Reçete ve Yemek Tarifi Yönetim Sistemi

GİRİŞ

Bu proje yemek tarifleri ve reçetelerin yönetim sistemidir. Sistem; farklı kategorilerdeki yemeklerin (ana yemek, çorba, tatlı) oluşturulmasını, tariflere ek bilgiler (besin değeri, vegan alternatifler vb.) eklenmesini, tariflerin gruplandırılmasını ve malzeme stok durumuna göre kullanıcıların bilgilendirilmesini amaçlamaktadır. Projeye en uygun alan “Yapılandırma Aracı”.

Projede dört temel tasarım deseni kullanılmıştır:

- **Factory Method**
- **Decorator**
- **Composite**
- **Observer**

PROJENİN SİSTEMİ

Sistem, dört temel amaca hizmet etmektedir:

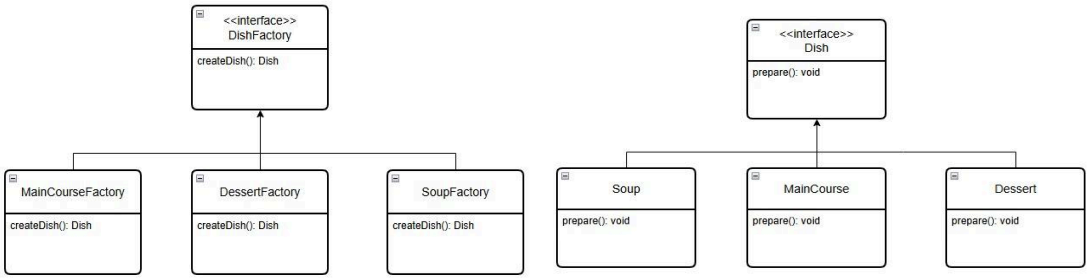
1. Farklı yemek türleri için fabrika yapıları (Factory Method)
2. Vegan, glutensiz gibi alternatif tariflerin eklenebilmesi (Decorator)
3. Tariflerin ana ve alt tarifler şeklinde gruplanabilmesi (Composite)
4. Malzeme stoklarının takibi ve azaldığında kullanıcıya bildirim gönderilmesi (Observer)

TASARIM DESENİ UYGULAMASI

1. Desen Adı: Factory Method

- **Problem:** Farklı türde yemekleri (çorba, ana yemek, tatlı) oluşturmak için doğrudan new anahtar kelimesi kullanmak yerine, soyutlama ile esnek bir yapı kurma ihtiyacı.

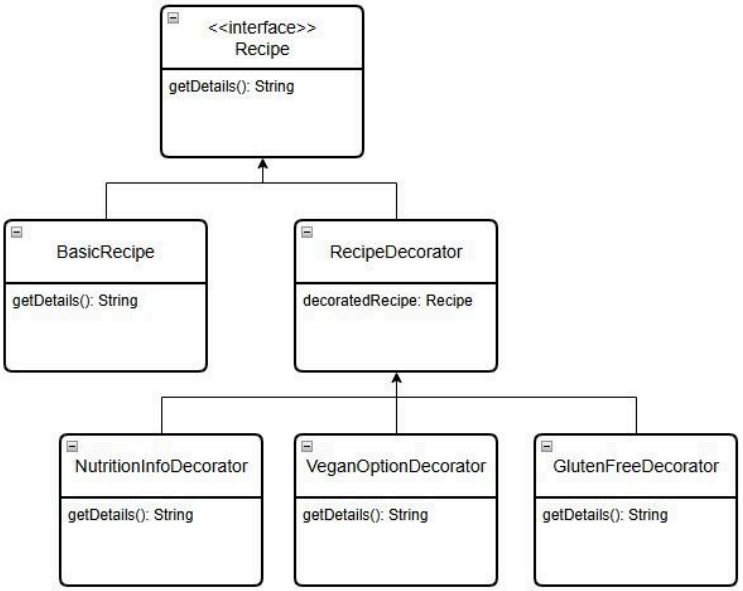
- **Çözüm:** Yemek türlerine özel olarak yemek üreten bir sistem kuruldu. DishFactory adında bir arayüz tanımlandı ve bu arayüzü temel alan SoupFactory, MainCourseFactory ve DessertFactory gibi alt sınıflar sayesinde her yemek türü için özel createDish() metodu kullanılarak nesne üretimi düzenli ve merkezi bir şekilde sağlandı.



- **Sınıflar:** Dish (interface), Soup, MainCourse, Dessert (implementasyonlar), DishFactory, SoupFactory, MainCourseFactory, DessertFactory.
- **Gerekçe:** Yeni bir yemek türü eklemek gerektiğinde, mevcut sınıflar değiştirilmeden yeni bir Factory sınıfı ile sistem genişletilebilir. Bu, open/closed ilkesine uygundur.
- **Karşılaşılan Zorluklar:** Factory'leri yapılandırırken interface soyutlamasının uygulanabilirliğini sağlamak için sınıf ilişkilerini net biçimde planlamak gerekti.

2. Desen Adı: Decorator

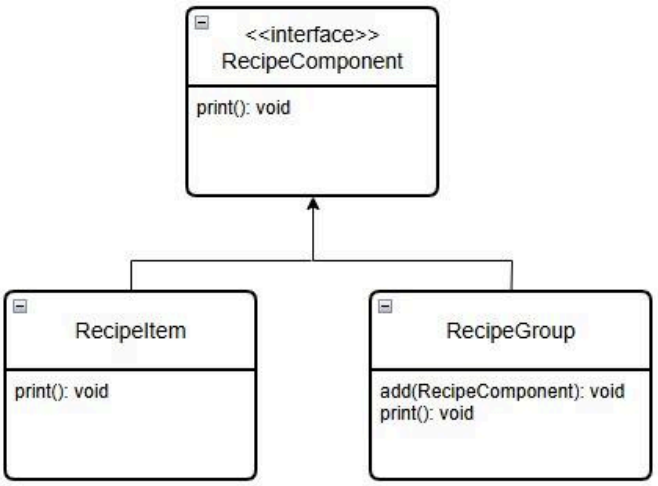
- **Problem:** Temel tarif nesnesine farklı beslenme etiketlerini (vegan, glutensiz, kalori bilgisi) dinamik olarak eklemek.
- **Çözüm:** Recipe arayüzü tanımlandı. BasicRecipe sınıfı, dekore edilmemiş haldeki basit tarif bilgilerini içeren sınıf olarak yazıldı. RecipeDecorator soyut sınıfından türetilen VeganOptionDecorator, GlutenFreeDecorator, NutritionInfoDecorator sınıfları ile tarif bilgisi dinamik olarak genişletildi.



- **Sınıflar:** Recipe (interface), BasicRecipe, RecipeDecorator, VeganOptionDecorator, GlutenFreeDecorator, NutritionInfoDecorator.
- **Gerekçe:** Bu desenle tariflere ihtiyaç duyulan etiketler esnek biçimde eklendi. Böylece örneğin bir tarif hem vegan hem de glutensiz yapılabildi; tüm olasılıklar için ayrı sınıf tanımlamaya gerek kalmadı.
- **Karşılaşılan Zorluklar:** Bu örnekte sorun yaşanmasa da, decorator sayısı arttıkça hangi decorator'ın hangi sırada uygulanacağının planlanması gerektiği fark edildi.

3. Desen Adı: Composite

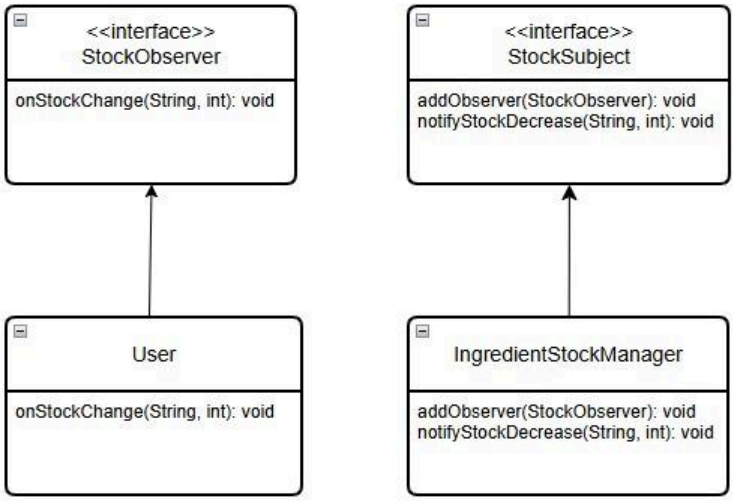
- **Problem:** Tarifleri hem bireysel olarak hem de gruplar hâlinde (örneğin "Akşam Yemeği") göstermek.
- **Çözüm:** RecipeComponent arayüzü tanımlandı. RecipeItem tekli tarifleri temsil ederken, RecipeGroup birden fazla tarif içeren grupları temsil etti. Tüm nesneler aynı arayüzü kullandığı için print() işlemi grup veya öğe fark etmeksizin uygulanabildi.



- **Sınıflar:** RecipeComponent (interface), RecipeItem, RecipeGroup.
- **Gerekçe:** Tarifleri hiyerarşik olarak gruplamak gerektiğinde bu desen büyük kolaylık sağlar. İster tek öğe, ister grup olsun, aynı metot çağrılarını kullanabilirsiniz.
- **Karşılaşılan Zorluklar:** RecipeGroup içinde alt gruplar (ana yemekler, çorbalar vb.) oluşturabilmesine izin vermek mimari olarak dikkatli bir tasarım gerektirdi.

4. Desen Adı: Observer

- **Problem:** Malzeme stoğu azaldığında kullanıcıları bilgilendirmek.
- **Çözüm:** StockSubject ve StockObserver arayüzleri tanımlandı. IngredientStockManager sınıfı gözlemcileri tutar ve bildirim gönderir. User sınıfı, stoğu izleyen ve kullanıcıya uyarı veren bir observerdır.



- **Sınıflar:** StockSubject, StockObserver, IngredientStockManager, User.
- **Gerekçe:** Malzeme yönetimi gerçek hayatta dinamik bir sistemdir. Bu desen sayesinde stok durumu değiştiğinde sistemin diğer parçalarına müdahale etmeden bildirim gönderilebildi.
- **Karşılaşılan Zorluklar:** Observer listesinin dinamik olarak yönetilmesi (ekleme/çıkarma) ve bildirimlerin tutarlı olması dikkatle yapılması gerekti.

SONUÇ

Bu proje, tasarım desenlerinin gerçek dünya problemine nasıl etkili çözümler sunduğunu gösteren bir örnek olmuştur. Factory Method ile nesne üretimi soyutlandı, Decorator ile esneklik sağlandı, Composite ile yapılar gruplandı ve Observer ile dinamik bilgilendirme gerçekleştirildi.

Bu desenleri uygularken esneklik, genişletilebilirlik ve bağımlılıkların azaltılması gibi yazılım mühendisliğinin temel prensiplerine uygun hareket edildi. Ayrıca her bir desenin sınıf yapısında neden var olduğu ve birbirleriyle nasıl etkileşim kurduğu anlaşılmıştır.

PROJEYİ HAZIRLAYANLAR:

210101027 Ümit Karadeniz
220101033 Ece Bayraktar