# OOPS CONCEPT

Object Oriented Programming (OOP) in C#. OOPs is a concept of modern programming language that allows programmers to organize entities and objects. Four key concepts of OOPs are abstraction, encapsulation, inheritance, and polymorphism. Here learn how to implement OOP concepts in C# and .NET.

OOP Features

Here are the key features of OOP:

- Object Oriented Programming (OOP) is a programming model where programs are organized around objects and data rather than action and logic.
- OOP allows decomposing a problem into many entities called objects and then building data and functions around these objects.
- A class is the core of any modern object-oriented programming language such as C#.
- In OOP languages, creating a class for representing data is mandatory.
- A class is a blueprint of an object that contains variables for storing data and functions to perform operations on the data.
- A class will not occupy any memory space; hence, it is only a logical representation of data.

To create a class, you use the keyword "class" followed by the class name:

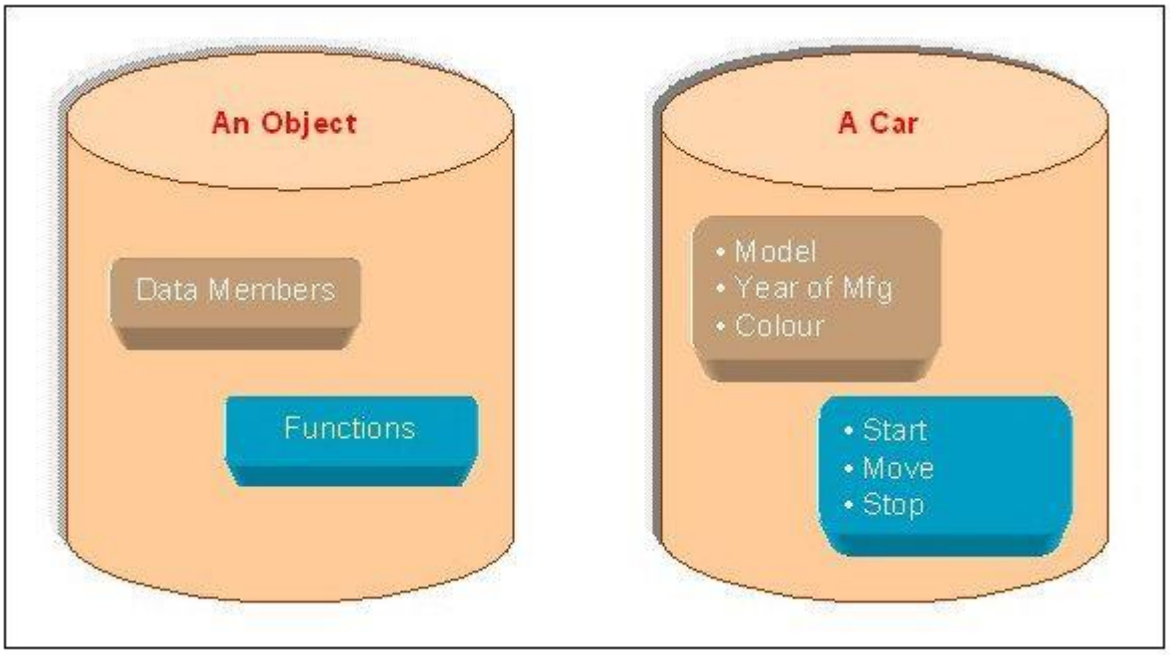
```
class Employee
{


}
```

Object

1. The software is divided into several small units called objects. The data and functions are built around these objects.
2. An object's data can be accessed only by the functions associated with that object.
3. The functions of one object can access the functions of another object.

Objects are the basic run-time entities of an object-oriented system. They may represent a person, a place, or any item the program must handle.

- "An object is a software bundle of related variables and methods."
- "An object is an instance of a class."




A class will not occupy any memory space. Hence to work with the data represented by the class, you must create a variable for the class, which is called an object.

- When an object is created using the new operator, memory is allocated for the class in a heap, the object is called an instance, and its starting address will be stored in the object in stack memory.
- When an object is created without the new operator, memory will not be allocated in a heap; in other words, an instance will not be created, and the object in the stack will contain the value **null**.
- When an object contains null, it is impossible to access the class members using that object.

```
class Employee
{


}
```

Syntax to create an object of class Employee:

```
Employee objEmp = new Employee();
```

OOPs Concepts

## The key concepts of OOPs are

1. **Abstraction**
2. **Encapsulation**

3. **Inheritance**
4. **Polymorphism**

# Abstraction

Abstraction is "To represent the essential feature without representing the background details."

- Abstraction lets you focus on what the object does instead of how it does it.
- Abstraction provides a generalized view of your classes or objects by providing relevant information.
- Abstraction is the process of hiding the working style of an object and showing the information about an object understandably.

**Real-world Example of Abstraction**

Suppose you have an object Mobile Phone.

Suppose you have three mobile phones as in the following:

- Nokia 1400 (Features: Calling, SMS)
- Nokia 2700 (Features: Calling, SMS, FM Radio, MP3, Camera)
- BlackBerry (Features: Calling, SMS, FM Radio, MP3, Camera, Video Recording, Reading E-mails)

Abstract information (necessary and common information) for the "Mobile Phone" object is that it calls any number and can send an SMS.

So, for a mobile phone object, you will have the abstract class as in the following,

```
abstract class MobilePhone {
   public void Calling();
   public void SendSMS();
}
public class Nokia1400: MobilePhone {}
public class Nokia2700: MobilePhone {
   public void FMRadio();
   public void MP3();
   public void Camera();
}
public class BlackBerry: MobilePhone {
   public void FMRadio();
   public void MP3();
```

```
    public void Camera();

    public void Recording();

    public void ReadAndSendEmails();
}
```

Abstraction means putting all the necessary variables and methods in a class.

**Example**

If somebody in your college tells you to fill in an application form, you will provide your details, like name, address, date of birth, which semester, the percentage you have, etcetera.

If some doctor gives you an application, you will provide the details, like name, address, date of birth, blood group, height, and weight.

See in the preceding example what is in common?

Age, name, and address, so you can create a class that consists of the common data. That is called an abstract class.

That class is not complete, and other classes can inherit it.

# Encapsulation

Wrapping up a data member and a method together into a single unit (in other words, class) is called Encapsulation. Encapsulation is like enclosing in a capsule. That is, enclosing the related operations and data related to an object into that object.

Encapsulation is like your bag in which you can keep your pen, book, etcetera. It means this is the property of encapsulating members and functions.

```
class Bag {

    book;

    pen;

    ReadBook();
}
```

- Encapsulation means hiding the internal details of an object, in other words, how an object does something.
- Encapsulation prevents clients from seeing its inside view, where the behavior of the abstraction is implemented.
- Encapsulation is a technique used to protect the information in an object from another object.
- Hide the data for security, such as making the variables private, and expose the property to access the private data that will be public.

So, when you access the property, you can validate the data and set it.

Example 1

```
class Demo {
   private int _mark;
   public int Mark {
      get {
         return _mark;
      }
      set {
         if (_mark > 0) _mark = value;
         else _mark = 0;
      }
   }
}
```

# Inheritance

When a class includes a property of another class, it is known as inheritance. Inheritance is a process of object reusability.

For example, a child includes the properties of its parents.

```
public class ParentClass {
   public ParentClass() {
      Console.WriteLine("Parent Constructor.");
   }
   public void print() {
      Console.WriteLine("I'm a Parent Class.");
   }
}
public class ChildClass: ParentClass {
   public ChildClass() {
      Console.WriteLine("Child Constructor.");
   }
```

```
    public static void Main() {
        ChildClass child = new ChildClass();
        child.print();
    }
}
```

**Output**

*Parent Constructor.*
*Child Constructor.*
*I'm a Parent Class.*

# Polymorphism

Polymorphism means one name, many forms. One function behaves in different forms. In other words, "Many forms of a single object is called Polymorphism."

**Real-world Example of Polymorphism**

**Example 1**

A teacher behaves with his students.

A teacher behaves with their seniors.

Here the teacher is an object, but the attitude is different in different situations.

**Example 2**

A person behaves like a son in a house at the same time that the person behaves like an employee in an office.

Differences between Abstraction and Encapsulation

| | |
|---|---|
| 1. Abstraction solves the problem at the design level. | 1. Encapsulation solves the problem at the implementation |
| 2. Abstraction hides unwanted data and provides relevant data. | 2. Encapsulation means hiding the code and data in a single data from the outside world. |
| 3. Abstraction lets you focus on what the object does instead of how it does it | 3. Encapsulation means hiding the internal details or mecha object does something. |
| 4. Abstraction: Outer layout, used in terms of design.<br>For example:<br>An external Mobile Phone like it has a display screen and keypad buttons to dial a number. | 4. Encapsulation- Inner layout, used in terms of implementa<br>For example, the internal details of a Mobile Phone and how and display screen are connected using circuits. |