Лекция 5. Наследование

Наследование является мощнейшим инструментом ООП. Оно позволяет строить иерархии, в которых классы-потомки получают свойства классов-предков и могут дополнять их или изменять. Наследование применяется для следующих взаимосвязанных целей: исключения из программы повторяющихся фрагментов кода; упрощения модификации программы; упрощения создания новых программ на основе существующих. Кроме того, наследование является единственной возможностью использовать классы, исходный код которых недоступен, но которые требуется использовать с изменениями.

Класс в C# может иметь произвольное количество потомков и только одного предка. При описании класса имя его предка записывается в заголовке класса после двоеточия. Если имя предка не указано, предком считается базовый класс всей иерархии System.Object:

[атрибуты] [спецификаторы] class имя_класса [: предки] тело класса

особенности: - класс в C# может иметь произвольное количество потомков
- класс можно наследовать только от одного класса-предка и от произвольного количества интерфейсов
- при наследовании потомок получает все элементы предка
- элементы private не доступны потомку непосредственно
- элементы protected доступны только потомкам.

Базовый класс обеспечивает для производных классов:
- Элементы, которые потомки могут использовать непосредственно
- Функции, которые должны быть реализованы после их переопределения

Абстрактный класс (abstract)– показывает, что класс фиктивный и имеет пустое тело; конкретное значение присваивается в производных классах. Абстрактные классы предназначены для представления общих понятий, которые предполагается конкретизировать в производных классах. Закрытые классы и методы (sealed) – от такого класса нельзя ничего наследовать, а метод нельзя переопределить. Наследование, вместе с инкапсуляцией и полиморфизмом, является одной из трех основных характеристик (или базовых понятий) объектно-ориентированного программирования. Наследование позволяет создавать новые классы, которые повторно используют, расширяют и изменяют поведение, определенное в других классах. Класс, члены которого наследуются, называется базовым классом, а класс, который наследует эти члены, называется производным классом. Производный класс может иметь только один непосредственный базовый класс. Однако наследование является транзитивным. Если ClassC является производным от ClassB, и ClassB является производным от ClassA, ClassC наследует члены, объявленные в ClassB и ClassA.
C++:язык, поддерживающий множественное наследование

Пример наследования

```
class Point
   {
     public double x;
      public double y;
      public Point()
{
       }
```

```
public Point(double a, double b)
      {
         x = a;
         y = b;
      }
  class Line : Point
      {
     public Point A;
      public Point B;
       public Line()
         {
         }
      public Line(Point A1, Point B1)
         {
          this.A = A1;
           this.B = B1;
         }
?public double dlina(Line C)
         {
       if ((C.A.x == null) || (C.B.x == null) || (C.A.y == null) || (C.B.y == null))
            {
            return 0;
            }
       else
            {
          double t = C.A.x + C.B.x;
        return t;
```

```
            }
        }
    }
class Program
    {
     static void Main(string[] args)
        {
    double x = 1;
     double y = 1;
    Point A = new Point(x, y);
   Point B = new Point(0, 0);
      Line E = new Line(A, B);
       double s = E.dlina(E);
          Console.WriteLine(s);
     Console.ReadKey();
        }
    }
}
```

Спецификаторы

public Неограниченный доступ.

protected Доступ ограничен содержащим классом или типами, которые являются производными от содержащего класса.

private Доступ ограничен содержащим типом.

Ключевое слово private является модификатором доступа к члену. Закрытый (private) доступ является уровнем доступа с минимальными правами. Доступ к закрытым членам можно получить только внутри тела класса или структуры, в которой они объявлены, как показано в следующем примере:

```
using System;

class Test
```

```
 {
private int Compute1()
 {  return 1; // Private instance method that computes something.
    }
 public int Compute2()
  {  return this.Compute1() + 1; // Public instance method.
  }
private static int Compute3()
  {  return 3; // Private static method that computes.
   }
public static int Compute4()
  {  return Compute3() + 1; // Public static method.
  }
}

class Program
 {     static void Main()
 {    Test test = new Test();
 Console.WriteLine(test.Compute2());
 Console.WriteLine(Test.Compute4());
  }
}
```

Конструкторы не наследуются, поэтому производный класс должен иметь собственные конструкторы

Поля, методы и свойства класса наследуются.