Obyektlər, obyekt yönümlü proqramlaşdırma (OOP) dilinin əsas elementləridir. Adlandıra biləcəyimiz və təsvir edə biləcəyimiz hər şey obyektdir.

OOP dörd hissədən ibarətdir:

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

Class lardan istifadə edərək yeni obyekt yaratmaq mümkündür.

**ENCAPSULATİON**

OOP-nin ilk prinsipidir. Bunun vasitəsilə istifadəçi üçün lazımsız olan detaylar gizlədilir. Məsələn, kitab class -mız var. Biz bilirik ki, kitab sayca çoxdur, lakin hər bir kitabın ortaq cəhətləri də var, biz bu cəhətləri class daxilində property lər vasitəsilə göstəririk:

FIELD – class içərisində yer tutur, class ın özəlliklərini bildirmək üçün istifadə olunur. public string name; public int page;

PROPERTY – public string name {get; set;}, private int page { get; set;}

Bir class üzvü olaraq bir obyektin yaradılması **instantasiya**adlanır.

**access modifier (**giriş göstəricisi**) ilə bu proses daha rahat olur. Metodlar və property lər bir** girişgöstərici**sindən istifadə edə bilər:**

1. Public – Sistemdəki bütün siniflərin əlçatan olmasını təmin edir.
2. Private – Hər hansısa bir metod yaxud property olduğu class ın içində istifadə edilə bilər. Əgər metod yaxud property yaradarkən açar sözü unutsaq sistem özü onu avtomatik olaraq private qəbul edəcək.
3. Protected – yalnızca içində olduğu sinif və bu sinifin törəmələri (inheritance) üçün əlçatan olur.
4. İnternal – Assembly içində olan bütün siniflərdən əlçatan olur.

class Book

{

private int page;

static void Main(string[] args){

Book Forrest\_Gump = new Book(152);

Console.WriteLine(Forrest\_Gump.page);

}}

class Book

{

public string name = "Forrest Gump";

}

class Program

{

static void Main(string[] args)

{

Book myBook = new Book();

Console.WriteLine(myBook.name);

}}

**Inheritance (Miras alma)**

C # -da sahələri və metodları bir sinifdən digərinə miras almaq mümkündür. "Miras anlayışı" nı iki kateqoriyaya ayırırıq: Derived Class (child), Base class (parent)

Hər hansısa bir sinifdən miras almaq üçün : istifadə olunur.

Kodun təkrar istifadə edilməsinin qarşısını alır: yeni bir sinif yaratdığımız zaman mövcud bir sinifin sahələrini və metodlarını yenidən istifadə etmək mümkün olur.

Əgər miras almanın qarşısını almaq istəyiriksə **sealed** açar sözündən istifadə olunur.

**Polymorphism (çoxsahəli)**

**Polimorphizm**— bir obyektin müxtəlif şəkillərdə istifadə edilə biləcəyini göstərən OOP prinsipidir.

public class Books{  
...  
public void genre(){  
Console.WriteLine("Kitablar müxtəlif janrlarda yazıla bilər");   
}  
}

public class Sherlock\_Holmes:Books{  
...  
 @Override  
 public void genre(){  
 Console.WriteLine("dedektiv");  
 }  
}public class Olasılıksız:Books{  
...  
 @Override  
 public void genre(){  
 Console.WriteLine("elmi-fantastik");  
 }  
}

Buradan gördük ki, bütün class’lar üçün ortaq metod (genre()) var, amma bu metod eyni prosesi müxtəlif çıxışlarla verir.

Polimorphizm **statik** və **dinamik** olmaqla iki hissəyə ayrılır.

Statik polimorphizm metod və operatorların artıq yüklənməsi (**overloading**) olaraq bilinir və həmçinin **compile** time polymorphism də adlanır.

Dinamik polimorphizm **overriding** olaraq bilinir və həmçinin **runtime** (proses zamanı) polymorphism də adlanır. Burada eyni method həm base class’da(əsas), həm də derived(törəmə) class’da çağrıla bilər.

Overloding və overridingin fərqi ondadır ki, overloadingdə eyni method eyni class’ın içində müxtəlif parametrlərlə işlənə bilir, overriding də isə eyni method base və derived class’larda istifadə olunur.

**Abstraction (mücərrədlik)**

Bir sıra detalları gizlətmək və yalnız istifadəçi üçün lazımlı məlumatları göstərmək üçün istifadə edilən xüsusiyyətdir. Başqa sözlə desək, burada yalnız istifadəçi ona lazım olan məlumatları görə bilir. Məsələn, e-manat ödəniş terminalında hər kəs özünə lazım olan xidməti istifadə edir, burda insanlara prosesin necə getdiyi maraqlı deyil.

C# da abstraction abstract class’ların köməyi ilə yaranır. Abastract class’lar “**abstract** ” açar sözü vasitəsi ilə təyin olunur.

Abstraction’la encapsulation bir-birinə çox bənzəyir lakin onlar fərqlidirlər. Abstraction’da prosesin arxa hissəsi gizlədilir (yəni e-manatın necə işlədiyi), encapsulation da isə ümumiyyətlə məlumatlar gizlədilir.

**Puffff ☹**