C# Polymorphism

The term "Polymorphism" is the combination of "poly" + "morphs" which means many forms. It is a greek word. In object-oriented programming, we use 3 main concepts: inheritance, encapsulation and polymorphism.

There are two types of polymorphism in C#: compile time polymorphism and runtime polymorphism. Compile time polymorphism is achieved by method overloading and operator overloading in C#. It is also known as static binding or early binding. Runtime polymorphism in achieved by method overriding which is also known as dynamic binding or late binding.

C# Method Overriding

Let's see a simple example of runtime polymorphism in C#.

```
1. using System;
2. public class Animal
3. {
4.     public virtual void eat()
5.     {
6.         Console.WriteLine("eating...");
7.     }
8. }
9.
10. public class Dog: Animal
11. {
12.     public override void eat()
13.     {
14.         Console.WriteLine("eating bread...");
15.     }
16.
17. }
18. public class TestPolymorphism
19. {
20.     public static void Main()
21.     {
22.         Animal a= new Dog();
23.         a.eat();
24.     }
25. }
```

Output: eating bread...

C# Method Overloading

You can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other by the types and/or the number of arguments in the argument list. You cannot overload function declarations that differ only by return type.

The following example shows using function **print()** to print different data types –

```
using System;

namespace PolymorphismApplication
{
  class Printdata
   {
      void print(int i)
      {
        Console.WriteLine("Printing int: {0}", i );
       }

      void print(double f)
     {
        Console.WriteLine("Printing float: {0}" , f);
      }

    void print(string s)
   {
      Console.WriteLine("Printing string: {0}", s);
    }

    static void Main(string[] args)
   {

    Printdata p = new Printdata();

      p.print(5);

      p.print(500.263);

      p.print("Hello C++");
      Console.ReadKey();
    }
  }
}
```

When the above code is compiled and executed, it produces the following result −

```
Printing int: 5
Printing float: 500.263
Printing string: Hello C++
```