Nesne Yönelimli Programlama <OOP>

→ C#, Java gibi bir nesne yönelimli programlama dilidir.

OOP ile tüm yazılım anlayışı kökten değişmiştir. 1960'larda OOP fikrini ilk ortaya atan Alan Kay, önerdiği metodolojiyi şu şekilde ifade etmiştir.

* Uygulama, nesneler ve onların ilişkileri çerçevesinde belirli bir iş yapmak için geliştirilebilmelidir.

* Her nesnenin bir sınıfı olmalıdır ve sınıflar nesnelerin ortak davranışlarını ifade etmelidir.

* Nesneler birbirleri ile iletişime geçebilmelidir.

Yazacağınız sınıflar birbirinden bağımsız olarak geliştirilebilir. Bu sayede program böl, parçala, fethet mantığı çerçevesinde çok kolay bir şekilde parçalanır ve her parça ayrı ayrı ele alınabilir.

Sınıflar (Classes)

Nesneler sınıflardan türetilir ve yetenekleriyle yapabilecekleri sınıflarda belirtilir.

Class; Belirli bir nesne türü oluşturmak için bir plan, şablon veya talimatlar dizisidir.

Object; Belirli bir tür veriyi kullanışlı hale getirmek için gereken nitelikleri ve davranışları içeren bir bileşendir.

Instance; Belirli bir sınıftan oluşturulmuş belirli bir nesnedir.

Erişim Belirleyiciler (Access Modifiers)

Public; Her yerden erişilebilir.

Private; Sadece tanımlandığı sınıf içerisinden erişilebilir.

Internal; Sadece bulunduğu projede erişilebilir.

Protected; Sadece tanımlandığı sınıfta ya da o sınıfı miras alan sınıflardan erişilebilir.

Protected Internal; Sadece tanımlandığı sınıfta ya da o sınıfı miras alan sınıflardan erişilebilir. Ayrıca tanımlamanın aynı proje içerisinde olma şartı yoktur.

* Eğer erişim belirteci belirtilmemişse, sınıflar internal'dır.

Sınıf içindeki yapılar

FIELDS

Genellikle, nesnenin özellikleri (property) için değer saklama alanıdır. Varsayılan durumda private yapılardır. Field, global alanda tanımlanmış bir değişken olarak da düşünülebilir. Yani class içerisindeki tüm metotlardan erişilebilecek bir değerdir.

PROPERTIES

Nesnenin özellikleridir. Kendi içinde iki metot barındırabilir.

→ Set metodu: Bir property'ye yeni bir değer atamak için kullanılır. Set metodu olmayan property'ler read-only durumundedir.

→ Get metodu: Bir property'nin değerini okumak için kullanılır.

Genelde bir field'da bulunan değeri değiştirmek/okumak için kullanılır.

METHODS

Nesnenin yapabildiği işlerdir.

Virtual, override, abstract metotlar oluşturulabilir. Bu sayede, miras alan sınıfın bu metot ile yapabileceği işler belirlenebilir.

Static olarak oluşturulabilir. Bu sayede, metot nesnenin değil sınıfın metodu haline gelir. Static metotlar o metoda başka sınıflardan instance alınmadan ulaşılmasına olanak sağlar.

CONSTRUCTOR

Constructor ismi, class'ın ismi ile aynı olmak zorundadır.
Nesne ortaya çıkartılırken yapılacak işlemleri barındırır.
Farklı parametreleri alarak aşırı yüklenebilir. (overload)

EVENTS

Nesnelerin tepkileridir. Delegate'ler ile birlikte kullanılır.

ENUM

Bu tip, değişkenin alabileceği değerlerin belli (sabit) olduğu durumlarda programı daha okunabilir hale getirmek için kullanılır.

enum TipAdı { değer1, değer2, ..., değerN } DeğişkenAdı;

Sarmalama / Paketleme (Encapsulation)

Bir nesnenin özelliklerinin dışardan kullanılmasını sınırlamak için kullanılır. Nesnelerimizin bazı işleri kendi içinde yapması ve bu sayede daha kolay kullanılabilmesi sağlanır böylece veri güvenliği de sağlanmış olur.

Bu işlem yapılırken access modifierlarla işlemlere ne şekillerde erişileceğini belirlemeniz yeterlidir.

Miras alma / Kalıtım (Inheritance)

Bir nesnenin özelliklerinin farklı nesneler tarafından da kullanabilmesine olanak sağlayan OOP özelliğidir. Yazılan bir sınıf başka sınıf tarafından miras alınabilir. Bu işlem yapıldığı zaman temel alınan sınıfın tüm özellikleri yeni sınıfa aktarılır.

Arayüz (Interface)

Temelde sınıflara sunabileceğiniz, sınıfın hangi isimde ve hangi tipte parametreleri alan bir metoda sahip olacağını söylemenizi sağlayan yapıdır.

Bir sınıfa istediğiniz kadar arayüz uygulayabilir / giydirebilirsiniz (implement). Arayüz içine yazılan metotların işlemleri kesinlikle yazılmaz. Arayüzler hangi işin yapılması gerektiğini belirtir ama nasıl yapılacağına karışmaz. Bu metotların işlemleri, arayüzün uygulandığı sınıfta yazılır. Arayüzün access modifier'ı olsa da, içine yazılanların olmaz.

Arayüzler esas gücünü ve önemini çok biçimlilikten (polymorphism) alır, çünkü çokbiçimlilik normal bir sınıfla yapılabildiği gibi, aynı arayüz giydirilmiş nesneler arasında da kullanılabilir. Arayüzler .NET Framework içinde de sıklıkla kullanılır.

Arayüzler sınıflara kural getirmek için kullanıldığı gibi, ortak kullanım alanları da oluştururlar.

Çokbiçimlilik (Polymorphism)

Statik çokbiçimlilik; metot ve operatörlerin aşırı yüklenmesi (overload) olarak belirtilir.

Dinamik çokbiçimlilik; özet sınıflardan miras alma yoluyla işlemlerin gerçekleştirilmesi işlemine veriler isimdir.