SoC (Seperation of Concern) & SRP (Single Responsibilty Principle) bunlar nedir ? okuyun dediler..

**SoC (Separation of Concerns)** ve **SRP (Single Responsibility Principle)**, yazılım geliştirme sürecinde kodun daha organize, bakımı kolay ve anlaşılır olmasını sağlayan iki önemli prensiptir. Her ikisi de temelde **"bir şeyi sadece bir iş yapacak şekilde tasarla"** felsefesine dayanır. Şimdi bu kavramları detaylıca inceleyelim:

**1. SoC (Separation of Concerns) - İlgili Alanların Ayrılması**

**Tanım:**

* **SoC**, bir yazılım sisteminin farklı işlevsel parçalara (concern) ayrılması ve her bir parçanın birbirinden bağımsız olarak tasarlanmasıdır.
* Her bir "concern" (ilgi alanı), belirli bir işlevi veya sorumluluğu temsil eder.
* Bu prensip, karmaşıklığı azaltır ve kodun daha modüler olmasını sağlar.

**Örnek:**

Bir web uygulaması düşünelim:

* **Frontend**: Kullanıcı arayüzü (UI) ve kullanıcı etkileşimi.
* **Backend**: İş mantığı (business logic) ve veritabanı işlemleri.
* **Veritabanı**: Veri depolama ve yönetimi.

Bu üç alan birbirinden ayrılmıştır. Frontend, backend ile iletişim kurar ama backend'in nasıl çalıştığını bilmez. Bu, her bir alanın bağımsız olarak geliştirilmesini ve test edilmesini kolaylaştırır.

**Faydaları:**

* **Modülerlik**: Her bir parça bağımsız olarak geliştirilebilir ve test edilebilir.
* **Bakım Kolaylığı**: Bir parçada yapılan değişiklik, diğer parçaları etkilemez.
* **Yeniden Kullanılabilirlik**: Ayrılmış parçalar farklı projelerde tekrar kullanılabilir.

**2. SRP (Single Responsibility Principle) - Tek Sorumluluk Prensibi**

**Tanım:**

* **SRP**, bir sınıfın veya fonksiyonun **sadece bir sorumluluğu** olması gerektiğini söyler.
* Bu prensip, Robert C. Martin tarafından **SOLID** prensiplerinin ilki olarak tanıtılmıştır.
* Bir sınıf veya fonksiyon, yalnızca bir işi yapmalı ve o işi iyi yapmalıdır.

**Örnek:**

Bir rapor oluşturma sistemi düşünelim:

* **Yanlış Yaklaşım**:

python

Copy

class Report:

def generate\_report(self, data):

# Rapor oluşturma işlemi

pass

def save\_report(self, report):

# Raporu dosyaya kaydetme işlemi

pass

Bu sınıf, hem rapor oluşturma hem de raporu kaydetme sorumluluğunu üstleniyor. Bu, SRP'ye aykırıdır.

* **Doğru Yaklaşım**:

python

Copy

class ReportGenerator:

def generate\_report(self, data):

# Rapor oluşturma işlemi

pass

class ReportSaver:

def save\_report(self, report):

# Raporu dosyaya kaydetme işlemi

pass

Burada, rapor oluşturma ve raporu kaydetme işlemleri iki ayrı sınıfa ayrılmıştır. Her sınıf sadece bir sorumluluğa sahiptir.

**Faydaları:**

* **Kodun Anlaşılırlığı**: Her sınıf veya fonksiyonun tek bir amacı olduğu için kod daha kolay anlaşılır.
* **Bakım Kolaylığı**: Bir sorumlulukta değişiklik yapmak, diğer sorumlulukları etkilemez.
* **Test Edilebilirlik**: Her bir sorumluluk ayrı ayrı test edilebilir.

**SoC ve SRP Arasındaki İlişki:**

* **SoC**, daha geniş bir kapsamda sistemin farklı parçalara ayrılmasını ifade eder. Örneğin, bir uygulamanın frontend, backend ve veritabanı gibi büyük parçalara ayrılması.
* **SRP**, daha küçük ölçekte, bir sınıfın veya fonksiyonun sadece bir iş yapmasını ifade eder.
* İkisi de **"bir şeyi sadece bir iş yapacak şekilde tasarla"** felsefesine dayanır.

**Örnek: SoC ve SRP Birlikte Kullanımı**

Bir e-ticaret uygulaması düşünelim:

1. **SoC**:
   * **Frontend**: Kullanıcı arayüzü.
   * **Backend**: Sipariş işlemleri, ödeme işlemleri, kullanıcı yönetimi gibi iş mantıkları.
   * **Veritabanı**: Ürün bilgileri, kullanıcı bilgileri, sipariş bilgileri.
2. **SRP**:
   * Backend'deki her bir iş mantığı ayrı sınıflara ayrılır:

python

Copy

class OrderManager:

def create\_order(self, user, products):

# Sipariş oluşturma işlemi

pass

class PaymentProcessor:

def process\_payment(self, order):

# Ödeme işlemi

pass

class UserManager:

def register\_user(self, user\_data):

# Kullanıcı kayıt işlemi

pass

**Sonuç:**

* **SoC**, sistemin büyük parçalara ayrılmasını sağlar.
* **SRP**, her bir parçanın içindeki sınıf veya fonksiyonların sadece bir iş yapmasını sağlar.
* Bu prensipler, yazılımın daha **modüler**, **bakımı kolay** ve **anlaşılır** olmasını sağlar.

Bu prensipleri uyguladığınızda, yazılımınız daha esnek ve gelecekteki değişikliklere daha hazır hale gelir.