Java projelerimde sıkça Singleton, MVC ve Repository tasarım şablonlarını kullanırım. İşte her birinin ne zaman ve neden tercih ettiğime dair somut açıklamalar**:**

1- Singleton Design Pattern

Ne zaman kullanırım?

- Eğer bir sınıftan yalnızca bir tane olması gerekiyorsa, Singleton kullanırım.
- Konfigürasyon yönetimi, loglama, önbellekleme (cache) gibi durumlarda Singleton en iyi çözümdür.

Somut Örnek:

Bir uygulamada loglama işlemlerini yöneten bir sınıf düşünelim. Eğer her log kaydında yeni bir nesne oluşturursam, gereksiz bellek tüketimi olur ve performans düşer. Bunun yerine tek bir Logger nesnesi oluşturarak her yerden aynı nesneyi kullanırım.

2 - **MVC (Model-View-Controller) Design Pattern**

Ne zaman kullanırım?

- Web ve masaüstü uygulamalarında kodun düzenli olmasını sağlamak için kullanırım.
- Veri, iş mantığı ve kullanıcı arayüzünü ayrı tutarak daha yönetilebilir ve genişletilebilir bir yapı oluştururum.

Somut Örnek:

Bir e-ticaret sitesinde ürün bilgilerini listeleyen bir sayfa düşünelim:

- Model (Veri Katmanı): Ürünlerin bilgileri (isim, fiyat, stok durumu) burada saklanır.
- Controller (İş Mantığı Katmanı): Kullanıcı, sepete ekleme işlemi yaptığında bu isteği işler.
- View (Görsel Arayüz): Kullanıcıya ürünleri gösteren ve işlemleri yapmasını sağlayan sayfadır.

Bu yapı sayesinde veri değiştiğinde yalnızca ilgili kısım güncellenir, kod daha okunaklı ve sürdürülebilir olur.

3 -**Repository Design Pattern**

Ne zaman kullanırım?

- Veritabanı erişimini soyutlamak ve merkezi hale getirmek için kullanırım.
- Kodun veri kaynağına bağımlı olmamasını sağlarım, böylece farklı veri tabanlarıyla çalışmak kolay olur.

Somut Örnek:

Bir kullanıcı yönetim sistemi düşünelim. Eğer her yerde doğrudan SQL sorguları yazarsam, kod karmaşık hale gelir ve veritabanı değişirse kodu güncellemek zor olur. Bunun yerine, bütün veri tabanı işlemlerini tek bir "UserRepository" sınıfında toplarım. Böylece kullanıcı ekleme, silme ve güncelleme işlemleri merkezi bir noktadan yönetilir.