Bu rapor, yukarıdaki C# kodunda kullanılan nesneye yönelik programlama (OOP) prensiplerini ve diğer yazılım tasarım ilkelerini detaylı bir şekilde açıklamaktadır.

**1. Encapsulation (Kapsülleme)**

**Kullanım Yerleri:**

* Telephone sınıfı, telefonun Brand, Model ve Price özelliklerini yalnızca okunabilir (read-only) olacak şekilde kapsülledi. Bu özelliklere yalnızca kurucu metod (constructor) aracılığıyla değer atanabiliyor ve dışarıdan değiştirilmesi engelleniyor.
* Özelliklere erişim, GetPrice ve türetilmiş sınıflarda uygulanmış GetModelDetails gibi metotlarla kontrollü bir şekilde sağlanıyor.

**Avantajları:**

* Veri bütünlüğü korunur.
* Sınıf özelliklerine yalnızca belirli yollarla erişim sağlanarak güvenlik artırılır.

**2. Inheritance (Kalıtım)**

**Kullanım Yerleri:**

* Smartphone ve FeaturePhone sınıfları, Telephone soyut (abstract) sınıfını miras alıyor.
* Tüm telefon türleri için ortak özellikler (Brand, Model, Price) Telephone sınıfında tanımlanmış ve bu özellikler alt sınıflara kalıtılmıştır.
* GetModelDetails metodu, farklı telefon türleri için özelleştirilebilir hale getirilmiştir.

**Avantajları:**

* Kod tekrarını önler.
* Sınıflar arasındaki ilişkileri düzenler ve daha okunabilir bir yapı oluşturur.

**3. Abstraction (Soyutlama)**

**Kullanım Yerleri:**

* ITelephone arayüzü (interface), tüm telefonların ortak işlevselliklerini belirlemek için kullanıldı (GetModelDetails, GetPrice).
* Telephone sınıfı soyut bir sınıf olarak tanımlandı ve her telefon türü için temel bir çerçeve oluşturdu.

**Avantajları:**

* Kodun kullanıcıya gereksiz detaylar göstermeden çalışmasını sağlar.
* Gelecekte yeni telefon türleri kolayca eklenebilir.

**4. Polymorphism (Çok Biçimlilik)**

**Kullanım Yerleri:**

* GetModelDetails metodu, her telefon türünde farklı biçimde uygulanmıştır. Örneğin, Smartphone sınıfı işletim sistemini eklerken, FeaturePhone sınıfı kamera varlığını belirtir.
* Bu yapı, MarketService sınıfındaki telefonların listeleme işlemlerinde, türden bağımsız olarak doğru bilgilerin gösterilmesini sağlar.

**Avantajları:**

* Kodun genişletilebilirliğini artırır.
* Birden fazla türde nesnenin tek bir arayüz veya temel sınıf üzerinden işlenmesine olanak tanır.

**5. Dependency Inversion Principle (Bağımlılıkların Ters Çevrilmesi İlkesi)**

**Kullanım Yerleri:**

* MarketService sınıfı, yüksek seviyeli bir modül olarak IMarketService arayüzünü uygular. Böylece, ana program doğrudan bir sınıfa bağımlı olmaz.
* IMarketService, pazar hizmetlerinin eklendiği, silindiği ve güncellendiği tüm işlevselliklerin soyut bir tanımını sağlar.

**Avantajları:**

* Bağımlılıkları soyut katmanlara çevirerek esnekliği artırır.
* Farklı pazar hizmeti sınıfları kolayca entegre edilebilir.

**6. Kodun Yönetim ve Genişletilebilirliği**

Kod, telefon pazarındaki yeni özelliklerin kolayca eklenebilmesi için tasarlanmıştır:

* **Silme İşlevselliği**: DeleteTelephone metodu, telefonları model adına göre silmek için kullanılır.
* **Güncelleme İşlevselliği**: UpdateTelephone metodu, mevcut bir telefonu güncellenmiş bir nesneyle değiştirme imkanı sağlar.
* **Komut Satırı Arayüzü (CLI)**: Kullanıcılar, Main metodunda çeşitli işlemleri kolayca gerçekleştirebilir (ekleme, listeleme, silme, güncelleme).

**Genel Değerlendirme**

Bu proje, OOP prensiplerini ve Dependency Inversion gibi modern yazılım ilkelerini etkili bir şekilde uygulamaktadır. Kod:

* Güvenli ve yeniden kullanılabilir bir yapı sağlar.
* Esnek ve genişletilebilir.
* Kullanıcı dostu bir arayüz sunar.