**KÜBRA SULAR**

**HAFTA 1 ÖDEV**

1. **OOP NEDİR?**

Nesne yönelimli programlama (NYP), (İngilizce: Object - Oriented Programming)(OOP), Her işlevin nesneler olarak soyutlandığı bir [programlama](https://tr.wikipedia.org/wiki/Programlama) yaklaşımıdır. NYP destekleyen programlama dilleri yüksek seviye diller olarak adlandırılır.

1960'lı yılların sonuna doğru ortaya çıkan bu yaklaşım, o dönemin yazılım dünyasında beliren bir bunalımın sonucudur. Yazılımların karmaşıklığı ve boyutları sürekli artıyor, ancak belli bir nitelik düzeyi korumak için gereken bakımın maliyeti zaman ve çaba olarak daha da hızlı artıyordu. NYP'yi bu soruna karşı bir çözüm haline getiren başlıca özelliği, yazılımda birimselliği (modularity) benimsemesidir. NYP ayrıca, bilgi gizleme (information hiding), veri soyutlama (data abstraction), çok biçimlilik (polymorphism) ve kalıtım (inheritance) gibi yazılımın bakımını ve aynı yazılım üzerinde birden fazla kişinin çalışmasını kolaylaştıran kavramları da yazılım literatürüne kazandırmıştır. Sağladığı bu avantajlardan dolayı, NYP günümüzde geniş çaplı yazılım projelerinde yaygın olarak kullanılmaktadır.

NYP'nin altında yatan birimselliğin ana fikri, her bilgisayar programının (izlence), etkileşim içerisinde olan birimler veya nesneler kümesinden oluştuğu varsayımıdır. Bu nesnelerin her biri, kendi içerisinde veri işleyebilir, ve diğer nesneler ile çift yönlü veri alışverişinde bulunabilir. Hâlbuki NYP'den önce var olan tek yaklaşımda (Yordamsal programlama), programlar sadece bir komut dizisi veya birer işlev (fonksiyon) kümesi olarak görülmektedirler.

1. **(Polymorphism) Nedir ?**

Farklı işlemler yapan fakat aynı isimdeki özellik veya metotların kullanımına çok biçimlilik (polymorphism) denmektedir. Çok biçimlilik, “birçok form içeren” anlamına gelmektedir. Aynı isimde olmalarına rağmen farklı işlemlerin yer aldığı metotlar için kullanılır.

Bir temel sınıfın özellik ve metotlarını, temel sınıftan türetilen bir sınıfa aktardığımızda temel sınıfa ait metotları türemiş sınıf içerisinde de kullanabiliriz. Ancak her iki sınıfta da aynı metot bulunduğunda temel sınıftaki metot kullanılacak, türemiş sınıftaki metot ise kullanılmayacaktır. Temel sınıfa ait metotları türemiş sınıfta farklı biçimlerde tanımlayarak kullanmak için override ve virtual niteleyicilerini kullanmak gereklidir.

1. **Private, Protected ve Public**

Protected: Protected aslında private e benzer. Protected özelliği atanan bir değişken veya metot, sadece sınıf içerisinden veya türetilen bir sınıf içerisinden erişilebilir.. Yani bir sınıf extend ettiği sınıfın protected üyelerine erişebilir ama private üyelerine erişemez.

Private: Private üyelere o sınıf dışından herhangi bir şekilde erişilemez ve bu üyeler kalıtım yoluyla aktarılmaz.. Private olarak tanımlanan değişken ve fonksiyonlar sadece o sınıfın kendi metotları tarafından kullanılabilir. Yani işin özü o metod veya değişken sadece o sınıfa özel olur. Bir değişkeni private olarak tanımladığınızda ve bu değişkenin başka bir sınıf tarafından kullanılması gerektiğinde bu değişkeni döndüren bir public fonksiyon yazarak işlemi gerçekleştirebilirsiniz.

Public: Public kullanılan en yaygın tanımdır. Bir metod veya değişken public olarak tanımladığında o üyeye ilgili sınıfın her yerinden erişebilirsiniz. Aynı zamanda sınıftan türeyen bir sınıf içersinden de erişilebilir.

1. **Soyutlama**

Soyutlama , yalnızca temel nitelikleri “gösteren” ve gereksiz bilgileri “gizleyen” nesne yönelimli programlama kavramıdır. Soyutlamanın temel amacı, gereksiz detayları kullanıcılardan gizlemektir. Soyutlama, nesnenin yalnızca ilgili ayrıntılarını kullanıcıya göstermek için daha büyük bir havuzdan veri seçmektir. Programlama karmaşıklığını ve çabalarını azaltmaya yardımcı olur. OOP'lerin en önemli kavramlarından biridir.