**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# Runtime Polymorphism Example**

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

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

Output:

eating bread...

**C# Runtime Polymorphism Example 2**

Let's see a another example of runtime polymorphism in C# where we are having two derived classes.

1. using System;
2. public class Shape{
3. public virtual void draw(){
4. Console.WriteLine("drawing...");
5. }
6. }
7. public class Rectangle: Shape
8. {
9. public override void draw()
10. {
11. Console.WriteLine("drawing rectangle...");
12. }
14. }
15. public class Circle : Shape
16. {
17. public override void draw()
18. {
19. Console.WriteLine("drawing circle...");
20. }
22. }
23. public class TestPolymorphism
24. {
25. public static void Main()
26. {
27. Shape s;
28. s = new Shape();
29. s.draw();
30. s = new Rectangle();
31. s.draw();
32. s = new Circle();
33. s.draw();
35. }
36. }

Output:

drawing...

drawing rectangle...

drawing circle...

**Runtime Polymorphism with Data Members**

Runtime Polymorphism can't be achieved by data members in C#. Let's see an example where we are accessing the field by reference variable which refers to the instance of derived class.

1. using System;
2. public class Animal{
3. public string color = "white";
5. }
6. public class Dog: Animal
7. {
8. public string color = "black";
9. }
10. public class TestSealed
11. {
12. public static void Main()
13. {
14. Animal d = new Dog();
15. Console.WriteLine(d.color);
17. }
18. }

Output:

white