Bu rapor, yukarıdaki **HotelManagementLibrary** kodunda kullanılan nesne yönelimli programlama prensiplerinin (kalıtım, kapsülleme, soyutlama, çok biçimlilik ve bağımlılık tersine çevirme) nasıl uygulandığını açıklamaktadır.

### 1. ****Kalıtım (Inheritance)****

Kalıtım, bir sınıfın başka bir sınıfın özelliklerini ve davranışlarını devraldığı bir prensiptir. Bu kodda şu şekillerde uygulanmıştır:

* Room **sınıfı** soyut bir sınıf olarak tanımlanmış ve SingleRoom, DoubleRoom, ve Suite gibi alt sınıflar bu sınıfı kalıtım yoluyla devralmıştır. Bu sayede, tüm odalar ortak özelliklerini (RoomNumber, BasePrice, IsAvailable) ve davranışlarını (Book, CancelBooking) miras almıştır. Her alt sınıf, kendi özel davranışlarını (GetRoomType) ekleyerek işlevselliği genişletmiştir.
* Person **sınıfı**, Customer sınıfı tarafından kalıtılmıştır. Person temel özellikler olan Name ve Contact bilgilerini içerirken, Customer bu özelliklerin üzerine kendi davranışını eklemiştir.

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

Kapsülleme, verilerin doğrudan erişimini sınırlayıp, dış dünya ile belirli metotlar aracılığıyla iletişim kurulmasını sağlar. Kodda şu şekilde uygulanmıştır:

* Room **sınıfındaki** IsAvailable **özelliği**, dış dünyadan doğrudan değiştirilemez. Bu özellik yalnızca Book ve CancelBooking metodları ile güncellenebilir. Bu, oda durumlarının (rezerve veya müsait) güvenilirliğini sağlar.
* BookingService **sınıfında**, odalara ve rezervasyonlara erişim, yalnızca kontrollü metotlarla (AddBooking, DeleteBooking, UpdateBooking) sağlanmaktadır. Bu, veri tutarlılığını garanti eder.

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

Soyutlama, gereksiz ayrıntılardan uzaklaşarak, yalnızca önemli özelliklerin öne çıkarılmasını sağlar. Kodda şu şekilde uygulanmıştır:

* Room **sınıfı**, bir soyut sınıf olarak tanımlanmıştır ve temel odalara ait ortak özellikleri soyut bir şekilde sunar. Bu sınıftan türeyen alt sınıflar, GetRoomType metodunu kendilerine özgü olarak uygulamışlardır.
* IRoom **arayüzü**, odalara dair gerekli davranışları (Book, CancelBooking, GetRoomType) tanımlar. Böylece, farklı oda türleri için ortak bir davranış standartı sağlanmıştır.
* IBookingService **arayüzü**, rezervasyon hizmetlerinin nasıl uygulanması gerektiğini tanımlar ve uygulamanın alt detaylarından soyutlanmasını sağlar.

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

Çok biçimlilik, aynı metodun farklı sınıflarda farklı şekillerde uygulanmasını ifade eder. Kodda şu şekilde uygulanmıştır:

* GetRoomType **metodu**, Room sınıfında soyut olarak tanımlanmış ve her alt sınıf (örneğin, SingleRoom, DoubleRoom, Suite) bu metodu kendine özgü olarak uygulamıştır. Bu, farklı oda türlerinin belirli bir metod üzerinden farklı çıktılar üretmesini sağlar.
* IBookingService **arayüzü**, birden fazla farklı rezervasyon hizmeti sınıfı (örneğin, BookingService) için bir temel sağlar. Böylece farklı implementasyonlar aynı arayüz üzerinden çağrılabilir.

### 5. ****Bağımlılık Tersine Çevirme (Dependency Inversion)****

Bağımlılık tersine çevirme prensibi, üst seviye modüllerin (örneğin, iş mantığı) alt seviye modüllerden bağımsız olmasını sağlar. Kodda şu şekilde uygulanmıştır:

* HotelManager **sınıfı**, IBookingService arayüzüne bağımlıdır, doğrudan BookingService sınıfına değil. Bu, IBookingService arayüzünü implemente eden farklı rezervasyon hizmetlerinin kullanılabilmesine olanak tanır. Bu prensip sayesinde, HotelManager sınıfı, rezervasyon hizmetlerindeki değişimlere karşı daha esnek hale gelmiştir.
* Aynı şekilde, BookingService sınıfı da odalarla IRoom arayüzü üzerinden etkileşime geçmektedir. Bu, herhangi bir IRoom implementasyonunun (SingleRoom, DoubleRoom, vb.) BookingService ile çalışmasını sağlar.

### Sonuç

Bu proje, nesne yönelimli programlama prensiplerinin doğru bir şekilde uygulandığı kapsamlı bir örnektir. **Kalıtım**, kodun yeniden kullanılabilirliğini artırırken, **kapsülleme**, veri güvenilirliğini sağlamıştır. **Soyutlama**, uygulamanın detaylardan soyutlanmasını sağlarken, **çok biçimlilik**, genişletilebilir ve esnek bir yapı oluşturmuştur. Son olarak, **bağımlılık tersine çevirme prensibi**, üst seviye modüllerin alt seviye modüllerden bağımsız olmasını sağlayarak modüler ve sürdürülebilir bir sistem yaratmıştır.