# SOLID Tasarım İlkeleri Nedir?

Çalışma Kağıdı

SOLID, Tek Sorumluluk (Single Responsibility), Açık/Kapalı (Open/Closed), Yerine Koyma (Liskov Substitution), Arayüz Bölümleme (Interface Segregation) ve Bağımlılıkların Ters Çevrilmesi (Dependency Inversion) ilkelerinden oluşan bir akrostiştir.

## Sorular

1. Aşağıdakilerden hangisi SOLID prensiplerinden biri DEĞİLDİR?
A) Tek Sorumluluk Prensibi
B) Açık/Kapalı Prensibi
C) Tasarım Kalıpları Prensibi
D) Liskov Yerine Koyma Prensibi

2. Bir sınıfın yalnızca tek bir sorumluluğu olması gerektiğini belirten SOLID prensibi hangisidir?
A) Açık/Kapalı Prensibi
B) Tek Sorumluluk Prensibi
C) Arayüz Bölümleme Prensibi
D) Bağımlılıkların Ters Çevrilmesi Prensibi

3. Yazılım varlıklarının genişletilmeye açık, ancak değişikliğe kapalı olması gerektiğini savunan prensip hangisidir?
A) Liskov Yerine Koyma Prensibi
B) Bağımlılıkların Ters Çevrilmesi Prensibi
C) Tek Sorumluluk Prensibi
D) Açık/Kapalı Prensibi

4. Tek Sorumluluk Prensibi (SRP) örneği?

5. Açık/Kapalı Prensibi (OCP) örneği?

6. Liskov Yerine Koyma Prensibi (LSP) örneği?

7. Tanımla: S: Tek Sorumluluk Prensibi (SRP)

8. Tanımla: O: Açık/Kapalı Prensibi (OCP)

9. Tanımla: L: Liskov Yerine Koyma Prensibi (LSP)

# Cevap Anahtarı

1. C) Tasarım Kalıpları Prensibi — Tasarım Kalıpları (Design Patterns) SOLID prensiplerinden ayrı bir konsepttir, ancak SOLID prensipleri tasarım kalıplarının uygulanmasında rehberlik edebilir.

2. B) Tek Sorumluluk Prensibi — Tek Sorumluluk Prensibi (SRP), bir sınıfın yalnızca tek bir işlevi veya sorumluluğu olması gerektiğini savunur.

3. D) Açık/Kapalı Prensibi — Açık/Kapalı Prensibi (OCP), kodun yeni özellikler eklenerek genişletilebilmesini, ancak mevcut kodun değiştirilmeden çalışmaya devam etmesini hedefler.

4. Bir `Kullanici` sınıfının hem kullanıcı bilgilerini yönetmesi hem de veritabanı işlemlerini yapması SRP'yi ihlal eder. Bu durumu çözmek için `KullaniciBilgileri` ve `KullaniciVeritabaniIslemleri` gibi ayrı sınıflar oluşturulmalıdır.

5. Bir `Hesaplayici` sınıfının yeni matematiksel işlemler için her seferinde güncellenmesi OCP'yi ihlal eder. Bunun yerine, her işlem için ayrı bir `Islem` arayüzü ve bu arayüzü uygulayan sınıflar (örn. `Toplama`, `Cikarma`) kullanılarak sınıf, yeni işlemlere kapalı hale getirilir.

6. Bir `Kus` sınıfı ve onun `Penguen` alt sınıfı düşünelim. Eğer `Kus` sınıfının `uc` metodu varsa ve `Penguen` bu metodu geçersiz kılarak uçamaz hale geliyorsa, LSP ihlal edilir. Çünkü `Penguen` nesnesi, `Kus` nesnesinin beklenildiği her yerde kullanılamaz.

7. Bir sınıfın yalnızca tek bir sorumluluğu olmalıdır. Yani, bir sınıfı değiştirmek için tek bir neden olmalıdır.

8. Yazılım varlıkları (sınıflar, modüller, fonksiyonlar vb.) genişletilmeye açık, ancak değişikliğe kapalı olmalıdır.

9. Alt sınıflar, üst sınıflarının yerine geçebilmelidir. Yani, bir alt sınıf nesnesi, üst sınıf nesnesinin kullanıldığı her yerde kullanılabilmelidir.