# Giriş

Nesne tabanlı programlama, programlarda sınıflar tanımlamaya imkan veren ve bu sınıflardan yenilerinin türetilmesini sağlayan bir programlama tekniğidir.

Bu teknik yazılım geliştiren insanlara büyük avantajlar sağlamaktadır. Karmaşık projelerin üretmini ve bakımını kolaylaştırır, program kodunun tekrarlı kullanımına olanak sağlar (code reuseability). Bu da maliyeti azaltır, ulaşılabiliriliği artırır.

Bu teknikte esas olan gerçek hayatta var olan olguların programlamaya aktarılmasıdır. Burada iki önemli birim vardır: veri ve veriyi işleyen metodlar.

Mesela ütü sınıfımız olsun. Marka, model, renki çalıştığı elektrik voltajı, ütüleyebildiği kumaş türleri bu ütünün verilerindendir. Aynı zamanda ütü ısınabilir, ütüleme işleminde kullanılır ve soğumaya bırakılabilir. Bunlar ise ütünün metotlarıdır.

# Kapsülleme(Encapsulation)

İşlemlerin ve verilerin gizlenmesine veya bir grup içerisinde toplanmasına kapsülleme denir.

Kapsülleme sayesinde sınıf içerisinde kullanılan alanların dışarıdan herhangi bir etki ile doğrudan değiştirilmelerini engellenmiş olunur. Bu durum istek dışı atamaların önüne geçer ve eğer bu alanların değerleri değişecek ise belirlenen bir çizgide olması sağlanır. Bu sayede program üzerindeki kontrol artar.

Kapsüllemenin bir diğer işlevi ise veri korumadır. Bir sınıf tasarımında doğru olan nesnelerin durum verilerini **private** erişim belirleyicisi ile koruma altına almaktır. Bu yolla dış dünya alanın değerini elde etmek ya da değiştirmek istediğinde bu protokolden geçmek zorunda kalır. Dış dünyaya açık, yani **public** erişim belirleyicisine sahip alanların sorunu, ait oldukları sınıfın iş mantıklarını algılayabilme yeteneklerinin olmamasıdır. Dolayısıyla atanan değerin iş mantığına uygun olup olmadığına dair herhangi bir kontrolün sınıf içerisinde yapılabilmesi mümkün değildir.

Nesne kullanıcısı genelde sınıfı yazan kişi olabilir de olmayabilir de. Dolayısıyla böyle bir kontrolü nesne kullanıcısı tarafında yapmak verimli değildir. Sınıfların durum verileri olarak anılan alanlar, private erişim belirleyicisi alarak dış dünyaya kapatıldıklarında başlangıç değerleri, sınıf içerisinde tanımlandığı yerde ya da yapıcı metot yardımıyla verilebilir. Böyle bir kapsülleme tercih edilebilir ancak bazı durumlarda alanın değerinin dışarıdan okunması, değerinin sadece içsel olarak sınıf üyeleri tarafından atanması, durum verileri üzerinde yapılacak değişikliklerin dışarıya açılması gerekebilir. Burada değişiklik ile birlikte gelen verinin kontrol edilerek istenmeyen durumların önüne geçilmesi gerekliliği ortadadır.

Kapsülleme, durum verilerinin tutarlıulığını sağlamak için bir yol sunmaktadır. Public erişim belirtecine sahip alanlar tanımlamaktansa, alan verilerini private olarak tanımlamak alışkanlık haline getirilmelidir (herhangi bir iş kuralı olamasa da). Bu durumda alanlar nesne kullanıcısına iki teknikle açılabilir.

* Nesne yönelimli diğer diller için geleneksel yol olan Erişen ve Değiştiren (Accessor and Mutator) metotlar tanımlamak.
* Bir özelllik (property) tanımlamak.

Hangi teknik kullanılırsa kullanılsın, iyi kapsüllenmiş bir sınıf ham verisini (alanlarını) ve onu nasıl yönettiğinin ayrıntılarını dış dünyanın kem gözlerinden gizlemelidir. Bu yaklaşımın güzel ve faydalı olan yanı sınıfı yazan kişinin çalışan kodu bozmadan kapalı kapılar ardından metodun ya da özelliğin uygulanışını değiştirmekte serbest olmasıdır.

.NET Framework temel sınıf kütüphanesi, geleneksel Erişen/Değiştiren metotlar yerine tip özelliklerini tercih eder. Bu yüzden sınıf kütüphanesi ile sağlıklı bir şekilde etkileşen programlar için kendi tiplerimizi yazarken özelliklerden faydalanmak doğru bir davranış olacaktır. Ayrıca özelliklerin bir alanın değerini işaret edip alana değer atamaları zorunlu değildir. Bir alan olmadan da özelliklerden faydalanılabilinir.

# Kalıtım (Inheritance)

İnsanlardaki kalıtım gibi pragramlama dillerinde de kalıtım vardır. En basit tanımı ile kalıtım, bir sınıftan yeni sınıflar türetmektir. Kalıtım ile türetilen sınıflar, türetildiği sınıfın özelliklerinin tamamını da devralır. Buradan türetilmiş sınıfın türediği sınıfa ait üyelere erişebileceği sonucu da çıkarılır. Tüm sınıflarda ortak olan özellikleri tek bir sınıf içerisinde toplamak için bir sınıftan başka sınıflar türetilir. Her sınıfın özelliklerinin tekrar yazılması engellenir ve sınıflar arası hiyerarşi ortaya çıkar.

# Çok Biçimlilik (Polymorphism)

Bir metodun farklı nesnelerde veya farklı ortamlarda farklı sonuçlar üretmesidir.

C# dili tam anlamıyla nesne tabanlı bir dildir. Bu dil içerisinde sınıf kavramının önemli bir yeri vardır. Bu kavramı iyi anlamak, her türlü teknikte sınıfların avantajlarından yararlanılması ve kişiye özgü nesnelere sahip olunması sağlanabilir. Zaten .NET teknolojisinde yer alan her nesne mutlaka sınıflardan türetilmektedir.

Bir temel sınıf, içerisindeki kodların kendisinden türeyen sınıflar tarafından değiştirilebilmesini istiyorsa bu üye **virtual** anahtar kelimesi ile işaretlenmelidir. Türeyen sınıf, virtual anahtar kelimesi ile işaretli bir üyenin uygulanışını kendi sınıfına ait bir iş mantığıyla değiştirmek isteyebilir ancak zorunda değildir. Üyenin başına **override** anahtar kelimesi yazılarak yeniden kodlanması ile böyle bir değişiklik mümkün olmaktadır. Ayrıca ezilen (overriden) her üye, ihtiyaç duyulması halinde temel sınıfta yer alan uygulanışı yeniden çağırmakta serbesttir. Kod içerisinde istenen herhangi bir yerde temel sınıfın bir üyesi **base** anahtar kelimesi ile çağırılabilir.

Kapsülleme , kalıtım, çok biçimlilik neye yarar sorusuna cevap:

1. Temel sınıfta yer alan bir üyenin, türeyen sınıf tarafından nasıl değiştirileceğine dair bir yol sunmaktadır.
2. Bir türetilen sınıfın özellik ve davranışlarını, bir türeyen sınıf miras alarak sanki kendi üyesiymiş gibi kullanmaktadır.
3. Dilin nesne kullanıcısından gereksiz uygulama ayrıntılarını saklayabilmektedir.
4. Dilin var olan sınıf tanımlamalarının üzerine yeni bir sınıf tanımlaması inşa edilmesine izin verebilmesidir.

# Sınıf Üyeleri

Sınıf üyeleri temelde iki kısma ayrılmaktadır. Veri üyeleri ve fonksiyon üyeleridir.

## Veri Üyeleri

### Alanlar

Nesneye ait bilgilerin tutulduğu üye değişkenlerdir.

### Sabitler

Nesnenin değiştirilemeyen kısımlarıdır.

### Olaylar

Belirli bir tetilemenin meydana gelmesi sonucunda çağırılacak olan fonksiyonların adres bilgilerini tutmakla sorumludur.

Sınıfın veri üyeleri değer türlerinde olabileceği gibi referans türlerinde de olabilir. Her nesnenin sahip olduğu data diğer nesnelerden ayrılmıştır. Yani her nesnenin datası kendisine aittir ve hiçbir şekilde karışmaz.

## Fonksiyon üyeleri

Yöntemler (methods), özellikler (properties), yapıcı fonksiyonlar (constructors), yıkıcı fonksiyonlar (destructors), operatörler ve dizinleyiciler (indexer)

### Yöntemler

Nesnenin davranış şekilleridir. Nesne kullanıcısına üye veriler üzerinde işlem yapmasına izin verir ve nesnenin dışa kapalı üyelerine ulaşılmasını sağlar.

### Özellikler

Alanlar gibidir. Hatta bunlara akıllı alanlar da denilmektedir. Nesnenin üye verilerine değer atanmasının ve değer çekilmesinin kontrollü olarak yapılmasını sağlar.

### Yapıcı ve Sonlandırıcı Yöntemler

Nesnenin özel yöntemleridir. Yapıcı yöntemler nesne ilk oluşturulduğunda çağırılırken sonlandırıcı yöntemler nesnenin ömrü tamamlandığında çağırılır.

### Operatörler

Nesnelerin daha esnek kullanılmasına izin vermektedir.

### Dizinleyiciler

Nesnenin bir dizi gibi kullanılmasına izin verir.