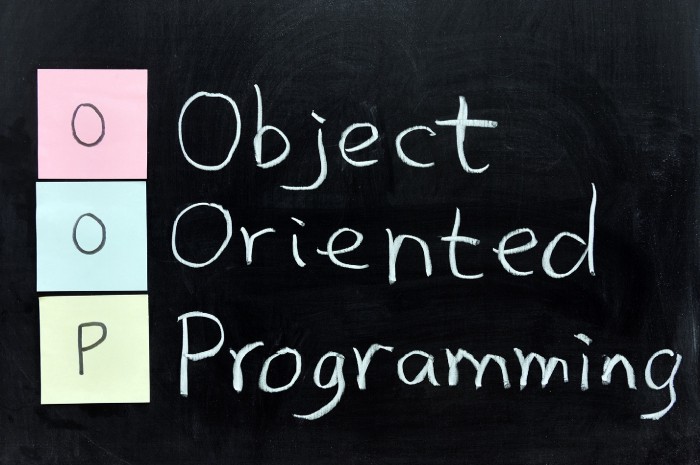
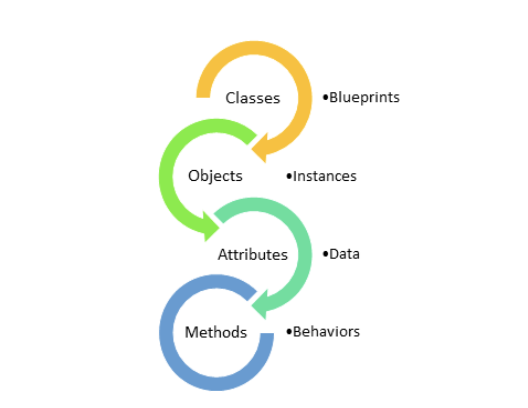
OOP-nin əsasları



Obyekt yönümlü proqramlaşdırma (Object Oriented Programming) təkrar kodların qarşısını alaraq böyük prosesləri kiçik hissələrə bölüb həll etməyin yollarını müəyyən edən paradiqmadır.Bu paradiqmanın köməyi ilə kod yazma prosesi daha sadə və asan olur.

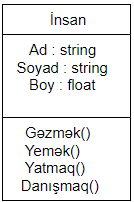
OOP dörd hissədən ibarətdir:

1. Sinif(class)
2. Obyekt(object)
3. Atributlar(atributes, fields, properties)
4. Metodlar(Methods, functions)



Obyekt yönümlü proqramlaşdırmanın (Object Oriented Programming)əsas məqsədi yuxarıda yazıldığı kimi böyük problemləri kiçik hissələrə(class) bölərək həmin problemi həll etməkdir. Class’lar da bu prosesi etmək üçün bizə lazım olan alətlərdir. Başqa sözlə desək class bütün qəlibin infrastrukturunu özündə saxlayan alətdir. Class’lar əsasən iki hissədən ibarətdir: atributlar (fields, properties) və metodlar (funksiyalar).

Bunu nümunə üzərindən başa düşməyə çalışaq.



Burada “İnsan” class’ın adı, “ad”, “soyad”, “boy” atributlar (fields), “Gəzmək()”, ”Yemək()”, ”Yatmaq()”, “Danışmaq()” isə metodlar və ya funksiyalar adlanır. Aşağıda class nümunəsi göstərilmişdir.

public class İnsan { public string ad;  
 public string soyad;  
 public int yaş;public void Gəzmək(){  
 //....  
 }  
public void Yemək(){  
 //....  
 }  
public void Yatmaq(){  
 //....  
 }}

**Public** giriş göstəricisi(**access modifier**) adlanır, burada göstərir ki, bu metodu başqa class’lar görə və ya istifadə edə bilər.

Obyekt yaddaşda olan class’ın bir nümunəsidir. Class-obyekt əlaqəsinə belə bir nümunə verilə bilər: İnsan bir class’dır, bu class’a daxil olan hər şey isə bir obyektdir. Nümunəyə baxsaq burada bizim obyektimiz hər hansı bir ad “Əli”, soyad “Həsənov” və ya yaş 13 ola bilər.

static void Main(string[] args)  
 {  
 İnsan insan = new İnsan();   
 ad="Əli";   
 soyad="Həsənov";   
 yaş=13;  
 }}

Obyekt yaradarkən “**new**” açar sözündən istifadə edirik.

Tez-tez “**constructor**” termini ilə qarşılaşırıq.**Bəs constructor nədir?**

**Constructor** metoddur və class’da olan field’lərə qiymət mənimsətmək üçün istifadə olunur. C# da constructor class’ın instance’sı yaranandan sonra çağrılır və həmin class’ın adı ilə eyni olur.

public class İnsan {public string Ad;  
 public string Soyad;  
 public int Yaş;public İnsan(ad,soyad,yaş){  
 this.Ad=ad;  
 this.Soyad=soyad;  
 this.Yaş=yaş; }

**“this”** açar sözü constructor və class’da olan parametrləri əlaqələndirir. **“this”** ilə çağrılan Ad, Soyad, Yaş İnsan class’ının üzvləridir.

OOP-nin əsas dörd prinsipi var: **absraction, inheritance, polimorphizm, encapsulation.**

**Kapsullaşdırma(encapsualtion)** —bir obyektin xüsusiyyətlərini xaricdən istifadə edilməsinin qarşısını almaq üçün istifadə olunur. İnsanı nümunə kimi götürək. Bədənimizin gördüyü işlər sərhədlənib və biz buna müdaxilə edə bilmirik. Məsələn, biz sadəcə nəfəs alırıq, amma daxildə ciyərlərimizin hava ilə dolması, qana oksigenin qarışması, çirkli qanın təmizlənməsi kimi işlər görülür. Bizim işimiz isə sadəcə nəfəs alıb verməkdən ibarətdir. Eyni məntiqlə obyektlərin də bəzi işləri öz aralarında görməsi onları daha asan istifadə edə bilməyimizi təmin edir və beləcə də təhlükəsizliyi gözləyir .

Private giriş göstəricisi sayəsində bu şəkildə göstərilən field’ı başqa class’lardan gizlətmiş oluruq. Həmçinin bu field digər class’larda istifadə edilə biməz. Bəzən private field’ların xüsusiyyətlərini istifadə etməyə ehtiyacımız ola bilər. Belə vəziyyətlərdə **“property”** termini ortaya çıxır.**Bəs property nədir?**

Property bir field’ın dəyərini geri döndərməyə (**get**) və dəyər verməyə(**set**) imkan verir.

İndi isə verdiyimiz nümunəni kod şəklində yazaq.

public class İnsan  
{  
 private void QandakıKarbonDioksidiAl()  
 {  
 //.....  
 }  
   
 private void OksigeniQanaVer()  
 {  
 //.....  
 }  
   
 private void CiyəriHavailəDoldur()  
 {  
 //.....  
 }  
   
 private void CiyəriBoşalt()  
 {  
 //.....  
 }public void NəfəsAl()   
{  
 CiyəriHavailəDoldur();  
 OksigeniQanaVer();  
 QandakıKarbonDioksidiAl();  
 CiyəriBoşalt();  
}  
   
}

Bu nümunədə göründüyü kimi , İnsan class’ının ancaq NəfəsAl() metodu public olduğu üçün xaricə açıqdı. Digər metodlar isə private olduğundan İnsan class’nı istifadə edənlər bu metodlardan xəbərsiz olacaq.

**İnheritance(miras alma)**— bir obyektin xüsusiyyətlərini fərqli obyektlər tərəfindən də istifadə edəbilməsini təmin edən OOP xüsusiyyətidir. Yazılan bir class bir başqa class tərəfindən miras alına bilər. Bu zaman miras alınan class’ın bütün xüsusiyyətləri yeni class’a da aid edilir. Burada bir class başqa bir class’dan törəyərək , o class’ın public və protected giriş göstəricilərini təhvil alır. Törəyən(miras alan) class’a “derived class”, miras verən class’a isə “base class” deyilir. Inheritance’ə real həyatdan misal olaraq insan və tələbə arasında əlaqəni nümunə kimi göstərmək olar. Tələbə də insan olduğu üçün insan class’ına aid bütün xüsusiyyətlərə malikdir. Buna görə də kodda təkrarlara yol verməmək üçün inheritance’dən istifadə edilir.

Aşağıdakı nümunə ilə dediklərimizi əsaslandıraq.

class Fiqur{  
 public void fiqurunEni(int e) {  
 en= e;  
 }  
 public void fiqurunHündürlüyü(int h) {  
 hündürlük = h;  
 }  
 protected int en;  
 protected int hündürlük;  
 }  
  
 // Törəmə class  
 class Dördbucaq: Fiqur{  
 public int sahəniTap() {   
 return (en \* hündürlük);   
 }  
 }  
 class DördbucaqlınıYoxla {  
 static void Main(string[] args) {  
 Dördbucaq dördbucaqlı= new Dördbucaq();  
  
 dördbucaq.fiqurunEni(5);  
 dördbucaq.fiqurunHündürlüyü(7);  
  
 // Dördbucaqlının sahəsinin tapılması  
 Console.WriteLine("Ümumi sahə: {0}" dördbucaq.sahəniTap()); Console.ReadKey();  
 }  
 }

Buradakı nümunədə base(əsas və ya törədən) class “Fiqur” classı, deriver (törəyən ) class’ı isə “Dördbucaq” class’dır. Deriver class base class’ın “hündürlük” və ”en” xüsusiyyətlərini özünə götürür. İstifadə olunan “protected” giriş göstəricisi verilən xüsusiyyətlərin yalnız base və ondan törəyən deriver class’da istifadə edilə biləcəyini göstərir. Bu class’lardan başqa heç bir yerdə onlar çağrıla və istifadə edilə bilmir.

**Qaynaq** - <https://medium.com/pragmatech/oop-nin-%C9%99saslar%C4%B1-1-ci-hiss%C9%99-fd4b275f5f95>