Nesne Yönelimli Programlama (OOP) Prensipleri ve Bağımlılık Ters Çevirme Prensibi Kullanımı

Bu rapor, verilen C# kodunda uygulanan Nesne Yönelimli Programlama (OOP) prensiplerini ve Bağımlılık Ters Çevirme Prensibini (Dependency Inversion Principle) incelemektedir. Aynı zamanda ana fonksiyon içinde uygulanan Komut Satırı Arayüzü (CLI) sistemi de ele alınmıştır.

1. Kalıtım (Inheritance)

Kodda kalıtım, `PerishableItem` ve `NonPerishableItem` sınıflarının `StorageItem` sınıfından türetilmesiyle uygulanmıştır.

```
public abstract class StorageItem
```

`StorageItem` soyut bir sınıftır ve temel depo öğelerinin (name, price, quantity) ortak özelliklerini tanımlar.

```
public class PerishableItem : StorageItem
```

Bu yapı sayesinde `PerishableItem` ve `NonPerishableItem` sınıfları, `StorageItem` sınıfının tüm özelliklerini ve metotlarını devralır. Bu, kod tekrarını azaltarak yeniden kullanılabilirliği artırır.

2. Kapsülleme (Encapsulation)

Kapsülleme, sınıf içindeki verilerin doğrudan erişilememesi ve yalnızca belirli metotlarla erişilmesi prensibidir.

```
public string Name { get; private set; }
```

`StorageItem` sınıfında `Name`, `Price` ve `Quantity` özellikleri `private set` ile tanımlanmıştır. Bu sayede bu alanlara sadece sınıf içinden değer atanabilir.

```
public void UpdateQuantity(int quantity)
```

Bu metot, `Quantity` değerinin doğrudan değiştirilmesini engeller.

3. Soyutlama (Abstraction)

`StorageItem` sınıfı soyut bir sınıftır ve `CalculatePrice` metodu soyut olarak tanımlanmıştır.

```
public abstract double CalculatePrice();
```

Bu metodun gerçekleşimi, alt sınıflarda (`PerishableItem` ve `NonPerishableItem`) sağlanır. Bu, her bir sınıfın kendi fiyat hesaplama yöntemini uygulamasına olanak tanır.

4. Polimorfizm (Polymorphism)

Polimorfizm, `CalculatePrice` metodu sayesinde uygulanmıştır.

```
public override double CalculatePrice()
```

`PerishableItem` sınıfı, bu metodu kendi ihtiyacına göre yeniden tanımlar ve son kullanma tarihine bağlı olarak indirim uygular. `Inventory` sınıfındaki `DisplayItems` metodu, `StorageItem` türündeki tüm nesneleri işleyebilir.

5. Bağımlılık Ters Çevirme Prensibi (Dependency Inversion Principle)

`Inventory` sınıfı, `StorageItem` sınıfına bağımlıdır ve bu bağımlılık soyut sınıfa yapılmıştır.

```
private List<StorageItem> items;
```

Bu, kodun esnekliğini artırarak gelecekte yeni türev sınıfların eklenmesini kolaylaştırır.

Ana Fonksiyon ve CLI Sistemi

Ana fonksiyon, kullanıcıdan girdiler alarak depo sistemini yöneten bir Komut Satırı Arayüzü (CLI) içerir.

```
while (running)
```

Bu döngü, sınırsız bir işlem süreci sağlar ve kullanıcıya seçenekler sunar. `AddPerishableItem`, `AddNonPerishableItem` ve `UpdateItemQuantity` gibi

metotlar kullanıcının depo yönetimi yapmasını sağlar. Bu sistem, esnek ve kullanıcı dostu bir arayüz sunarak depo yönetimini kolaylaştırır.