Object Oriented Programming

Lecture 4: **Classes and Objects**

Assistant Lecturer

Sipan M. Hameed

www.sipan.dev

2024-2025

Contents

Object Oriented Programming

Object-Oriented Programming (OOP) is a programming paradigm in computer science that relies on the concept of **classes** and **objects**. It is used to structure a software program into simple, reusable pieces of code blueprints (usually called **classes**), which are used to create individual instances of **objects**.

OOP is widely used in many programming languages, including Java, C++, Python, and **C#.** It provides a way to structure and design software in a more modular and maintainable manner, making it easier to understand, extend, and maintain large codebases.

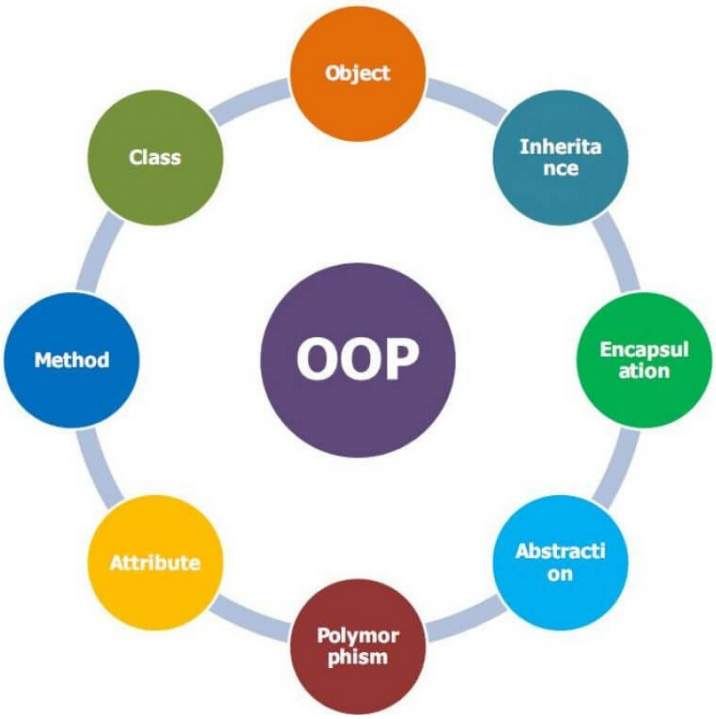
Procedural programming is about writing procedures or methods that perform operations on the data, while **OOP** is about creating **objects** that contain **both data and methods**.

OOP has several advantages over procedural programming:

- OOP is **faster** and **easier** to execute
- OOP provides a **clear structure** for the programs
- OOP helps to keep the C# code **DRY** "Don't Repeat Yourself", and makes the code **easier to maintain**, **modify** and **debug**
- OOP makes it possible to create full **reusable** applications with less code and shorter development time
- Securely **protects sensitive information** through encapsulation
- And more……

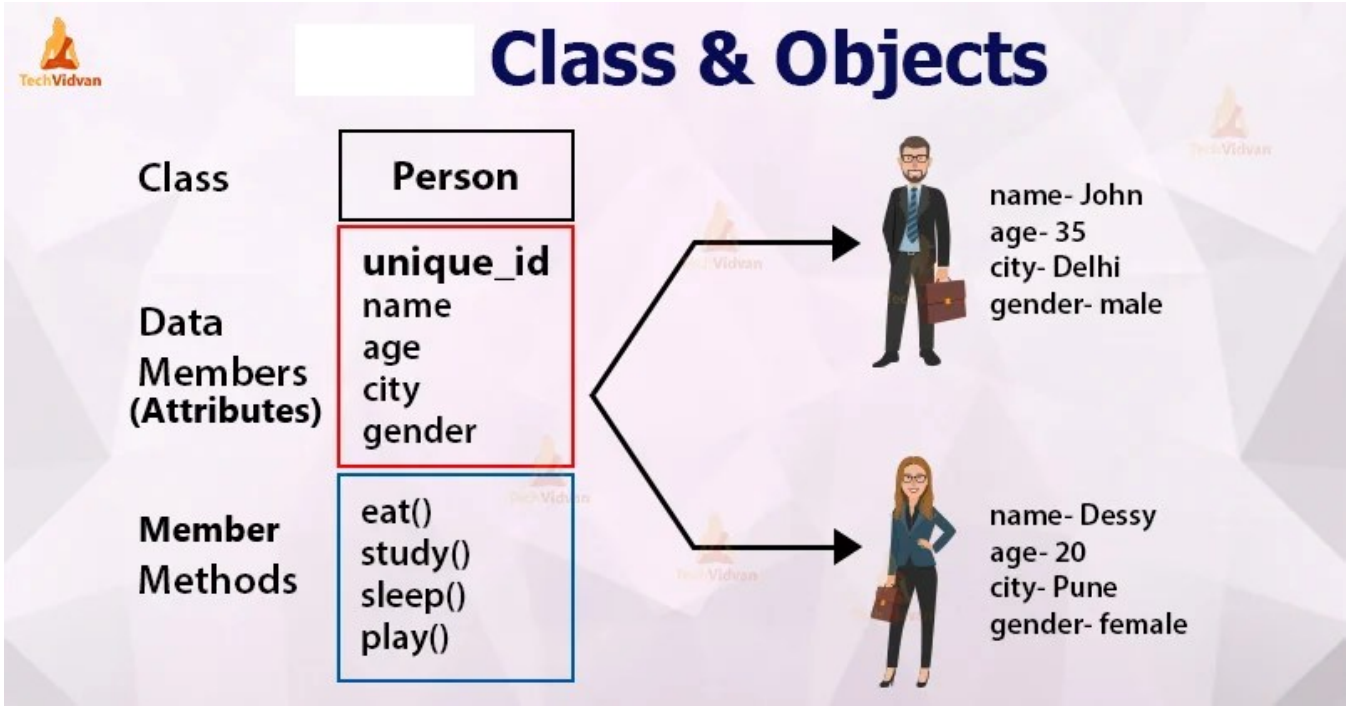
Basic concepts in OOP:

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism



Class and Objects

Classes and objects are the two main aspects of object-oriented programming.
Look at the following illustration to see the difference between class and objects:



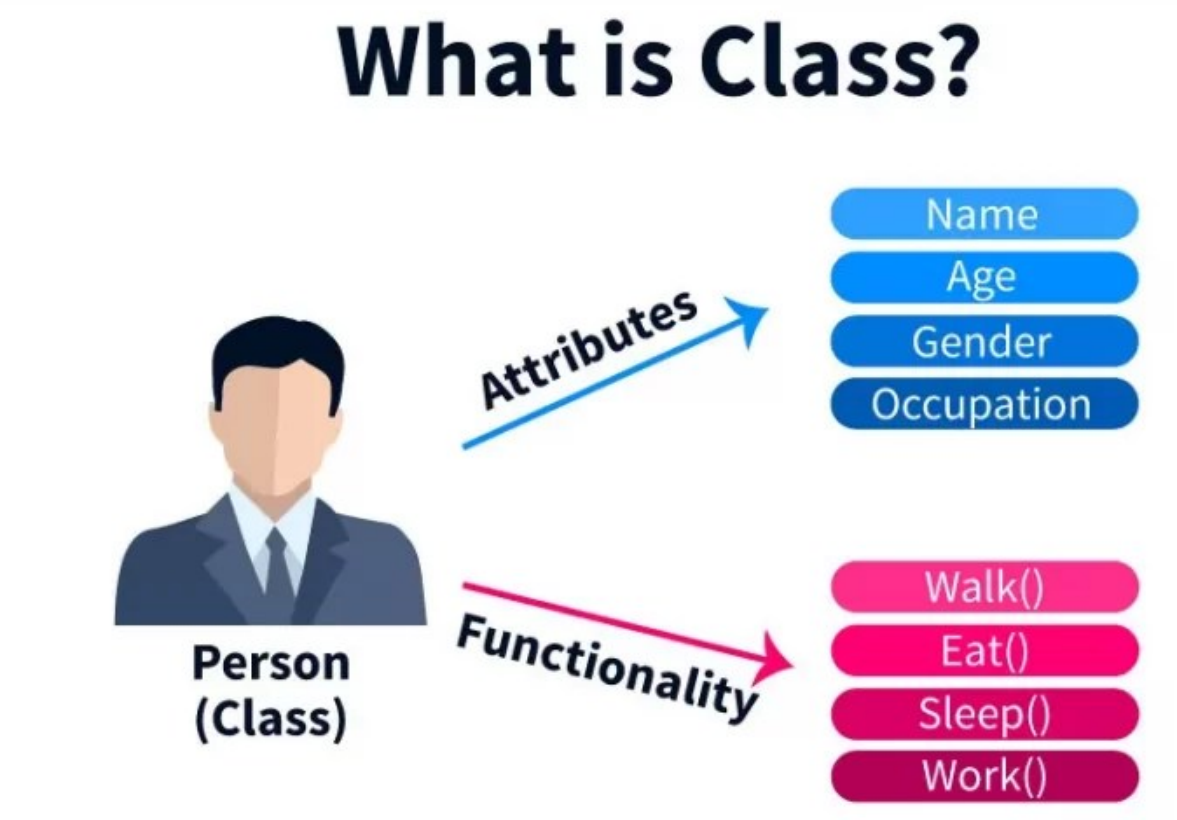
So, a class is a template for objects, and an object is an instance of a class. When the individual objects are created, they inherit all the variables and methods from the class.

Everything in C# is associated with **classes and objects**, along with its **attributes** and **methods**. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.

Class

In C#, a class is a fundamental concept in object-oriented programming (OOP) that serves as a blueprint for creating objects (**instances**). It defines the structure and behavior of objects.

- It is a way of creation user defined data.
- A class is a way to **bind** the **data** and its associated **function** together.
- Also, it allows the **data and function** to be **hidden** if necessary.



Access modifier

Class members, which include **attributes** (fields) and **methods**, can be specified with **access modifiers**, including the **public** and **private** keywords, among others. Access modifiers determine the visibility and accessibility of class members from outside the class. Here's a more detailed explanation:

1. **public**: Members declared as public are accessible from any part of the program, including other classes and assemblies. This means they have the broadest level of accessibility.
2. **private**: Members declared as private are only accessible within the class in which they are defined. They are not accessible from outside the class.

Hint* the **default access modifier** for class members, is **private**. This means that if you don't explicitly specify an access modifier, the member is considered private by default.

Creating a class

To create a class, use the class keyword:

```
class ClassName
{
    // Fields (data members or attributes)
    <access-modifier>  <data-type>  FieldName;

    // Methods (member methods)
    <access-modifier>  <return-type>  MethodName(parameters)
    {
        // Method code
    }
}
```

For example, to create a class **'Person'** with its members you can write this statement

```
class Person
{
    // Fields
    public string FirstName;
    public string LastName;
    private int age;

    // Method
    public void DisplayInfo()
    {
      Console.WriteLine($"Name: {FirstName} {LastName}, Age: {age}");
    }
}
```

This class contains three data members (**FirstName**, **LastName**, and **age**), and one member method (**DisplayInfo**).

All members are accessible inside the same class, while for the other classes (outside class 'person') only public members can be accessed.

Creating an object

Syntax:

```
Class_name Object_name = new Class_name( )
```

Now create an **object** to from the class **'Person'**, and try to access all members.

To create an object from the 'Person' class that was defined in the previous example, you can follow this syntax:

```
Person person1 = new Person();
```

Calling class members

```
namespace Class_and_Objects
{
   using System;

   class Person
   {
      public string FirstName; // accessable inside and outside the class
      public string LastName; // accessable inside and outside the class
      private int age; // accessable only inside the class
      public void DisplayInfo( )
      {
         Console.WriteLine($"Name: {FirstName} {LastName}, Age: {age}");
      }
   }
   class Program
   {
      static void Main( )
      {
         // Create a Person object
         Person person1 = new Person( );

         // Call the DisplayInfo method to display the information
         person1.DisplayInfo( );
      }
   }
}
```

Creating an instance of the class Person named **person1**

```
Name:  , Age: 0
```

Now try to assign values to all data member

```
namespace Class_and_Objects
{
    using System;

    class Person
    {
        public string FirstName; // accessible inside and outside the class
        public string LastName; // accessible inside and outside the class
        private int age; // accessible only inside the class
        public void DisplayInfo()
        {
          Console.WriteLine($"Name: {FirstName} {LastName},Age: {age}");
        }
    }
    class Program
    {
        static void Main()
        {
            // Create a Person object
            Person person1 = new Person();

            person1.FirstName = "Ayad";
            person1.LastName = "Abdulrahman";
            person1.age = 27;

            // Call the DisplayInfo method to display the information
            person1.DisplayInfo();
        }
    }
}
```

Error: Inaccessible due to its protection level (**private**)

To allow age to be accessible, just change the access modifier from **private** to **public**.

```
namespace Class_and_Objects
{
    using System;

    class Person
    {
        public string FirstName; // accessible inside and outside the class
        public string LastName; // accessible inside and outside the class
        public int age; // accessible inside and outside the class
        public void DisplayInfo()
        {
            Console.WriteLine($"Name: {FirstName} {LastName}, Age: {age}");
        }
    }
    class Program
    {
        static void Main()
        {
            // Create a Person object
            Person person1 = new Person();
```

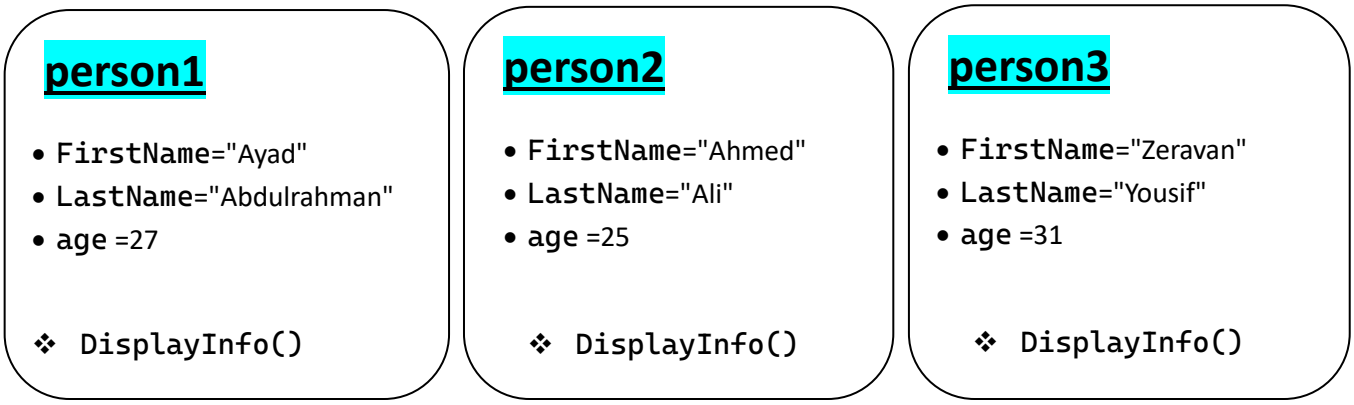
```
            person1.FirstName = "Ayad";
            person1.LastName = "Abdulrahman";
            person1.age = 27;

            // Call the DisplayInfo method to display the information
            person1.DisplayInfo();
        }
    }
}
```

```
Name: Ayad Abdulrahman, Age: 27
```

Now create more than one object (**person1**, **person2**, and **person3**) and assign value to each of them.

```
namespace Class_and_Objects
{
    using System;

    class Person
    {
        public string FirstName; // accessible inside and outside the class
        public string LastName; // accessible inside and outside the class
        public int age; // accessible inside and outside the class
        public void DisplayInfo()
        {
            Console.WriteLine($"Name: {FirstName} {LastName}, Age: {age}");
        }
    }
    class Program
    {
        static void Main()
        {
            Person person1 = new Person();// an instance (object)
            person1.FirstName = "Ayad";
            person1.LastName = "Abdulrahman";
            person1.age = 27;

            Person person2 = new Person();  // an instance (object)
            person2.FirstName = "Ahmed";
            person2.LastName = "Ali";
            person2.age = 25;

            Person person3 = new Person(); // an instance (object)
            person3.FirstName = "Zeravan";
            person3.LastName = "Yousif";
            person3.age = 31;

            person1.DisplayInfo();
            person2.DisplayInfo();
            person3.DisplayInfo();
        }
    }
}
```

```
Name: Ayad Abdulrahman, Age: 27
Name: Ahmed Ali, Age: 25
Name: Zeravan Yousif, Age: 31
```



Three independent instances were created.

Array of objects

You can create a list of objects in the same way you create an array

```
namespace Class_and_Objects
{
    using System;

    class Person
    {
        public string FirstName; // accessible inside and outside the class
        public string LastName; // accessible inside and outside the class
        public int age; // accessible only inside the class
        public void DisplayInfo()
        {
            Console.WriteLine($"Name: {FirstName} {LastName}, Age: {age}");
        }
    }
    class Program
    {
        static void Main()
        {
            Person[] per = new Person[3];
            for (int i = 0; i < per.Length; i++)
            {
                per[i] = new Person();
                Console.Write("please enter first name: ");
                per[i].FirstName = Console.ReadLine();
                Console.Write("please enter last name: ");
                per[i].LastName = Console.ReadLine();
                Console.Write("please enter age :");
                per[i].age=int.Parse( Console.ReadLine() );
                Console.WriteLine("_________________________");
            }
            Console.WriteLine("all person information are");
            foreach (Person item in per)
            {
                item.DisplayInfo();
            }
            Console.WriteLine( );
```

```
        }
    }
}
```

per[0]

- FirstName="Ali"
- LastName="Kawa"
- age =22

❖ DisplayInfo()

per[1]

- FirstName="Ayad"
- LastName="Abdulrahman"
- age =33

❖ DisplayInfo()

per[2]

- FirstName="Rami"
- LastName="Ahmed"
- age =19

❖ DisplayInfo()

```
please enter first name: Ali
please enter last name: Kawa
please enter age :22
_____________________________
please enter first name: Ayad
please enter last name: Abdulrahman
please enter age :52
_____________________________
please enter first name: Rami
please enter last name: Ahmed
please enter age :19
_____________________________
all person information are
Name: Ali Kawa, Age: 22
Name: Ayad Abdulrahman, Age: 52
Name: Rami Ahmed, Age: 19
```

Practical Examples

Example-1:

- Define a class named "**Student**" with the following members:
  - Data members (fields):
    - **Full_name**
    - **Gender**
    - **Address**
    - **Age**
    - **Marks** (three marks)
    - **Average** : the value of this variable should be calculated based on student marks
  - Member methods
    - **setInfo**: to set the student information (The information should be passed from **Main()** method).
    - **displayInfo**: to display the student information including Full_Name, Gender, Address, Age, and Average.
- In the Main( ) program, create an instance (object) of class **Student**. Then:
  1. Set student information.
  2. Print student information.

Example-2:

- Upgrade the previous example (**Example-1**) by defining another class named "**Employee**", containing the following members:
  - Data members (fields):
    - **Full_name**
    - **Gender**
    - **Address**
    - **Age**
    - **Salary**
  - Member methods
    - **setInfo**: to set the employee information
    - **displayInfo**: to display the employee information including Full_Name, Gender, Address, Age, and Salary.
- In the Main( ) program, create an instance (object) of class **Employee** and then invoke (call) the two created methods.

Example 3- Create an object

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);
  }
}
```

Example 4:- Multiple class

You can create multiple objects of one class:Create two objects of `Car`:

```
using System;
namespace app55
{
    class Car
    {
        public string color = "red";
    }

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

}
```

Example 5- Class Members

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

```
using System;
namespace app55
{
    class Car
    {
        string color;
        int maxSpeed;

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

}
```

Output:

```
red
200
```

Example 6- Multiple Object

```
using System;
namespace app55
{
    class Car
    {
        string model;
        string color;
        int year;

        static void Main(string[] args)
        {
            Car Ford = new Car();
            Ford.model = "Mustang";
            Ford.color = "red";
            Ford.year = 1969;

            Car Opel = new Car();
            Opel.model = "Astra";
            Opel.color = "white";
            Opel.year = 2005;

            Console.WriteLine(Ford.model);
            Console.WriteLine(Opel.model);
            Console.ReadKey();
        }
    }
}
```

Output

```
Mustang
Astra
```

Example 7 - Use Multiple Classes

```
using System;
namespace app55
{
    class Car
    {
        public string model;
        public string color;
        public int year;
        public void fullThrottle()
        {
         Console.WriteLine("The car is going as fast as it can! - year = {0}",year);
        }
   }


    class Program
    {
        static void Main(string[] args)
        {
            Car Ford = new Car();
            Ford.model = "Mustang";
            Ford.color = "red";
            Ford.year = 1969;

            Car Opel = new Car();
            Opel.model = "Astra";
            Opel.color = "white";
            Opel.year = 2005;

            Console.WriteLine(Ford.model);
            Console.WriteLine(Opel.model);
            Ford.fullThrottle();
            Opel.fullThrottle();
            Console.ReadKey();
        }
    }

}
```

Output:-

```
Mustang
Astra
The car is going as fast as it can! - year = 1969
The car is going as fast as it can! - year = 2005
```