**State (Durum) Pattern (Kalıbı)**

• **Intent**: Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.

• **Amaç**: Bir nesnenin iç durumu değiştiğinde davranışının değişmesine izin ver. Nesne sanki sınıfı değişiyormuş gibi görünür.

**PROBLEMS**

• Uygulamalarda sıklıkla karmaşık durumlara sahip nesnelerle karşılaşırız.

• Bu nesnelerin davranışları da durumuna bağlı olarak değişir.

• Örneğin bir shopping cart nesnesine yeni mal ekleyip ekleyememek ya da bir banka hesabından para çekilebilmesi ya da ödeme yapılabilmesi gibi işlemler, bu nesnelerin durumuna bağlıdır.

• **GoF**, **TCPConnection** nesnesini örnek **verir**.

• Bu nesnenin **Established,** **Listening** ve **Closed** gibi durumları vardır.

• **TCPConnection** nesnesi diğer nesnelerden mesaj aldığında hangi durumda olduğuna bağlı olarak farklı şekillerde cevap verir.

• Örneğin **acknowledge**() çağrısına **Closed** durumunda **Established** durumundaki gibi cevap vermez.



• Tipik olarak yapılan, nesnenin durumunu farklı alanlarla ifade etmek ve nesne üzerinde yapılan metot çağrılarına duruma bağlı cevapları **ifelse** ya da **switch**-**case** yapılarıyla yönetmektir.

• Böyle bir yaklaşımda, metotlar gittikçe karmaşıklaşır.

• Hatta iş süreçlerinin ve kurallarının sayıca gittikçe artması ve karmaşıklaşmasından dolayı bu tür duruma bağlı davranışın kodlandığı metotlar bir müddet sonra içinden çıkılmaz hale gelir.

// • Nesnelerin karmaşık durumları karmaşık iş mantığını gerektirir.

//

// • İş mantığında durumlar arasında geçişlerin de yönetilmesi gerekir.

//

// • Dolayısıyla çok sayıda ve karmaşık if-else karar mekanizmaları söz konusu olur.

**SOLUTION**

• Böyle durumlarda daha rahat bir çözüm State kalıbıyla uygulanabilir.

• State kalıbında, nesnenin durumları ayrı ayrı ele alınır ve bir arayüzü gerçekleştiren sınıflarda diğer durumlardan yalıtılır.

• Bu şekilde her durum için ayrı bir sınıf oluşturulur ve duruma bağlı davranışların, içinde bulunulan duruma özgü halleri burada gerçekleştirilir.

• Sınıfların ortak davranışları ise üstteki tipte soyut olarak toplanır.

• Durumlu nesnelerin durumları, durumlar arasındaki geçişler, geçişler

için gerekli şartlar (guard condition) ve geçişlere sebep olan olaylar

(event) **bir sonlu durum makinası (Finite-State Machine, FSM)**

olarak modellenebilir.

**DURUM YONETIMI ( State Management)**

• **State** kalıbıyla uygulanmasından en temel karar noktası durumların yönetimiyle ilgilidir.

• **Nesnenin hangi durumda olduğuna kim karara verecek ve durumlar**

**arasındaki geçişleri (state transition) kim yönetecek?**

**• İki alternatif olabilir:**

**• Durum değişimleri durumların kendileri tarafından yönetilir.**

**• Durum değişimleri merkezi olarak yönetilir.**

**• Bu durumda merkezi bir nesne, arabulucu (mediator) gibi davranarak,**

**durumlardan gelen bilgiye göre bir sonraki durumu belirler.**

**• İlk alternatifte durum yönetimi dağıtılmış, ikinci alternatifte ise**

**merkezileştirilmiştir.**

• İlk alternatifte durum değişimleri, durumlar tarafından yönetildiğinden

her durum, hangi olayın kendisinden diğer bir duruma geçişe sebep

olacağını bilmelidir.

• Bu çözümde durumlar arası geçiş yönetimi durumlara dağıtıldığından

daha karmaşık ve çok sayıda durumun olduğu hallerde daha rahat

olacağı düşünülebilir.

Durum yönetiminde **Observer** kalıbı kullanılabilir.

Özellikle durum geçişlerinin merkezi olarak yönetildiği çözümde

durumları temsil eden nesneler, değişiklik halinde event fırlatarak

kendilerini dinleyen merkezi nesneye durumu haber verirler.

**SONUCLAR**

• Duruma bağlı davranışilar ayrı nesnelerde ele alındığından karmaşıklık azalır.

• Aksi halde **if-else if vb**. yapılarla bir yerde toplanacak olan duruma bağlı davranış mantığı

• Durumlar arası geçişler çok daha belirgin hale gelir.

• Durum nesneleri paylaşılabilir.

• Yeni durumları da sisteme katmak daha kolay hale gelir.

• Her yeni durum için bir alt sınıf oluşturulur ve durumlar arası geçiş

mantığı güncellenir.

• Alexander Ran’ın MOODS Models for Object-Oriented Design of

State başlıklı makalesinde state kalıbının tasarımıyla ilgili pek çok

karar noktası tartışılmaktadır.

<http://www.soberit.hut.fi/tik-76.278/alex/plop95.htm>

**DIGER KALIPLARLA ILISKI**

• Durum nesneleri **Singleton** olabilir.

• Durum nesneleri paylaşım için **Flyweight** olarak modelenebilir.

• Durum geçişlerinde **Observer** kalıbı kullanılabilir.

**UYGULAMA**

**State kalıbını kullanarak bir klimanın (AC, AirCondition) durumlarını modelleyin.**