**Single Responsibility Principle (Tek Sorumluluk Prensibi)**: Bu prensip, her bir sınıfın veya modülün (fonksiyonun) yalnızca bir tek sorumluluğu olması gerektiğini belirtir. Bir sınıfın değişmesi gerektiğinde bu değişikliğin sadece bir nedeni olmalıdır. Böylece, kodun anlaşılması, bakımı ve genişletilmesi kolaylaşır.

Örneğin, Bir online mağaza uygulaması düşünelim. Ürünleri listeleme, sepet işlemleri, ödeme işlemleri gibi farklı işlevleri olan birçok sınıf bulunur. Sepet işlemlerini yapan bir sınıfın aynı zamanda ödeme işlemlerini de üstlenmesi, bu prensibe aykırıdır. Her sınıfın sadece kendi sorumluluğu olan bir işlevi olmalıdır. Yani, sepet işlemleri ve ödeme işlemleri farklı sınıflara ayrılmalıdır.

Örnek2, Bir blog uygulaması düşünelim. Bu uygulamada yazıların saklandığı bir veritabanı, kullanıcı oturumlarını yöneten bir oturum yöneticisi ve görsel arayüzü oluşturan bir görüntüleme bileşeni bulunmaktadır. Bu prensip gereği, her bir bileşen sadece kendi sorumluluğu olan işleri yapmalıdır. Örneğin, oturum yöneticisi, kullanıcı oturumlarını yönetmeli ancak aynı zamanda veritabanına yazı eklememelidir.

Örnek3, Bir e-ticaret platformu düşünelim. Ürün yönetimi, sepet işlemleri, ödeme işlemleri ve müşteri yönetimi gibi farklı bileşenler bulunur. Her biri sadece kendi alanında sorumludur. Örneğin, ödeme işlemleriyle ilgilenen bir sınıf, aynı zamanda ürün yönetimi işlemleri yapmamalıdır.

**Open/Closed Principle (Açık/Kapalı Prensibi)**: Bu prensip, yazılım bileşenlerinin (sınıflar, modüller, vs.) yeni özellikler eklenebilmesine açık olmalı, ancak mevcut kodların değiştirilmesine de kapalı olmasını söyleyen prensiptir. Yani mevcut kodu değiştirmeden, yeni işlevselliği eklemek mümkün olmalıdır. Bu sayede mevcut kodun stabilitesi korunur.

Örneği, Bir grafik çizim kütüphanesi düşünelim. Başlangıçta yalnızca basit şekiller (örneğin, daireler, kareler) çizimini destekliyor olsun. Ancak daha sonra üçgenlerin çizimi eklenmek istenirse, bu yeni özellik eklenirken mevcut daire ve kare çizim kodlarına dokunulmadan sadece üçgen çizimi eklenmelidir.

Örnek2, Bir çevrimiçi oyun düşünelim(LOL). Başlangıçta sadece temel karakter türleri bulunmaktadır: savaşçı, büyücü, okçu. Ancak ilerleyen zamanlarda yeni karakter türleri eklemek istenebilir. Bu prensibe uygun şekilde, mevcut karakter türlerinin kodlarına dokunmadan yeni karakter türleri eklemek mümkün olmalıdır.

Örnek3, Bir rapor oluşturma sistemi düşünelim. Başlangıçta sadece PDF formatında raporlar oluşturulabiliyor olsun. Ancak ileride Excel, Word gibi farklı rapor formatları da eklemek istenebilir. Bu prensibe uygun olarak, mevcut PDF rapor oluşturma kodlarını değiştirmeden yeni rapor formatlarını sisteme entegre etmek mümkün olmalıdır.

**Liskov Substitution Principle (Liskov Yerine Koyma Prensibi)**: Bu prensip, alt sınıfların üst sınıfların yerine geçebilir olması gerektiğini, üst sınıflarının fonksiyon veya davranışlarını yapabileceğini ifade eder. Yani alt sınıflar, üst sınıfların yerine kullanılabilir ve bu kullanım sonucunda programın davranışı değişmemelidir.

Örneğin, Bir şekil sınıfı hiyerarşisi düşünelim. Üçgen ve dikdörtgen alt sınıfları olsun. Bu prensibe uygunluk, her iki alt sınıfın da üst sınıfın işlevlerini (örneğin, alan hesaplama) aynı şekilde uygulayabildiği ve programın herhangi bir beklenmedik davranış sergilemediği durumda gerçekleşir. Yani, bir üçgen veya dikdörtgen nesnesi bir şekil olarak kullanıldığında, beklenen sonuçlar elde edilmelidir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Örnek2, Bir dosya işlemleri senaryosu düşünelim. Üst sınıf "Dosya" sınıfı ve alt sınıflar "MetinDosyasi" ve "ResimDosyasi" olsun.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sonuç olarak, Her iki alt sınıf da "Dosya" sınıfının yerine kullanıldığında, beklenen dosya okuma davranışı elde edilir.