**object-oriented programming (OOP)**

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them.

The organization of an object-oriented program also makes the method beneficial to collaborative development, where projects are divided into groups. Additional benefits of OOP include code reusability, scalability and efficiency.

ملاحظة :

تعمل مع كل شي او كل كائن لواحدة مثل لعبة الشطرنج كل كائن له بيانات خاص به مثل المصفوفات فقط تحفظ نوع واحد من البيانات سوى رقمية او رموز فقط لكن بعد يأتي

و الذي ممكن نحفظ في اكثر من نوع مع بعض لكن (Structure)

لا يمكن ان نعرف (وظائف) داخل (الاستركشر) مثل حركة الفيل في رقعه الشطرنج

### **What is the structure of object-oriented programming?**

The structure, or building blocks, of object-oriented programming include the following:

* **Classes** are Template of objects.(الفئة هي المحتوى للكائنات )

**Ex:**

**Class** Feel:

**Position**

**Color**

**DeadOrNot**

**Move()**:

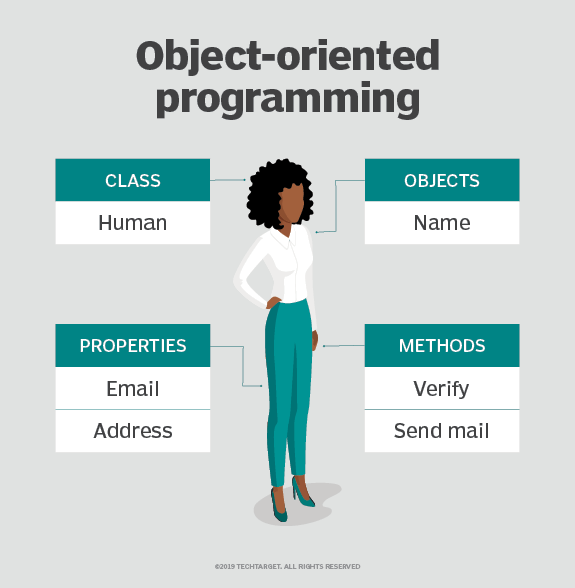
**Return newPosition**

* **Objects** are instances of a class .(الكائن هو نموذج للفئة)

**Feel feel1=new Feel(3,"red",False)**

### **Feel feel2=new Feel(-3,"wh** ",false)

* **Methods** are functions that are defined inside a class that describe the behaviors of an object. Each method contained in class definitions starts with a reference to an instance object.

**Attributes** are defined in the class template and represent the state of an object. Objects will have data stored in the attributes field. Class attributes belong to the class itself. 

### What are the main principles of OOP?

Object-oriented programming is based on the following principles:

* **Encapsulation.**Data and methods in single class

**Ex:**

Class {

Data members

+

Methods(behavior)

}

------------------------------------------------------------

class User

{

private string location;

private string name;

public string Location

{

get

{

return location;

}

set

{

location = value;

}

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

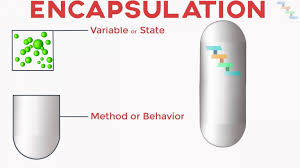
}

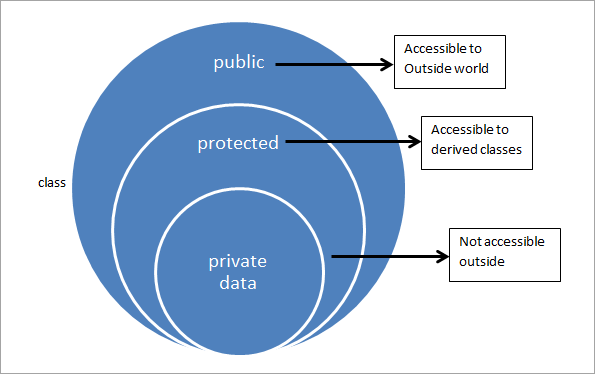
}

 class Program  
    {  
       static void Main(string[] args)  
       {  
          User u = new User();  
          // set accessor will invoke  
          u.Name = "Suresh Dasari";  
          // set accessor will invoke  
          u.Location = "Hyderabad";  
          // get accessor will invoke  
          Console.WriteLine("Name: " + u.Name);  
          // get accessor will invoke  
          Console.WriteLine("Location: " + u.Location);  
          Console.WriteLine("\nPress Enter Key to Exit..");  
          Console.ReadLine();  
       }  
    }  
}

ملاحظة :

يمكن ان اخفى البيانات والوظائف لاستخدمه في البرنامج من غير ربطها باي كلاس اخر في البرنامج ,مثل التلفازيون .



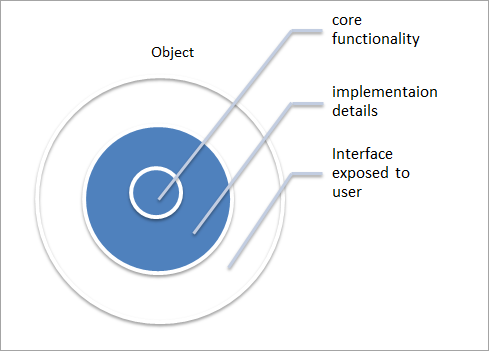


**Encapsulation vs Abstraction**

The difference between encapsulation and abstraction is as follows.

| **Encapsulation** | **Abstraction** |
| --- | --- |
| In encapsulation, we focus on grouping the properties and methods of the object together inside a single unit. | In abstraction, we focus on hiding the complex methods and only showing the essential things to the user. |
| It makes code more modular and easy to understand | It makes applications easy to use by hiding underlying complex working. |
| We provide security to the properties and methods by deciding who can access them. | We provide security to the application by hiding the working part from the user. |
| Encapsulation is about hiding the information. | Abstraction is about hiding the implementation. or internal working. |

**Abstraction.** Objects only reveal internal mechanisms that are relevant for the use of other objects,



using System;

namespace MyApplication

{

// Abstract class

abstract class Animal

{

// Abstract method (does not have a body)

public abstract void animalSound();

// Regular method

public void sleep()

{

Console.WriteLine("Zzz");

}

}

// Derived class (inherit from Animal)

class Pig : Animal

{

public override void animalSound()

{

// The body of animalSound() is provided here

Console.WriteLine("The pig says: wee wee");

}

}

class Program

{

static void Main(string[] args)

{

Pig myPig = new Pig(); // Create a Pig object

myPig.animalSound();

myPig.sleep();

}

}

}



## **Inheritance** :

## is used to inherit the properties from one class (base) to another (child) class.