

Object Based Programming

OOP Concepts

Object Based Programming

OOP	Structured Programming
<ul style="list-style-type: none"> Based on Object oriented. Program consists of a set of classes. Collect variables (data) with a set of related methods. 	<ul style="list-style-type: none"> Based on Action Oriented. Program consist of : a set of method , each of which perform a specific function . Separated data from method.
<p>OOP based on four basic concepts :</p> <ol style="list-style-type: none"> 1. Encapsulation. 2. Information Hiding. 3. Inheritance. 4. Polymorphism. 	
<p>Encapsulation :</p> <p>الكبسلة :</p> <p>عملية تجميع البيانات Variables والدوال methods في عنصر واحد يحتوي علي جميع البيانات والدوال الخاصة بتنفيذ مهمة معينة ، وهذا العنصر يسمى بال Class .</p> <p>✓ نستطيع من مفهوم الكبسلة تحقيق مبدأ اخفاء البيانات Data Hiding</p>	
<p>Information Hiding :hide implementation from object .</p> <p>اخفاء البيانات :</p> <p>○ اخفاء البيانات عن المستخدم لحماية الكائن من تغيرات غير مرغوبة .</p> <p>✓ اخفاء تفاصيل برمجة ال Class عن المستخدم بحيث يتم التعامل مع ال Class من خلال واجهه تخفي التفاصيل .</p>	
<p>Inheritance :</p> <p>✓ you can derive a new class that inherit all class features (members data and functions) in addition to adding new features to the derived class .</p> <p>الوراثة :</p> <p>○ عملية اشتقاق Class جديدة من Class اكثر عموميه لتحتوي بذلك علي مميزاتها .</p> <p>✓ تعتمد فكرة الوراثة علي اعادة استخدام ال Classes الموجودة لبناء Classes جديدة دون الحاجة الي اعادة كتابة الكود الذي يحتوي علي التفاصيل المتشابهة .</p>	

تعدد الاشكال :

- ### Method :

- It can be public or private :**

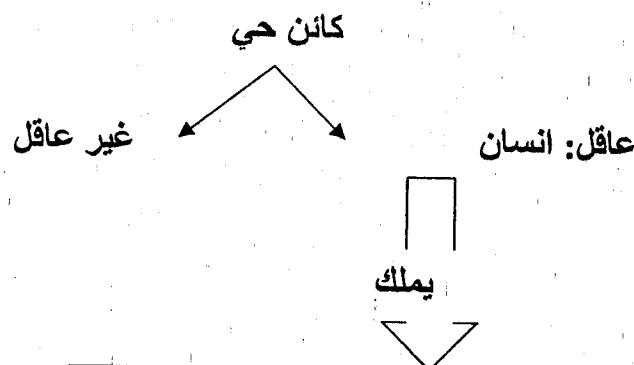
- ⊕ **Predicate Method** : a common use of access method that check the truth of of condition , such as : `isFull ()` : test if object is full or not .

⊕ Private method. >>> **Helper Method : (Utility Method)**

- A private method within class, called only by other methods of that class.
- The purpose of is to support the operation of class's other methods.

Overload :

- يسمح لل Method للدوال بأن تكون لها أسماء متشابهة ولكن ببارامتر مختلفة .



<p>وظائف ، سلوك :</p> <p>Methods</p> <p>يأكل يشرب ينام</p>	<p>خصائص :</p> <p>Variables</p> <p>اسم عمر لون</p>
---	---

انسان يعتبر : فصيلة Class والذي يحتوي بداخله علي :

احمد ، محمد ، محمود :
افراد من الفصيلا ، ويطلق
عليهم Object .

عتمادا على مبدأ الفصائل :

دل ان نقول ان احمد :

ملك : اسم - عمر - لون ويستطيع ان : يأكل - يشرب - ينام .
كذلك محمد :

ملك : اسم - عمر - لون ويستطيع ان : يأكل - يشرب - ينام .
كذلك محمود :

ملك : اسم - عمر - لون ويستطيع ان : ياكل - يشرب - ينام .

مکن ان نقول :

حمد ومحمدو محمود انسان ، والانسان :

Class	Object
<p>A class is just a collection of variables (often with different types) combined with a set of related methods.</p> <p>صنف : قالب يتم من خلاله انشاء كائنات مختلفة او متشابهة</p>	<p>An <i>object</i> is an individual instance of a class.</p> <p>كائن : مجموعة من الدوال والبيانات التي تعمل في اطار موحد لتشكل شيئا مترابط ومتناسق .</p>
<p>○ الخصائص العامة لفئة معينة من بيانات ووظائف يتم تحديدها داخل ال Class .</p> <p>○ لكن قيم الافراد يتم تخصيصها لل Object .</p> <p>○ لذلك نقول اننا نحتاج الي Class واحد ، ولكن نحتاج الي اكثر من Object .</p>	
<pre>class class_name { // class variables and methods declared here }</pre>	<p>Class_name Object = new Class_name() ;</p> <p>ويتم تخليق ال Object اي كتابة هذا السطر داخل ال .main method</p>
<ul style="list-style-type: none"> • Member variables are known as data member; the variables in your class that should be set as private. • Static members belongs to the class itself & object don't have access its value . • Member functions are known as methods, the function in your class that determines what the objects of your class can do & must be set as public. 	
Private	Public
<p><i>Private</i> data member can be accessed only within methods of the class itself</p>	<ul style="list-style-type: none"> • <i>Public</i> data member can be accessed through any object of the class .
<ul style="list-style-type: none"> • All members of a class--data and methods--are private by default. 	
<p>Protected</p> <p><i>protected</i> data member can be accessed directly by object of any inherited class .</p>	

<pre> Class Human { Public int age ; Public string name ; Public void speak () { Console.WriteLine ("Human speak "); } } </pre>	<pre> ClassName ObjectName = new ClassName() ; Human h1 = new Human () ; ObjectName . VarName = Value ; h1. age = 22 ; h1.name="a7md"; . Method الاستدعاء ال ObjectName . MethodName() ; h1.speak() ; </pre>
--	---

Constructor	vs.	Destructor
<ul style="list-style-type: none"> ➤ <i>Constructor</i> create and initialize objects of your class, ➤ <i>Constructor</i> class method with the same name as the class itself. ➤ <i>Constructor</i> can take parameters as needed, it cannot have a return value, not even void. 		<ul style="list-style-type: none"> ➤ <i>Destructors</i> clean up and free any memory you might have allocated. ➤ <i>Destructor</i> class method with the same name as the class , preceded by a tilde (~). ➤ <i>Destructor</i> take no arguments & have no return value .

- Method داخل ال Class تستخدم في اعطاء قيم ابتدائية للمتغيرات المعرفة في ال Class
- ولكن ال Constructor يتميز عن باقي ال Method في ان لها :
- نفس اسم ال class .
 - لا ترجع قيمة ولا حتي void .

<pre> Human () { } </pre>	<p>Constructor Destructor</p> <p>← →</p>	<pre> ~ Human () { } </pre>
----------------------------	--	------------------------------

```
class class_name
```

```
{
// class variables and
methods declared
here
}
```

عمل initialization
لقيم المتغيرات ، فاننا نحتاج

الي Constructor
بمعني اعطاء المتغيرات قيم
ابتدائية عند استخدامها .

```
class class_name
```

```
{
//class variables & methods here
ClassName ( parameters)
{
Var = value ;
}
}
```

```
Public Human ()
```

```
{
}
```

ال Constructor can be overloaded بمعنى :
يمكن عمل اكثر من داله لها اسم ال Class ولكنها تختلف في عدد المعاملات .

تعريف ثم :
استدعاء

```
Public Human ()
```

```
{
}
```

```
Public Human (int a)
```

```
{
age=a ;
}
```

```
Public Human (int a,
string b)
```

```
{
age = a ;
name = b ;
}
```

```
Human h1 = new Human (22);
Console.WriteLine("age"+h1.age) ;
```

```
Human h2 = new Human (30,"ahmed");
Console.WriteLine("age"+h2.age+"name"+h2.name) ;;
```

نجد ان age:22

نجد ان age:30 و ال name : ahmed

```
h1= null ;
h2= null ;
System.GC.Collect ( );
```

بعد انتهاء العمل مع ال Class فاننا نريد تدمير ه ، بمعني ازالته من ال
Memory حفاظا علي مساحة الذاكرة لذلك نقوم بعمل
ومن ثم يتم استدعاء ال Destructor بصورة تلقائية .

تعرفنا علي كيفية اعطاء قيم ابتدائية للمتغيرات ، وذلك بواسطة ال Programmer . نريد التعرف علي إمكانية ادخال ال user قيم للمتغيرات لاستخدامها لاحقا .

Properties	
get accessor	set accessor
<ul style="list-style-type: none"> ○ Enables objects to read data . ○ Obtain the values of private data 	<ul style="list-style-type: none"> ○ Enables objects to modify data ○ Assign values to private data. ○ Ensure that the new value is appropriate for the data member .

```
// property Hour
public int Hour
{
    get
    {
        return hour;
    }

    set
    {
        hour = value;
    }
}
```

Write a program that declare a class called player . class should contain three private variables: age, weight, and speed . The class should contain a constructor that initializes the three data members. The class should contain variable that keep track of number of players, then Define a player object, assign 23 to age, and output speed .

Using system;

namespace playerdesign

Class player

```
{
    Private int age ;
    Private int speed ;
    Private int weight ;
```

```
Public Static int count ;
```

```
Public player(int age , int speed, int weight )
```

```
{  
this.age= age;  
this.speed=speed;  
this.weight=weight;  
this.count++;  
}
```

```
Public int Age
```

```
{  
set  
{  
This.age=value;  
}  
}
```

```
Public int Speed
```

```
{  
get  
{  
return speed ;  
}  
}
```

```
}// End class player
```

```
Class test
```

```
{  
Static void main(string [] args)  
{  
Player p1=new player( 20,30,40) ;  
P1.Age=23 ;  
Console.WriteLine("player's speed is :"+p1.Speed )  
}//End main method  
}//End class test  
}//End namespace
```


⊕ Create a class called Complex for performing arithmetic +with complex numbers. Write a driver program to test your class .

Use floating-point variables to represent the private data of the class. Provide constructor that enables an object of this class to be initialized when it is declared. Provide a no-argument constructor with a default values in case no initializers are provided. Provide public methods for each of the following :

- a) Addition of two Complex numbers. The real parts are added together and the imaginary parts are added together .
- b) Subtraction of two Complex numbers. The real parts of the right operand is subtracted from the real part of the left operand and the imaginary part of the right operand is subtracted from the imaginary part of the left operand.
- c) Printing of Complex numbers in the form (a,b) , where a is the real; part and b is the imaginary part .

Class Complex

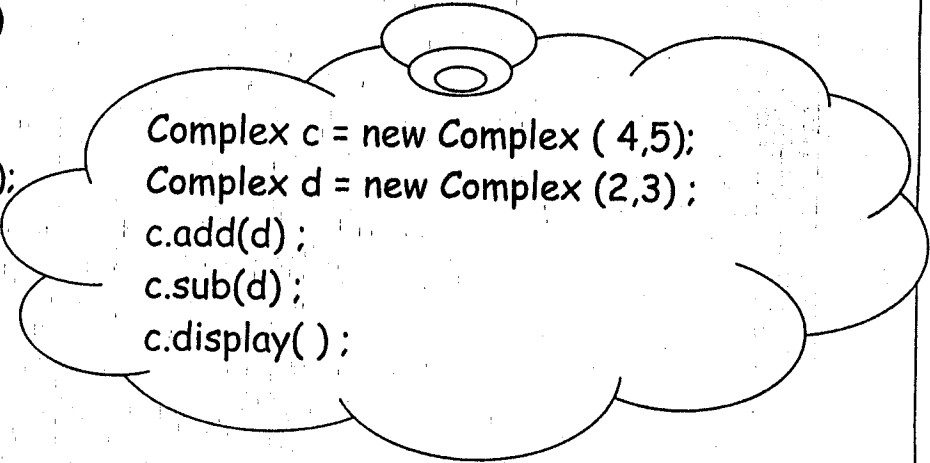
```
{  
Private float realPart,  
Private float imaginaryPart,  
  
//Constructor : default , with no-argument  
Public Complex()  
{  
}  
  
//Constructor, initialize values  
Public Complex( float real, float imaginary)  
{  
this. realPart = real;  
this. imaginaryPart = imaginary ;  
}
```

```
Public complex add (complex c)
{
this.realpart+=c.realpart ;
this.imaginarypart+=c.imaginary ;
}
Public complex sub (complex c)
{
this.realpart -= c.realpart ;
this.imaginarypart -=c.imaginary ;
}

Public string display ()
{
Console.WriteLine( "("+this.realpart+", "+this.imaginarypart+" )" );
}

Static void main(string [] args)
{
Complex c = new Complex ( 4,5);
c.add(new Complex ( 2,3)) ;
c.sub(new Complex ( 2,3)) ;
c.display( ) ;
} //End main method

} // End class Complex
```



```
Complex c = new Complex ( 4,5);
Complex d = new Complex (2,3) ;
c.add(d) ;
c.sub(d) ;
c.display( ) ;
```