Unıty wıth c# 101 Bootcamp araştırma ödevi

İbrahim Birkan Öztürk

# **Solid Yazılım Geliştirme Nedir?**

Solid Yazılım Geliştirme, Robert C. Martin tarafından ortaya atılan ve yazılımın beş temel ilkesine dayanan bir yazılım geliştirme prensipleri kümesidir. Bu prensipler, daha bakımı kolay, güvenilir ve sürdürülebilir yazılımlar oluşturmayı amaçlamaktadır.

**Solid prensipleri şunlardır:**

1. **Tek Sorumluluk Prensibi (Single Responsibility Principle - SRP):** Her bir sınıfın sadece tek bir sorumluluğu olmalıdır. Bu, sınıfların daha odaklı ve anlaşılır olmasını sağlar.
2. **Açık Kapalı Prensibi (Open/Closed Principle - OCP):** Mevcut kodları değiştirmeden yeni işlevler eklenebilmelidir. Bu, kodun daha esnek ve genişletilebilir olmasını sağlar.
3. **Liskov Yerine Koyma Prensibi (Liskov Substitution Principle - LSP):** Bir alt sınıf, üst sınıfının yerine kullanılabiliyor olmalıdır. Bu, kodun daha sağlam ve test edilebilir olmasını sağlar.
4. **Arayüz Ayırma Prensibi (Interface Segregation Principle - ISP):** Yüksek seviyeli modüller düşük seviyeli modüllere değil, soyutlamalara bağımlı olmalıdır. Bu, kodun daha gevşek bir şekilde bağlı ve test edilebilir olmasını sağlar.
5. **Bağımlılıkların Ters Çevrilmesi Prensibi (Dependency Inversion Principle - DIP):** Birden fazla istemciye hizmet veren bir arayüz, her bir istemcinin ihtiyaç duymadığı tüm yöntemleri içermemelidir. Bu, kodun daha az karmaşık ve daha kullanılabilir olmasını sağlar.

**Solid Yazılım Geliştirmenin Faydaları:**

* **Daha az Hata:** Solid prensipleri, hatalara yol açabilecek karmaşık kodlardan kaçınmaya yardımcı olur.
* **Daha Kolay Bakım:** Solid kod, daha iyi organize edilmiş ve belgelendirilmiş olma eğilimindedir, bu da hataları bulmayı ve düzeltmeyi kolaylaştırır.
* **Daha Fazla Esneklik:** Solid kod, yeni işlevler eklemek veya mevcut işlevleri değiştirmek için daha kolay uyarlanabilir.
* **Daha İyi Yeniden Kullanılabilirlik:** Solid kod, diğer projelerde yeniden kullanılmak için daha kolay uyarlanabilir.
* **Artan Geliştirici Üretkenliği:** Solid prensiplerini anlayan geliştiriciler, daha az hata içeren ve daha bakımı kolay kodlar üretebilirler.

**Solid** prensiplerine uymak karmaşıklığı önlemek ve “İyi Kod” yazmayı sağlar. Bu sayede geliştirilen projenin herkes tarafından müdahale edilebilir olmasını sağlar. Her yazılımcının bu prensiplere uyması geliştirilen projenin sürdürülebilir ve anlaşılabilir bir şekilde geliştirmesine olanak sağlar. Geliştirilen yazılımın gelecekteki teknolojilere uyumlu olmasını ve kolay entegre edilebilmesini sağlar.

# **OOP'nin 5 Temel İlkesi Nedir?**

Nesne yönelimli programlama (OOP) beş temel ilkeye dayanır. Bu prensipler, yazılım geliştiricilerin kodlarını daha düzenli, daha anlaşılır, daha sürdürülebilir ve daha esnek hale getirmeye yardımcı olur.

**Sınıflar ve Nesneler (Classes and Objects):** OOP'nin temelini sınıflar ve nesneler oluşturur. Bir sınıf, bir nesnenin yapısını ve davranışını tanımlar. Nesneler, bu sınıflardan türetilir ve belirli bir durumu veya davranışı temsil ederler.

**Kalıtım (Inheritance):** Kalıtım, bir sınıfın başka bir sınıftan özelliklerini ve davranışlarını miras almasını sağlar. Bu, kodun tekrar kullanılabilirliğini artırır ve sınıflar arasında hiyerarşi oluşturur. Alt sınıflar (türetilmiş sınıflar), üst sınıfların (temel sınıfların) özelliklerini ve davranışlarını devralır.

**Kapsülleme (Encapsulation):** Kapsülleme, verilerin ve işlevlerin bir sınıf içinde gruplandırılmasını ve sınıf dışındaki erişimin kısıtlanmasını sağlar. Bu, verilerin gizliliğini korur ve sınıfın iç yapısının dış dünyadan gizlenmesini sağlar.

**Çok Biçimlilik (Polymorphism):** Çok biçimlilik, aynı adı taşıyan ancak farklı işlevlere sahip olan metodların kullanılmasını sağlar. Bu, kodun daha esnek olmasını ve farklı türlerdeki nesnelerin aynı arayüz üzerinden işlenebilmesini sağlar.

**Birlikte Kullanılabilirlik (Abstraction):** Birlikte kullanılabilirlik, bir nesnenin temel özelliklerini vurgulayarak detayları gizlemeyi sağlar. Bu, karmaşıklığı azaltır ve kodun daha anlaşılır olmasını sağlar. Birlikte kullanılabilirlik, soyut sınıflar veya arayüzler kullanılarak gerçekleştirilir.

# **Enum Class, Record, Struct nedir?**

**Enum Class (Numaralandırma Sınıfı):** Enum, “enumeration” (numaralandırma) anlamına gelir ve bir sabitler kümesini temsil etmek için kullanılır.

Örneğin, haftanın günlerini veya bir renk paletindeki renkleri temsil etmek için enum sınıfları kullanılabilir.

Kotlin programlama dilinde enum class olarak adlandırılır ve belirli değerleri bir tür altında gruplandırır.

Örnek kod:

public enum Days

{

Monday,

Tuesday,

Wednesday,

Thursday,

Friday,

Saturday,

Sunday

}

**Record (Kayıt):** Record, C# 9.0’da gelen bir özelliktir ve immutable (değişmez) verileri temsil eder. Bir record, bir nesnenin topyekün olarak sabit/değişmez olarak kalmasını sağlar ve bu durumu güvence altına alır. Record’lar, içerisinde veri barındıran hafif (lightweight) sınıflar olarak düşünülebilir.

Örnek Kod:

public record Person

{

public string Name { get; init; }

public int Age { get; init; }

}

**Struct (Yapı):** Struct, C programlama dilinde bir bellek bloğunda fiziksel olarak gruplandırılmış değişkenler listesini tanımlayan bir bileşik veri türüdür.

Struct, farklı veri tiplerinde elemanları içerebilir ve dosya içinde tutulacak kayıtları oluşturmakta kullanılır.

Struct, genellikle bitişik bellek bloğu olarak sınıflandırılır ve verileri sabit bir offsette bulunur.

Örnek Kod:

public struct Point

{

public int X;

public int Y;

}

Kaynakça

<http://cleancoder.com/>

<https://tr.wikipedia.org/wiki/SOLID_Yaz%C4%B1l%C4%B1m_%C4%B0lkeleri>

<https://gokhana.medium.com/solid-nedir-solid-yaz%C4%B1l%C4%B1m-prensipleri-nelerdir-40fb9450408e>

<https://www.linkedin.com/pulse/solid-yaz%C4%B1l%C4%B1m-prensipleri-nelerdir-techinday-53pyf/>

[Java OOP İlkeleri » Programlama Dersleri (codingbytime.com)](https://ders.codingbytime.com/java-oop-ilkeleri/)

[Object Oriented Programming(OOP) Nedir ? Nasıl Uygulanır ? | by Sefa Can Pehlivan | Medium](https://medium.com/@sefajobs/object-oriented-programming-oop-nedir-nas%C4%B1l-uygulan%C4%B1r-458dc0b62f85)

[C# 9.0 – Records İle Immutable Datalar – Yazılım Mimarileri ve Tasarım Desenleri Üzerine (gencayyildiz.com)](https://www.gencayyildiz.com/blog/c-9-0-records-ile-immutable-datalar/)

[C programlama dilinde struct - Vikipedi (wikipedia.org)](https://tr.wikipedia.org/wiki/C_programlama_dilinde_struct)

[Enum classes | Kotlin Documentation (kotlinlang.org)](https://kotlinlang.org/docs/enum-classes.html)