Polymorphism

Course Code : CS-127

Course Title : Object Oriented Programming

Semester : 2nd

Polymorphism

- In c#, Polymorphism means providing an ability to take more than one form and it's one of the main pillar concepts of object-oriented programming, after encapsulation and inheritance.

- Generally, the polymorphism is a combination of two words, one is poly and another one is morphs. Here poly means "multiple" and morphs means "forms" so polymorphism means many forms.

In c#, we have two different kinds of polymorphisms available, those are

- ✓ Compile Time Polymorphism
- ✓ Run Time Polymorphism

C# Compile Time Polymorphism

- Compile Time Polymorphism means defining multiple methods with the same name but with different parameters. By using compile-time polymorphism, we can perform different tasks with the same method name by passing different parameters.

- In c#, the compile-time polymorphism can be achieved by using **method overloading** and it is also called early binding or static binding.

Implementing **method overloading** to achieve compile-time polymorphism

```
public class Calculate
{
   public void AddNumbers(int a, int b)
   {
      Console.WriteLine("a + b = {0}", a + b);
   }
   public void AddNumbers(int a, int b, int c)
   {
      Console.WriteLine("a + b + c = {0}", a + b + c);
   }
}
```

If you observe above "**Calculate**" class, we defined a two methods with same name (**AddNumbers**) but with different input parameters to achieve **method overloading**, this is called a **compile time polymorphism** in c#.

Compile Time Polymorphism Example

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication61
{
   class sum
   {   //compile-time polymorphism can be achieved by using method overloading
      public void addnumbers(int a, int b)
      {
         Console.WriteLine("addnumbers(int a, int b)   "+(a+b));
      }
      public void addnumbers(int a, int b, int c)//1 method overloading
      {
         Console.WriteLine("addnumbers(int a, int b, int c)  " + (a + b + c));
      }
      public void addnumbers(double a, int b)// 2 method overloading
      {
         Console.WriteLine("addnumbers(double a, int b)  " + (a + b));
      }
      public void addnumbers(double a, double b)//3 method overloading
      {
         Console.WriteLine("addnumbers(double a, double b)   " + (a + b));
      }
      public void addnumbers(int a, double b, int c)// 4 method overloading
      {
         Console.WriteLine("addnumbers(int a, double b, int c)  " + (a + b + c));
      }
   }
   class Program
   {
      static void Main(string[] args)
      {
         sum s = new sum();
         s.addnumbers(5.5,5);
         Console.Read();
      }
   }
}
```

C# Run Time Polymorphism

- Run Time Polymorphism means overriding a base class method in the derived class by creating a similar function and this can be achieved by using override & virtual keywords along with inheritance principle.

- By using run-time polymorphism, we can override a base class method in the derived class by creating a method with the same name and parameters to perform a different task.

- In c#, the run time polymorphism can be achieved by using method overriding and it is also called late binding or dynamic binding.

implementing a **method overriding** to achieve run time polymorphism

```
// Base Class
public class Users
{
    public virtual void GetInfo()
    {
        Console.WriteLine("Base Class");
    }
}
// Derived Class
public class Details : Users
{
    public override void GetInfo()
    {
        Console.WriteLine("Derived Class");
    }
}
```

we created a two classes (“Users”, “Details”) and the derived class (Details) is inheriting the properties from base class (Users) and we are overriding the base class method GetInfo in derived class by creating the same function to achieve method overriding, this is called a run time polymorphism in c#.

we defined a GetInfo method with a virtual keyword in the base class to allow derived class to override that method using the override keyword.

Run Time Polymorphism Example

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication61
{
   //run time polymorphism can be achieved by using method overriding
   class shape
   {
      public virtual void Area(int a)
      {
         Console.WriteLine("Area Calculator");
      }
      public void test()
      {
         Console.WriteLine("Area Calculator");
      }
   }
   class circle : shape
   {
      public override void Area(int a)//1 method overriding
      {
         Console.WriteLine(" Circle Area ");
         int r;
         double A;
         Console.WriteLine(" Enter the Radius :");
         r= Convert.ToInt16(Console.ReadLine());
         A = (3.14) * r *r;
         Console.WriteLine(" Area of Circle : "+A);
      }
   }
   class rect : shape
   {
      public override void Area(int a)//2  method overriding
      {
         Console.WriteLine(" Rectangle Area ");
         int A, length, width;
         Console.WriteLine(" Enter Length ");
         length = Convert.ToInt16(Console.ReadLine());
         Console.WriteLine(" Enter Width ");
         width = Convert.ToInt16(Console.ReadLine());
         A = length * width;
         Console.WriteLine(" Area of Rectangle : " + A);
      }
   }
   class Program
   {
      static void Main(string[] args)
      {
         //circle c = new circle();
         //c.Area();
         rect r = new rect();
         r.Area(1);
         r.test();

         Console.Read();
      }
   }}
```