



C# OOP

(Object Oriented Programming)

Beginner to Advance



www.techprofree.com

C#- What Is OOP



Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods



C#- Classes & Object

class is a template definition of the methods and variables in a particular kind of object

object is a specific instance of a class; it contains real values instead of variables



Example

class

Fruit

objects

Apple

Banana

Mango



Create Class

To create a class, use the `class` keyword

Example



Create a class named "`Car`" with a variable color

```
class Car
{
    string color = "red";
}
```



Create Object

To create an object of `Car`, specify the class name, followed by the object name, and use the keyword `new`

Example

Create an object called "`myObj`" and use it to print the value of `color`

```
class Car
{
    string color = "red";
    static void Main(string[] args)
    { Car myObj = new Car();
      Console.WriteLine(myObj.color);
    } }

```



Create Object

To create an object of `Car`, specify the class name, followed by the object name, and use the keyword `new`

Example

Create an object called "`myObj`" and use it to print the value of `color`

```
class Car
{
    string color = "red";
    static void Main(string[] args)
    { Car myObj = new Car();
      Console.WriteLine(myObj.color);
    } }
}
```



Class Members

Fields and methods inside classes are often referred to as "Class Members"

Example

Create a **Car** class with three class members: two fields and one method

```
// The class
class MyClass
{ // Class members
  string color = "red"; // field
  int maxSpeed = 200; // field
  public void fullThrottle() // method
  { Console.WriteLine("The car is going as
  fast as it can!"); } }
```



Field

In the previous chapter, you learned that variables inside a class are called fields, and that you can access them by creating an object of the class, and by using the dot syntax (.)



Example

```
class Car
{
    string color = "red";
    int maxSpeed = 200;
    static void Main(string[] args)
    {
        Car myObj = new Car();
        Console.WriteLine(myObj.color);
        Console.WriteLine(myObj.maxSpeed);
    }
}
```



Constructor

A constructor is a **special method** that is used to initialize objects. The advantage of a constructor, is that it is called when an object of a class is created. It can be used to set initial values for fields:

Example



```
// Create a Car class
class Car { public string model; // Create a
field
// Create a class constructor for the Car class
public Car() { model = "Mustang"; // Set the
initial value for model } static void
Main(string[] args)
{ Car Ford = new Car(); // Create an object of
the Car Class (this will call the constructor)
Console.WriteLine(Ford.model); // Print the
value of model } } // Outputs "Mustang"
```



Inheritance



In C#, it is possible to inherit fields and methods from one class to another. We group the "inheritance concept" into two categories:

- **Derived Class (child)** - the class that inherits from another class
 - **Base Class (parent)** - the class being inherited from
- To inherit from a class, use the : symbol.



Example

```
class Vehicle // base class (parent)
{ public string brand = "Ford"; // Vehicle
field
public void honk() // Vehicle method
{ Console.WriteLine("Tuut, tuut!"); } }
class Car : Vehicle // derived class (child)
{ public string modelName = "Mustang"; // Car
field }
class Program { static void Main(string[] args)
{ // Create a myCar object
Car myCar = new Car(); // Call the honk() method
(From the Vehicle class) on the myCar object
myCar.honk(); // Display the value of the brand
field (from the Vehicle class) and the value of
the modelName from the Car class
Console.WriteLine(myCar.brand + " " +
myCar.modelName); } }
```



Polymorphism

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

Like we specified in the previous chapter; Inheritance lets us inherit fields and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.

For example, think of a base class called **Animal** that has a method called **animalSound()**. Derived classes of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.)



Example



```
class Animal // Base class (parent)
{ public void animalSound()
{ Console.WriteLine("The animal makes a
sound"); } }
class Pig : Animal // Derived class (child)
{ public void animalSound()
{ Console.WriteLine("The pig says: wee wee");
} }
class Dog : Animal // Derived class (child)
{ public void animalSound()
{ Console.WriteLine("The dog says: bow wow");
} }
```

