**Restoran Sistemi: Nesne Yönelimli Programlama (OOP) İlkeleriyle Uygulama**

Bu raporda, geliştirilmiş Restoran Sistemi kodunda kullanılan nesne yönelimli programlama (OOP) ilkeleri açıklanacak ve bu ilkelerin nerelerde uygulandığı detaylandırılacaktır.

#### 1. ****Kalıtım (Inheritance)****

1. **Açıklama:** Kalıtım, bir sınıfın başka bir sınıfın özelliklerini ve davranışlarını devralmasına olanak tanır. Bu, kodun tekrarını azaltır ve yeniden kullanılabilirliği artırır.
2. **Kodda Kullanımı:**
   1. Food ve Drink sınıfları, ortak özellikleri MenuItem soyut sınıfından devralır. Bu, her iki sınıfın Name ve Price özelliklerini paylaşmasını sağlar.

#### 2. ****Kapsülleme (Encapsulation)****

1. **Açıklama:** Kapsülleme, bir sınıfın verilerini doğrudan erişimden koruyarak sadece belirli yollarla erişime izin verir.
2. **Kodda Kullanımı:**
   1. Order sınıfı, \_items listesini özel (private) olarak tanımlar. Siparişlere yalnızca AddItem, RemoveItem ve DisplayOrder gibi yöntemlerle erişilebilir.

#### 3. ****Soyutlama (Abstraction)****

1. **Açıklama:** Soyutlama, bir nesnenin karmaşıklığını gizleyerek yalnızca önemli ayrıntıları ortaya çıkarır.
2. **Kodda Kullanımı:**
   1. MenuItem soyut sınıfı, Name ve Price gibi ortak özellikleri ve DisplayDetails soyut yöntemini içerir. Alt sınıflar (ör. Food, Drink) bu yöntemi kendilerine özel şekilde uygular.

#### 4. ****Çok Biçimlilik (Polymorphism)****

1. **Açıklama:** Çok biçimlilik, aynı arayüzün veya temel sınıfın farklı türev sınıflar tarafından farklı şekillerde uygulanmasını sağlar.
2. **Kodda Kullanımı:**
   1. DisplayDetails yöntemi, Food ve Drink sınıflarında farklı şekilde uygulanmıştır. Sistem, IMenuItem üzerinden bu yöntemleri çağırırken her nesneye uygun davranışı sergiler.

#### 5. ****Bağımlılıkların Ters Çevrimi İlkesi (Dependency Inversion Principle)****

1. **Açıklama:** Üst düzey sınıflar, alt düzey sınıflara doğrudan bağımlı olmamalıdır. Bunun yerine bir arayüz üzerinden etkileşim kurulmalıdır.
2. **Kodda Kullanımı:**
   1. Order sınıfı, doğrudan Food veya Drink sınıflarına bağımlı değildir. Bunun yerine, IMenuItem arayüzüne bağımlıdır.

**Sonuç**

Bu sistem, dört temel OOP ilkesini ve Bağımlılıkların Ters Çevrimi İlkesini tam olarak uygulamaktadır:

1. **Kalıtım:** Food ve Drink sınıflarının MenuItem sınıfından türetilmesiyle sağlanmıştır.
2. **Kapsülleme:** Verilerin doğrudan erişimden korunması Order sınıfında uygulanmıştır.
3. **Soyutlama:** MenuItem soyut sınıfı ve DisplayDetails yöntemi ile gerçekleştirilmiştir.
4. **Çok Biçimlilik:** Food ve Drink sınıflarında DisplayDetails yönteminin farklı uygulanması ile sağlanmıştır.
5. **Bağımlılıkların Ters Çevrimi:** Order sınıfının IMenuItem arayüzünü kullanmasıyla uygulanmıştır.

Bu ilkeler, sistemi esnek, genişletilebilir ve sürdürülebilir bir yapıya dönüştürmektedir.