WHAT IS OOP?

Object Oriented Programming (OOP) is one of the most popular programming languages. This article is an introduction to Object Oriented Programming (OOP) and how to implement OOP in C# including abstraction, encapsulation, inheritance and polymorphism.

FEATURES

Object Oriented Programming (OOP) is a programming model where programs are organized around objects and data rather than action and logic.

OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects.

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

OOP has the following important features.

CLASSES

A class is the core of any modern Object Oriented Programming language such as C#.

In OOP languages it is mandatory to create a class for representing data.

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 and hence it is only a logical representation of data.

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

class Employee

{

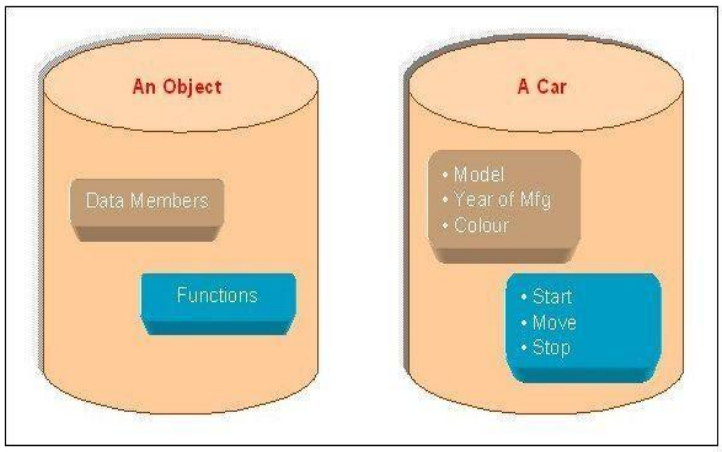
}

Object

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

"An object is a software bundle of related variable 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, that is called an object.
- When an object is created using the new operator, memory is allocated for the class in the 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 the heap, in other words an instance will not be created and the object in the stack contains the value **null**.
- When an object contains null, then it is not possible to access the members of the class using that object.

```
class Employee
{
}
```

- Syntax to create an object of class Employee:

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

All the programming languages supporting Object Oriented Programming will be supporting these three main concepts,

- Encapsulation
- Inheritance
- 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 you 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 of an object in an understandable manner.

- **Real-world Example of Abstraction**
- Suppose you have an object Mobile Phone.

  Suppose you have 3 mobile phones as in the following:

  Nokia 1400 (Features: Calling, SMS)
  Nokia 2700 (Features: Calling, SMS, FM Radio, MP3, Camera)
  Black Berry (Features:Calling, SMS, FM Radio, MP3, Camera, Video Recording, Reading Emails)

  Abstract information (necessary and common information) for the object "Mobile Phone" is that it makes a call to any number and can send SMS.

- So that, 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 variables and methods in a class that are necessary.
- For example: Abstract class and abstract method.
- Abstraction is a common thing.

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, percentage you have etcetera.
- If some doctor gives you an application to fill in the details, 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 it can be inherited by other classes.

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 it's 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

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

```
    }
  }
}
```

Real-world Example of Encapsulation

- Let's use as an example Mobile Phones and Mobile Phone Manufacturers.
- Suppose you are a Mobile Phone Manufacturer and you have designed and developed a Mobile Phone design (a class). Now by using machinery you are manufacturing Mobile Phones (objects) for selling, when you sell your Mobile Phone the user only learns how to use the Mobile Phone but not how the Mobile Phone works.

  This means that you are creating the class with functions and with objects (capsules) of which you are making available the functionality of your class by that object and without the interference in the original class.

Example 2

- **TV operation**
- It is encapsulated with a cover and we can operate it with a remote and there is no need to open the TV to change the channel.
  Here everything is private except the remote, so that anyone can access the remote to operate and change the things in the TV.

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.

Example 1

- **Real-world Example of Polymorphism**

A teacher behaves students.

A teacher behaves with his/her seniors.

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

- **Example 2**
- A person behaves the son in a house at the same time that the person behaves as an employee in an office.
- **Example 3**
- Your mobile phone, one name but many forms:
- As phone
- As camera
- As mp3 player
- As radio

Differences between Abstraction and Encapsulation

Abstraction	Encapsulation
1. Abstraction solves the problem at the design level.	1. Encapsulation solves the problem in the implementation level.
2. Abstraction hides unwanted data and provides relevant data.	2. Encapsulation means hiding the code and data into a single unit to protect the 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 mechanics of how an object does something.
4. Abstraction: Outer layout, used in terms of design. For example: An external of a Mobile Phone, like it has a display screen and keypad buttons to dial a number.	4. Encapsulation- Inner layout, used in terms of implementation. For example: the internal details of a Mobile Phone, how the keypad button and display screen are connected with each other using circuits.

Real-world Example

- Use an example of a Mobile Phone

You have a Mobile Phone; you can dial a number using keypad buttons. You don't even know how these are working internally. This is called Abstraction. You only have the information that is necessary to dial a number. But not the internal working of the mobile.

But how does the Mobile Phone work internally? How are the keypad buttons connected to the internal circuit? That is called Encapsulation.