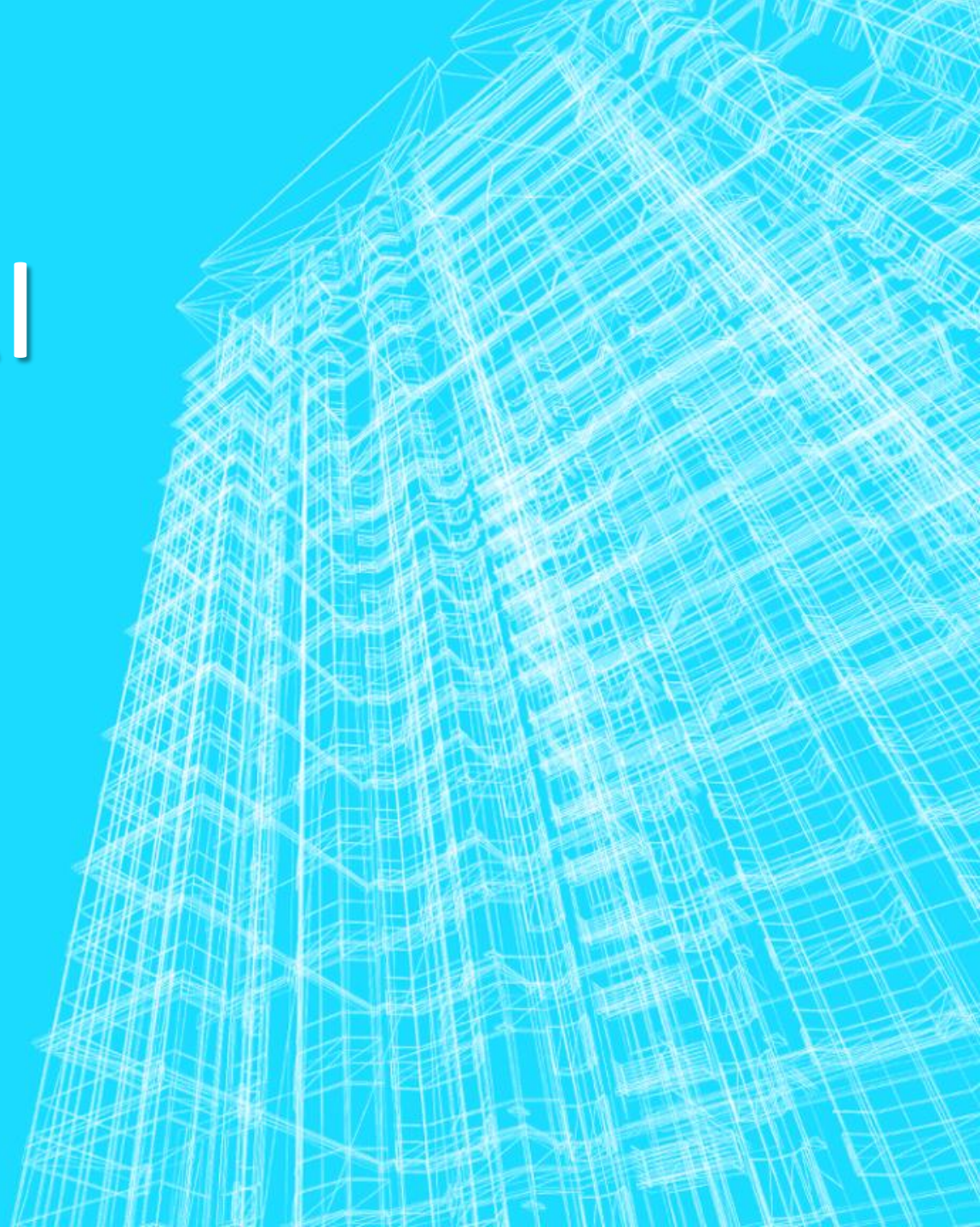
TASARIM KALIPLARI

ÇAĞLAR TELEF



TASARIM KALIPLARI

Tasarım Kalıpları ?

- Yaratıcı Tasarım Kalıpları
- Yapısal Tasarım Kalıpları
- Davranışsal Tasarım Kalıpları

Sorular ?

TASARIM KALIPLARI

Yazılım tasarımında sürekli karşılaşılan genel sorunlara esnek, yeniden kullanılabilir, başarılı çözümler getiren bir takım hazır kalıplardır.

Nesneye dayalı programlamada, tasarım desenleri sınıf ve nesneler arasındaki ilişkilerin en iyi şekilde nasıl olmaları gerektiğini açıklayan yöntemlerdir.

NEDEN TASARIM KALIPLARI

Daha önce sınanmış, ispatlanmış ve yararı görülmüş yazılım yöntemleri önerdiği için geliştirme sürecini hızlandırır. Yeniden kullanılabilir, esnek çözümler yazılımın büyümesini, genişlemesini kolaylaştırır. Tasarım kalıplarını bilen, yazılımcı ve yazılım mimarları için kod okunabilirliği yüksek olur. Ekip içerisinde tasarım kalıplarıyla ortak dil kullanılmış olur.

YARATICI TASARIM KALIPLARI

Nesnelerin uygun ve verimli bir şekilde yaratılma yollarıyla ilgilenirler.

Nesne yaratma sorumluluklarını, farklı sınıf düzenlerinde soyutlayarak yaparlar.

YARATICI TASARIM KALIPLARI

a. Abstract Factory

b. Builder

c. Factory Method

d. Prototype

e. Singleton

YARATICI TASARIM KALIPLARI

Abstract Factory: Yaratılma sorumluluklarının çeşitli sınıf hiyerarşileri ile sağlandığı bir tasarım kalıbıdır. İstemci sınıf, çeşitli kategorideki sınıflara ihtiyaç duyar. Bu sınıflar, sistemde alt/üst sınıf hiyerarşileri şeklinde bulunurlar. Abstract factory tasarım kalıbı bu karmaşık ve farklı hiyerarşideki sınıfların yaratılma sorumluluğunu çeşitli fabrika sınıflarına verir.

YARATICI TASARIM KALIPLARI

Abstract Factory:

- Çalıştığı platformdan bağımsız bir uygulama geliştirmek istiyoruz.
- Benzer özelliklere sahip objelerimiz var.
- Benzer objelerin yaratılma mantığını, objelerin yaratılacağı sınıfları tek tek belirtmeden interface bağlı gruplamak istiyoruz.

YARATICI TASARIM KALIPLARI

Builder: Tasarım kalıbı ile bir nesneyi, bir sınıfı, aşama aşama farklı özellikleri ile oluşturabiliriz.

- Kompleks yapıda bir objemiz var.
- Objeyi oluşturmak için belirli bir sırayı takip etmemiz gerekli.

YARATICI TASARIM KALIPLARI

Factory Mothod: Tasarım kalıbı, nesne yaratma sorumluluğunun bir yordama verilmesidir. Yaratılan nesne, bir sınıf hiyerarşisindeki alt sınıflardan biridir. Hangi alt sınıfın yaratılacağı kararı factory method içinde verilir.

- Oluşturacağımız objenin tipini önceden belirtmek istemiyoruz.
- Oluşturulacak objenin tipi runtime değişiklik gösterecek ise.

YARATICI TASARIM KALIPLARI

Prototype: Var olan bir nesneden, kopyalama yöntemi ile yeni nesne yaratmak için bu tasarım kalıbı kullanılır. Nesne yaratmak için "new" operatörü kullanılmaz. Yazılım dillerindeki "clone" gibi, nesne kopyalama yordamlarından faydalanılır.

- Bir objenin özelliklerine sahip olan aynı tipte bir obje yaratmak istiyoruz.

YARATICI TASARIM KALIPLARI

Singleton: Tasarım kalıbında, bir sınıfın sistem içinde yalnızca bir tane nesnesi oluşturulabilir. Tek bir arayüz sunularak, bu nesneye yalnızca buradan erişim sağlanabilir.

- Yalnızca 1 objeye ihtiyacımız var
- Objeye her sınıftan erişebilmemiz gerekli
- Obje ihtiyacımız olana kadar yaratılmasın.

YAPISAL TASARIM KALIPLARI

Bu sınıfta yer alan tasarım kalıpları, sınıflar arasındaki ilişkileri belirleyerek, tasarımı kolaylaştırırlar. Sınıflar arası ilişkiler nasıl olmalı? Türetme, soyutlama, nesnesini içerme? Hangi yol ile sınıflar birbirlerine bağlı olmalı. Bu soruların yanıtlarını açıklarlar.

YAPISAL TASARIM KALIPLARI

a. Adapter

b. Composite

c. Decorator

d. Facade

e. Flyweight

f. Proxy

YAPISAL TASARIM KALIPLARI

Adapter: Mevcut bir sınıfı veya arayüz sınıfını, eldeki farklı bir arayüz sınıfına uygun hale getirerek tekrar kullanmak amacıyla uygulanır.

- Birbirleriyle uyumsuz sınıfları çalışabilir hale getirmek istiyoruz.

YAPISAL TASARIM KALIPLARI

Composite: Tasarım kalıbının amacı, nesneleri ağaç yapısına göre düzenleyerek, ağaç yapısındaki alt üst ilişkisini kurmaktır. Bu tasarım kalıbına göre, ağaç yapısındaki üst ve alt nesneler aynı arayüz sınıfından türeyerek, birbirlerine benzerler.

- Basit objelerden ve bu basit objeleri barındıran kompleks objelerden oluşan bir yapı istiyoruz.

YAPISAL TASARIM KALIPLARI

Decorator: Tasarım kalıbı, nesneye ek özellikler eklemek için kullanılır. Nesnenin özeliklerini arttırmak için, temel nesneden türetilip de, yeni nesneler yaratılmasına gerek yoktur. Yani bu desen, bir nesneye alt sınıflar yaratılmaksızın, dinamik olarak yeni özellikler kazandırmak için kullanılır.

- Objelere yeni özellik kazandırmak istiyoruz.

YAPISAL TASARIM KALIPLARI

Facade: Tasarım kalıbı, sistemin detaylarını saklayarak, istemcinin dışarıdan sisteme ulaşabilmesi için tek bir ön yüz sunar. Sistemdeki alt sınıflara, bu ön yüz sınıfı ile ulaşılır.

- Birçok sistemi bir üst sistemde toplayarak bu alt sistemlere erişim kolaylığı sağlamak istiyoruz.

YAPISAL TASARIM KALIPLARI

Flyweight: Bu tasarım kalıbı çok sayıda nesnenin sistemde olduğu durumlarda, nesne sayısının sistemde problemlere neden olmaması için kullanılan tasarım desenidir.

- Birbirine benzer çok sayıda obje oluşturmak istemiyoruz.

YAPISAL TASARIM KALIPLARI

Proxy: Yaratılması pahalı bir çok işlem yapan bir nesneyi taklit eden bir başka nesnenin kullanılmasıdır.

- Bol bol kaynak tüketen bir objeye sahibiz.
- Bu sebeple objeyi kullanmadan önce initialize etmek istemiyoruz.

DAVRANIŞSAL TASARIM KALIPLARI

a. Chain of Responsibility

b. Command

c. Interpreter

d. Iterator

e. Mediator

f. Memento

g. Observer

h. State

i. Strategy

j. Template Method

k. Visitor

DAVRANIŞSAL TASARIM KALIPLARI

Chain of Responsibility: bir dizi işlev sınıflarıyla, bu sınıfların işlevlerini başlatmak için gereken komut sınıflarından oluşur. İşlev sınıfları, ne tür işler yapacağını kendi bünyesinde tutar, ayrıca dizideki diğer bir işlev sınıfının ne olacağını da belirler. Bir işlemin, belli miktarda işlevlerden sırayla geçmesi gerektiğinde bu tasarım deseni kullanılabilir.

DAVRANIŞSAL TASARIM KALIPLARI

Chain of Responsibility:

- Bir istek sonucu oluşacak etkiyi gerçekleştirecek yapıların birbirlerine bağlanmasıyla oluşur.
- Birçok yapıdan yalnıza biri yapılacak işi gerçekleştirir.
  - «Atm den çekilen paraya göre banknot vermesi gibi.»

DAVRANIŞSAL TASARIM KALIPLARI

Command: İşlemlerin nesne haline getirilip başka bir nesne(invoker) üzerinden tetiklendiği bir tasarım desenidir.

- Yapılacak bir çağrı için tüm bilgiyi bir objede tutmak istiyoruz.
- Alakalı komut istenen zamanda sıralı işlemler içeren bir batch şeklinde çağrılabilir.

DAVRANIŞSAL TASARIM KALIPLARI

Interpreter: İşlemlerin nesne haline getirilip başka bir nesne(invoker) üzerinden tetiklendiği bir tasarım desenidir.

- Belirli kural kümelerini bir gramerle ifade etmek istiyoruz.
- Bu kural kümelerini anlamlı ifadeye çeviren bir tercüman ihtiyacı duyuyoruz.
  - Müziğin notayla ifade edilmesi
  - SQL ifadeleri

DAVRANIŞSAL TASARIM KALIPLARI

Iterator: Nesne koleksyonlarının elemanlarını belirlenen kurallara göre elde edilmesini düzenler.

- Sırasıyla dolaşmak istediğimiz bir araya gelmiş obje kümemiz var.
- Amacımız yalnızca sırayla dolaşmak, nasıl dolaştığımızın detaylarını bilmemize gerek yok.

DAVRANIŞSAL TASARIM KALIPLARI

Mediator: sınıfların sayısı arttıkça, aralarındaki bağımlılıklar artar, uygulamanın bakımı ve değiştirilmesi zorlaşır. Mediator tasarım kalıbı, bu sınıflar arasındaki bağımlılıkları azaltmak ve aralarındaki iletişimi kolaylaştırmak için kullanılır.

- Sınıfların birbirleriyle direkt konuşmasını istemiyoruz ki sınıflar arası dependency oluşturmak durumunda kalmayalım.
    - «Hava trafik kontrol gibi»

DAVRANIŞSAL TASARIM KALIPLARI

Memento: nesnenin durumunu saklayıp, daha sonra bu duruma tekrar geri dönebilmesini sağlayan tasarım desenidir.

- Objenin bir iç durumuna ait değerler için checkpoint oluşturmak istiyoruz.
- Oyun karakterimiz öldüğünde en son kaydedilen checkpoint e dönmek istiyoruz.

DAVRANIŞSAL TASARIM KALIPLARI

Observer: nesnenin durumlarında değişiklik olduğunda, bu değişikliklerden haberdar olmak isteyen diğer nesnelere haber verilmesi gerektiği durumlarda bu tasarım deseni kullanılır.

- Bir objeyle alakalı değişim olduğu zaman, bu değişimden belirli objeleri haberdar etmeyi amaçlıyoruz.
- Dinamik biçimde değişikliğin takibini yapacak objeleri ekleyip çıkartmak istiyoruz.

DAVRANIŞSAL TASARIM KALIPLARI

State: nesnenin durumu değiştiğinde, davranışı da değişiyorsa, yani nesneler farklı durumlarda, farklı davranışlar gösteriyorsa, durum tasarım deseni kullanılabilir.

- Objenin davranış biçimini kendine ait olan belirli durumlara göre dinamik biçimde değiştirmesini istiyoruz.

DAVRANIŞSAL TASARIM KALIPLARI

Strategy: bir işlem için farklı yöntemlerin uygulanabilir olduğu durumlarda, bu yöntemi kullanacak olan nesne, hangi yöntemin uygulanacağını seçer. Çünkü bu içerik nesnesi, yöntemleri belirleyen üst sınıfı içerir. Farklı yöntem veya strateji alt sınıfları da, bu üst sınıftan türerler.

DAVRANIŞSAL TASARIM KALIPLARI

Strategy:

- Bir sınıfa ait davranışlar dinamik olarak değişebiliyorsa bu davranışlara algoritmaları ayrıştırarak saklamamız mantıklı olacaktır.
- Böylece runtime'da davranışları karışıklık yaşamadan değiştirebilir hale getireceğiz.

DAVRANIŞSAL TASARIM KALIPLARI

Template Method: bir algoritmanın adımlarının abstract sınıfta tanımlanarak farklı adımların concrete sınıflarında overwrite edilip çalıştırılmasını düzenler.

- Bir sınıfa ait birbirine benzer yapıya sahip algoritmalar da yalnızca bazı adımları değiştirmek istiyoruz.

DAVRANIŞSAL TASARIM KALIPLARI

Visitor: tasarım deseni, çok sayıda ve farklı tipteki nesneler üzerinde işlem yapabilmek amacıyla kullanılır. İşlem yapılacak nesnelerde herhangi bir değişiklik yapılmaz. İşlemi ziyaretçi nesneleri yapar. Eğer sisteme yeni nesneler eklenmiyor, fakat sık sık yeni işlemlerin eklenmesi gerekiyorsa bu tasarım deseni kullanılabilir.

DAVRANIŞSAL TASARIM KALIPLARI

Visitor:

- Sınıflarımıza yeni bir metot eklemek istiyoruz.
- Eklenmesi gereken sınıflara tek tek manuel biçimde ekleyip kod bakımını zorlaştırmak istemiyoruz.
- Birden fazla sınıfın implement ettiği interface'e ekleyip tek tek sınıflarda implementasyonu gerçekleştirmek istemiyoruz.

KAYNAKÇA

http://www.tasarimdesenleri.com/jsp/tasdes/tasdesnedir.jsp

https://sourcemaking.com/design_patterns

TEŞEKKÜRLER

ÇAĞLAR TELEF
2017