void print()
    { Console.Write("Прямоугольник. \t"); base.print(); }
}
class Triangle : Figure //Производный класс – треугольник:
{
    public Triangle(double xi, double yi)
    { dx = xi; dy = yi; }
    public new void print()
    { Console.Write("Треугольник. \t"); base.print(); }
}
```

# Переопределение метода

---

```
static void Main(string[] args)
{
    Rectangle rec = new Rectangle(3.0, 4.0);
    rec.print();
    Triangle tre = new Triangle(5.0, 4.0);
    tre.print();
    Figure fig = new Figure();
    fig.print();
    Console.ReadLine();
}
```



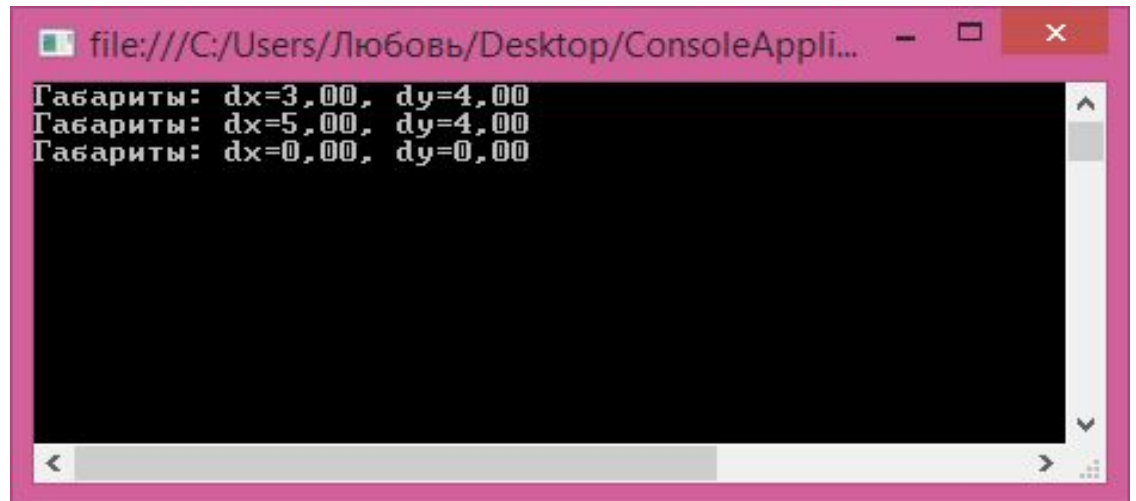
The screenshot shows a console window titled "file:///C:/Users/Любовь/Desktop/ConsoleApplication1/...". The output text is as follows:

```
Прямоугольник.  Габариты: dx=3,00, dy=4,00
Треугольник.    Габариты: dx=5,00, dy=4,00
Габариты: dx=0,00, dy=0,00
```

# Переопределение метода

---

```
static void Main(string[] args)
{
    Figure fig1 = new Rectangle(3.0, 4.0);
    Figure fig2 = new Triangle(5.0, 4.0);
    Figure fig3 = new Figure();
    fig1.print();
    fig2.print();
    fig3.print();
}
```



The screenshot shows a console window titled "file:///C:/Users/Любовь/Desktop/ConsoleAppli...". The output of the program is displayed in three lines:

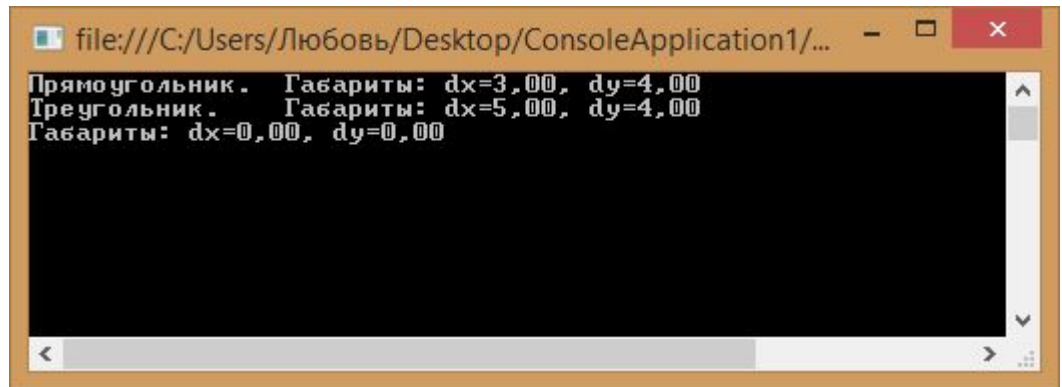
```
Габариты: dx=3,00, dy=4,00
Габариты: dx=5,00, dy=4,00
Габариты: dx=0,00, dy=0,00
```

# Виртуальные методы

---

`public virtual void print();` для базового класса

`public override void print();` для производных классов



The screenshot shows a Windows console application window with a title bar containing the file path `file:///C:/Users/Любовь/Desktop/ConsoleApplication1/...`. The console output displays the dimensions of three shapes in Russian: a rectangle, a triangle, and a point. The text is as follows:

```
Прямоугольник. Габариты: dx=3,00, dy=4,00
Треугольник. Габариты: dx=5,00, dy=4,00
Габариты: dx=0,00, dy=0,00
```

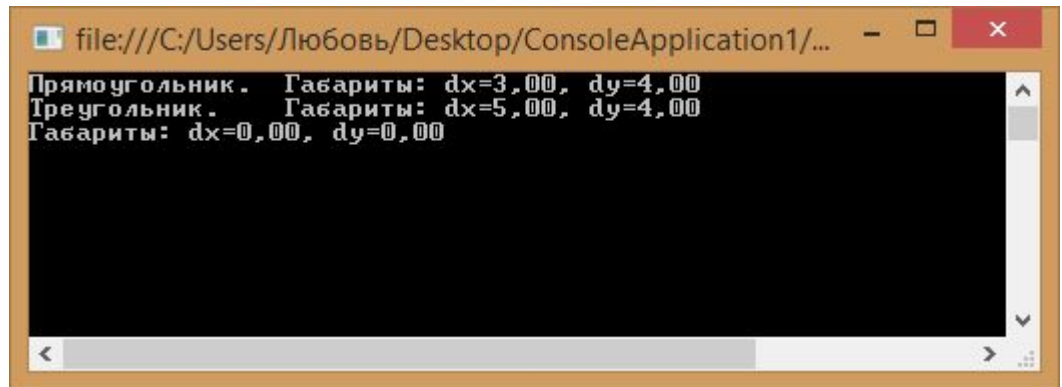
# Виртуальные методы и полиморфизм

---

`public virtual void print();` для базового класса

`public override void print();` для производных классов

**Виртуальный метод** – это метод, который может быть переопределен в классе наследнике.



```
file:///C:/Users/Любовь/Desktop/ConsoleApplication1/...
Прямоугольник. Габариты: dx=3,00, dy=4,00
Треугольник. Габариты: dx=5,00, dy=4,00
Габариты: dx=0,00, dy=0,00
```

# Абстрактные методы и классы. Особенности

---

1. Абстрактный метод может быть объявлен только в абстрактном классе.
2. В заголовке абстрактного метода указывается модификатор **abstract**.
3. У абстрактного метода после скобки, ограничивающей спецификацию параметров, помещается символ ; и не тела.
4. Абстрактный метод по умолчанию является виртуальным (добавлять модификатор **virtual** не требуется)
5. Для того чтобы класс был определен как абстрактный, в его заголовок помещают модификатор **abstract**.
6. Создавать объекты абстрактных классов невозможно.
7. Если в абстрактном классе объявлены несколько абстрактных методов, а производный класс содержит реализацию не всех из них, то производный класс в свою очередь становится абстрактным.
8. В абстрактном классе могут быть определены любые не абстрактные члены (методы, поля

# Абстрактные методы и классы. Особенности

---

1. Абстрактный метод может быть объявлен только в абстрактном классе.
2. В заголовке абстрактного метода указывается модификатор **abstract**.
3. У абстрактного метода после скобки, ограничивающей спецификацию параметров, помещается символ ; и не тела.
4. Абстрактный метод по умолчанию является виртуальным (добавлять модификатор **virtual** не требуется)
5. Для того чтобы класс был определен как абстрактный, в его заголовок помещают модификатор **abstract**.
6. Создавать объекты абстрактных классов невозможно.
7. Если в абстрактном классе объявлены несколько абстрактных методов, а производный класс содержит реализацию не всех из них, то производный класс в свою очередь становится абстрактным.
8. В абстрактном классе могут быть определены любые не абстрактные члены (методы, поля).



# Абстрактные методы и классы.

---

```
abstract class Figure // Абстрактный базовый класс
```

```
{
```

```
    protected double dx, dy; // Размеры
```

```
    public abstract void print();
```

```
    abstract public double square();
```

```
}
```

# Абстрактные методы и классы.

---

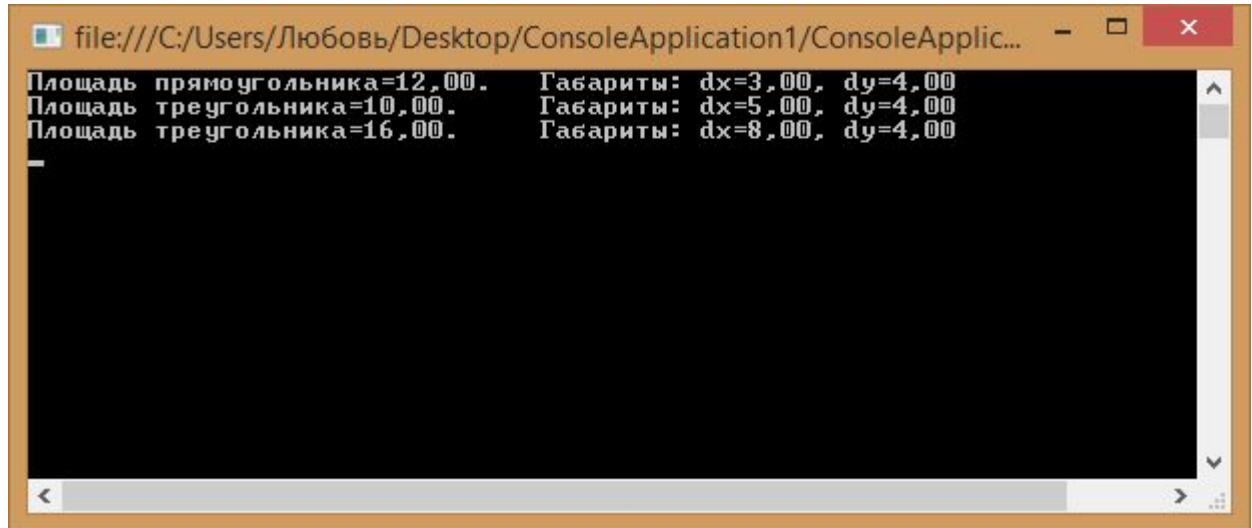
```
class Rectangle : Figure
{
    public Rectangle(double xi, double yi)
    { dx = xi; dy = yi; }
    public override void print()
    {
        Console.Write("Площадь прямоугольника={0:f2}. \t", square());
        Console.WriteLine("Габариты: dx={0:f2}, dy={1:f2}", dx, dy);
    }
    public override double square() { return dx * dy; }
}

class Triangle : Figure
{
    public Triangle(double xi, double yi)
    { dx = xi; dy = yi; }
    public override void print()
    {
        Console.Write("Площадь треугольника={0:f2}. \t", square());
        Console.WriteLine("Габариты: dx={0:f2}, dy={1:f2}", dx, dy);
    }
    public override double square() { return dx * dy / 2; }
}
```

# Абстрактные методы и классы.

---

```
static void Main(string[] args)
{
    Figure fig = new Rectangle(3.0, 4.0);
    fig.print();
    fig = new Triangle(5.0, 4.0);
    fig.print();
    Triangle tri = new Triangle(8.0, 4.0);
    tri.print();
}
```



The screenshot shows a console window titled "file:///C:/Users/Любовь/Desktop/ConsoleApplication1/ConsoleApplic...". The output text is as follows:

```
Площадь прямоугольника=12,00.    Габариты: dx=3,00, dy=4,00
Площадь треугольника=10,00.      Габариты: dx=5,00, dy=4,00
Площадь треугольника=16,00.      Габариты: dx=8,00, dy=4,00
```