# Nesne Yönelimli Analiz ve Tasarım

## Nesne Tasarımı

Tasarım Desenleri

Design Patterns

# Tasarım Desenleri

* Yazılımlar geliştirilirken karşılaşılan genel tasarım problemlerini tanımlayarak, bu problemin en uygun nasıl çözülebileceğini (kod tekrar kullanımını artırmak ve değişikliği kolaylaştırmak için) ve çözümün ortaya çıkaracağı sonuçları anlatır.
* Programlama dillerinden bağımsızdır.
* Yazılım geliştiricilerin tasarımlarla ilgili tartışma yapmasını kolaylaştırır.
* Tasarım desenleri dört bölümden oluşur
  * Desenin adı
  * problemin tanımı: desenin ne zaman/nerede kullanılabileceği
  * çözüm: problemin nasıl çözüleceği (kullanılması gereken bileşenler, aralarındaki bağıntı vb.)
  * sonuç: deseni uygulamanın sonuçları?

Gamma, E. vd. (1994). "Design Patterns:Elements of Reusable Object-Oriented Software "

# Tasarım Desenleri

* Creational(Nesne oluşturma): Nesnelerin uygun bir şekilde oluşturulmasını sağlayacak mekanizmalar içerir.
* Structural(Yapısal): Nesnelerin sistemler içerisine uygun olarak yerleştirilmesini sağlayacak desenlerdir.
* Behavioral(Davranışsal): Nesnelerin birbirleriyle uygun olarak etkileşiminini düzenleyen mekanizmalar.

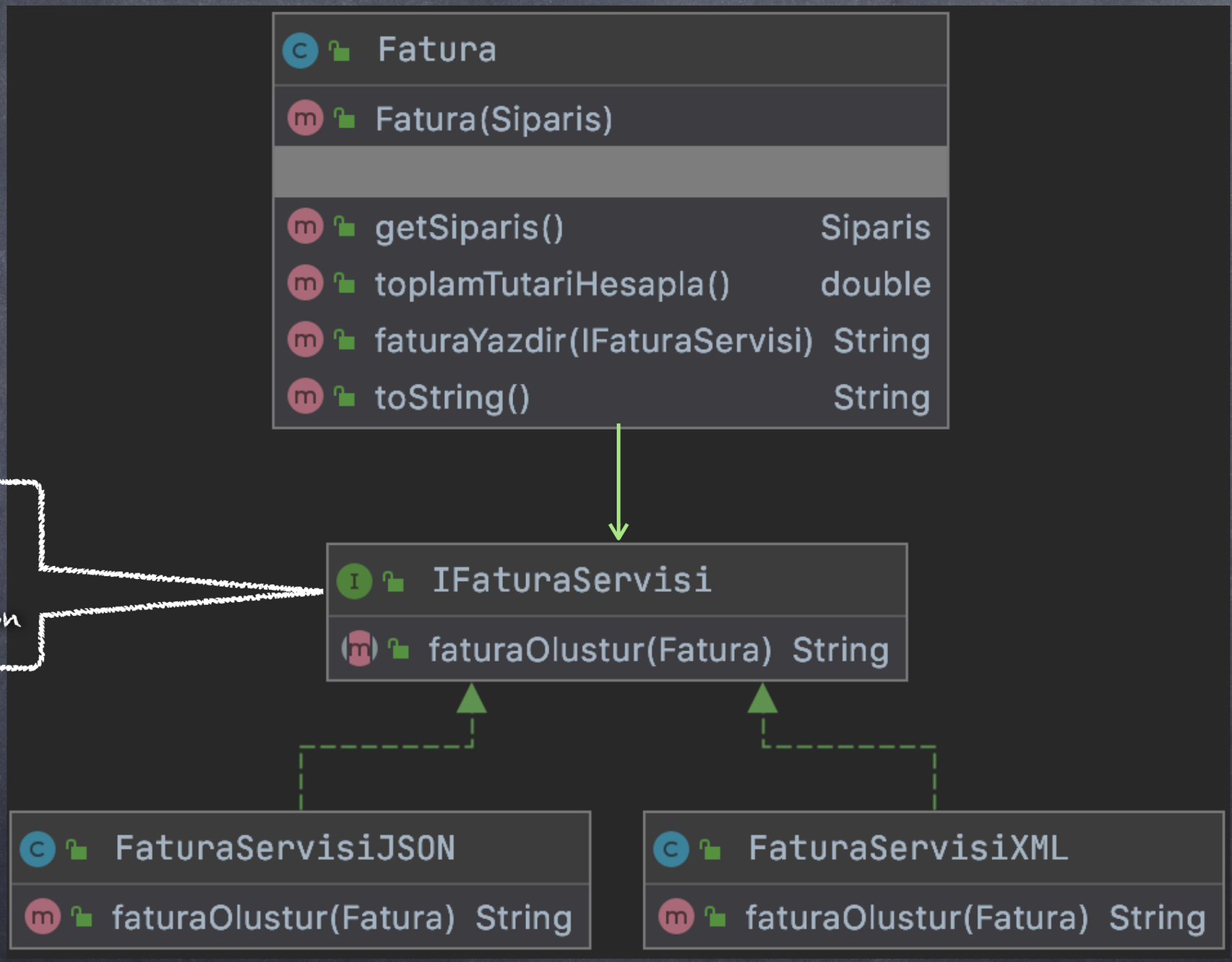
| Creational (Nesne oluşturma) | Structural (Yapısal) | Behavioral (Davranışsal) |
|---|---|---|
| Singleton Pattern | Adapter Pattern | Template Method Pattern |
| Factory Pattern | Composite Pattern | Mediator Pattern |
| Abstract Factory Pattern | Proxy Pattern | Chain of Responsibility Pattern |
| Builder Pattern | Flyweight Pattern | Observer Pattern |
| Prototype Pattern | Facade Pattern | Strategy Pattern |
| | Bridge Pattern | Command Pattern |
| | Decorator Pattern | State Pattern |
| | | Visitor Pattern |
| | | Interpreter Pattern |
| | | Iterator Pattern |
| | | Memento Pattern |

Gamma, E. vd. (1994). "Design Patterns:Elements of Reusable Object-Oriented Software "

# Tasarım Desenleri : Strategy

* Davranışsal desenlerden biridir.
* Aynı istemci kodun, farklı algoritmaları/stratejileri desteklemesini sağlar.
* Örneğin; geliştirdiğiniz uygulama "buble sort" algoritmasını kullanıyorken, istemci kodu değiştirmeden, bu algoritma yerine "quick sort" algoritmasını çalıştırmasını sağlamak isterseniz, bu deseni kullanabilirsiniz.
* Farklı algoritmalar/stratejiler soyut bir modülden (arayüz) türetilir/gerçeklenir.
* İstemci kod içerisinde, bu algoritmalar yerine soyut modül kullanılır (program to interface...)

# Tasarım Desenleri : Strategy

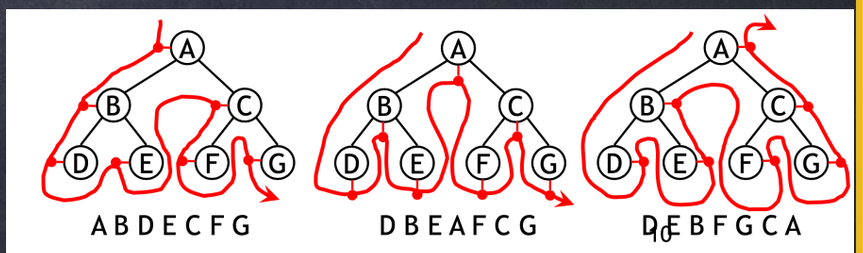



OCP
Open for extension,
Closed for modification

https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders12

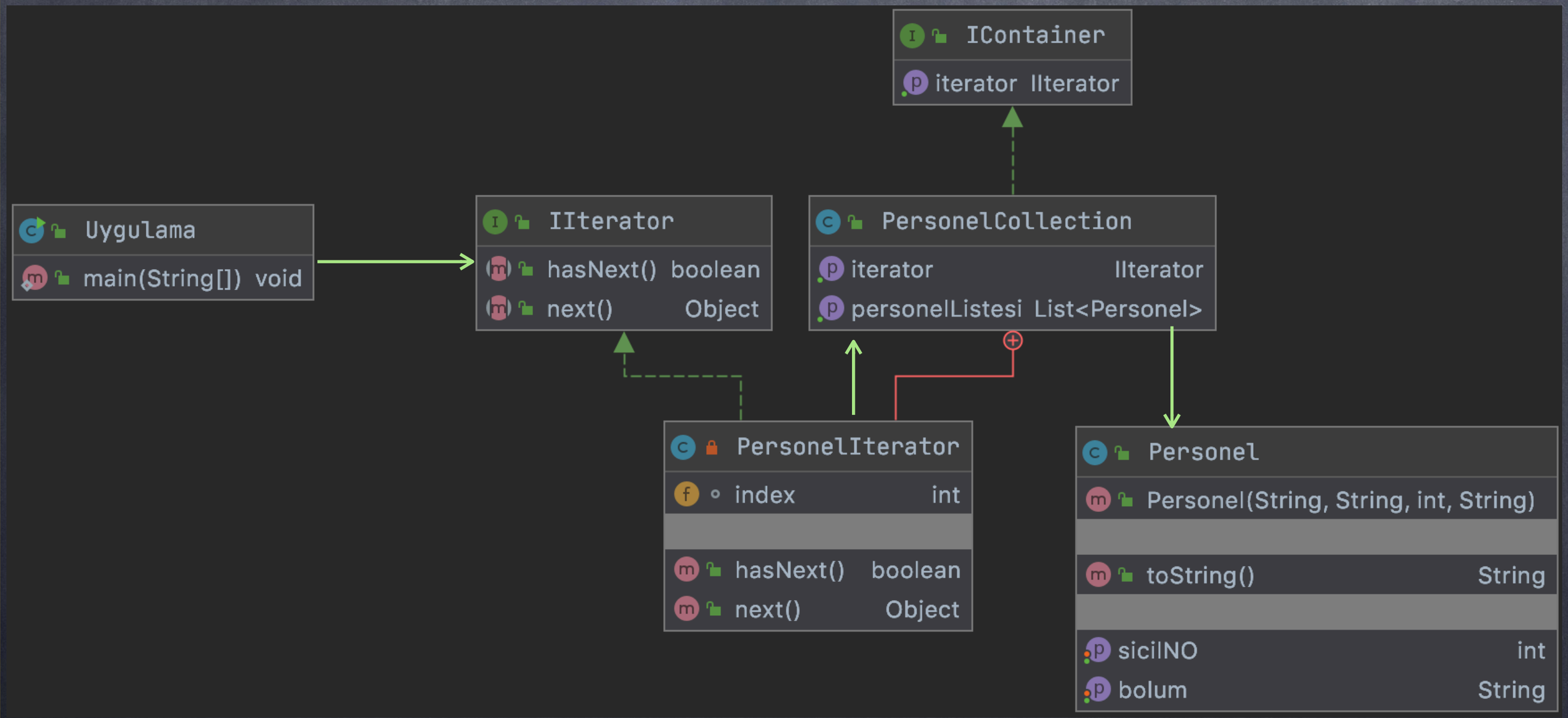
# Tasarım Desenleri : Iterator

* Davranışsal desenlerden biridir.
* Veri toplulukları (collections) içerisinde elementler (nesneler) bulunur ve bu elementler çeşitli veri yapıları (dizi, bağlı liste, ağaç, graf vb.)kullanılarak bir arada tutulur.
* Veri toplulukları içerisindeki her bir elemente erişilmesi gerekir. Bu işleme dolaşım (traversal) denir.
* Veri toplulukları için kullanılan veri yapıları basit olduğunda (dizi, liste vb.) dolaşım için kullanılacak algoritma basit bir şekilde gerçekleştirilebilir.
* Kullanılan veri yapıları karmaşıklaştıkça dolaşım algoritmaları zorlaşabilir ve zaman zaman farklı dolaşım algoritmalarına (ağaç veri yapısı için preorder, postorder, inorder gibi) ihtiyaç duyulabilir.
* Iterator deseni, istemci modülün, veri topluluğu içerisinde kullanılacak dolaşım algoritmalarından etkilenmesini önlemek için, bu işlemi (sorumluluk) başka bir nesnenin (iterator) nesnesinin (SRP gereği) yapmasını ister.




www.cis.upenn.edu/~matuszek/cit594-2008 adresinden alınmıştır.

# Tasarım Desenleri : Iterator

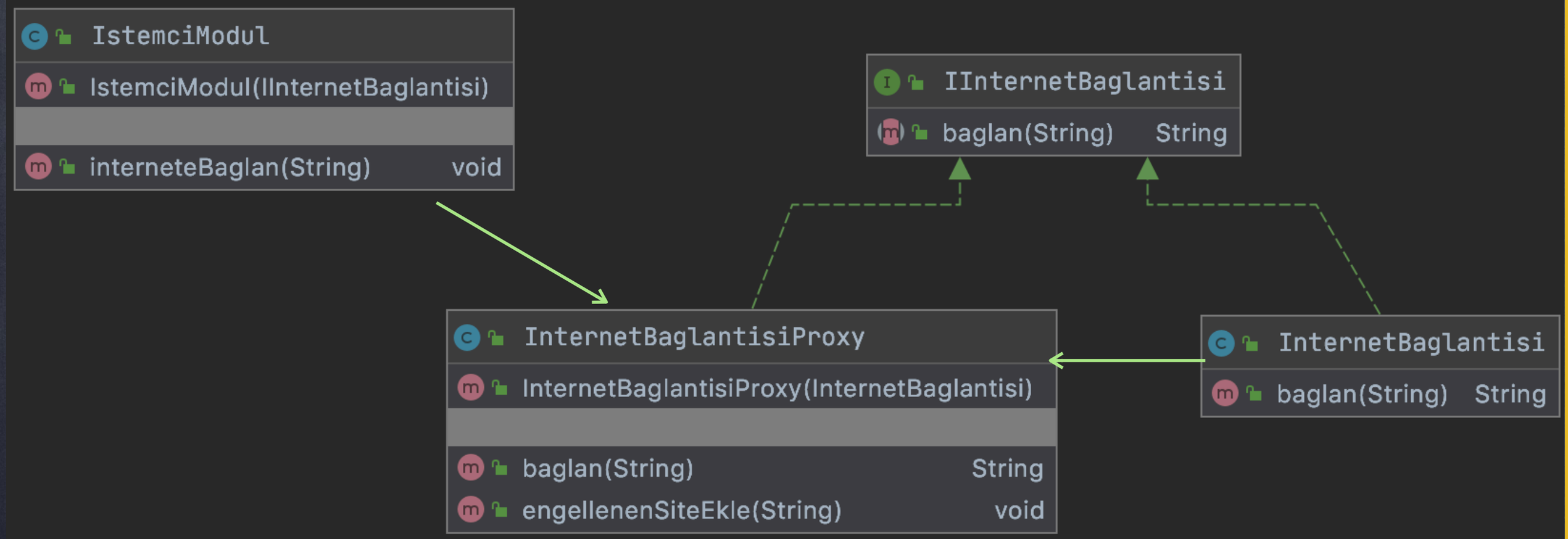



https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders12

# Tasarım Desenleri : Proxy

* Yapısal desenlerden biridir.
* Sınıfın fonksiyonlarını (arayüzünü) değiştirmeden davranışını değiştirmek istediğimizde kullanabiliriz.
* Sınıfı değiştirmek, onu kullanan diğer sınıfların etkilenmesine neden olabilir.

# Tasarım Desenleri : Proxy



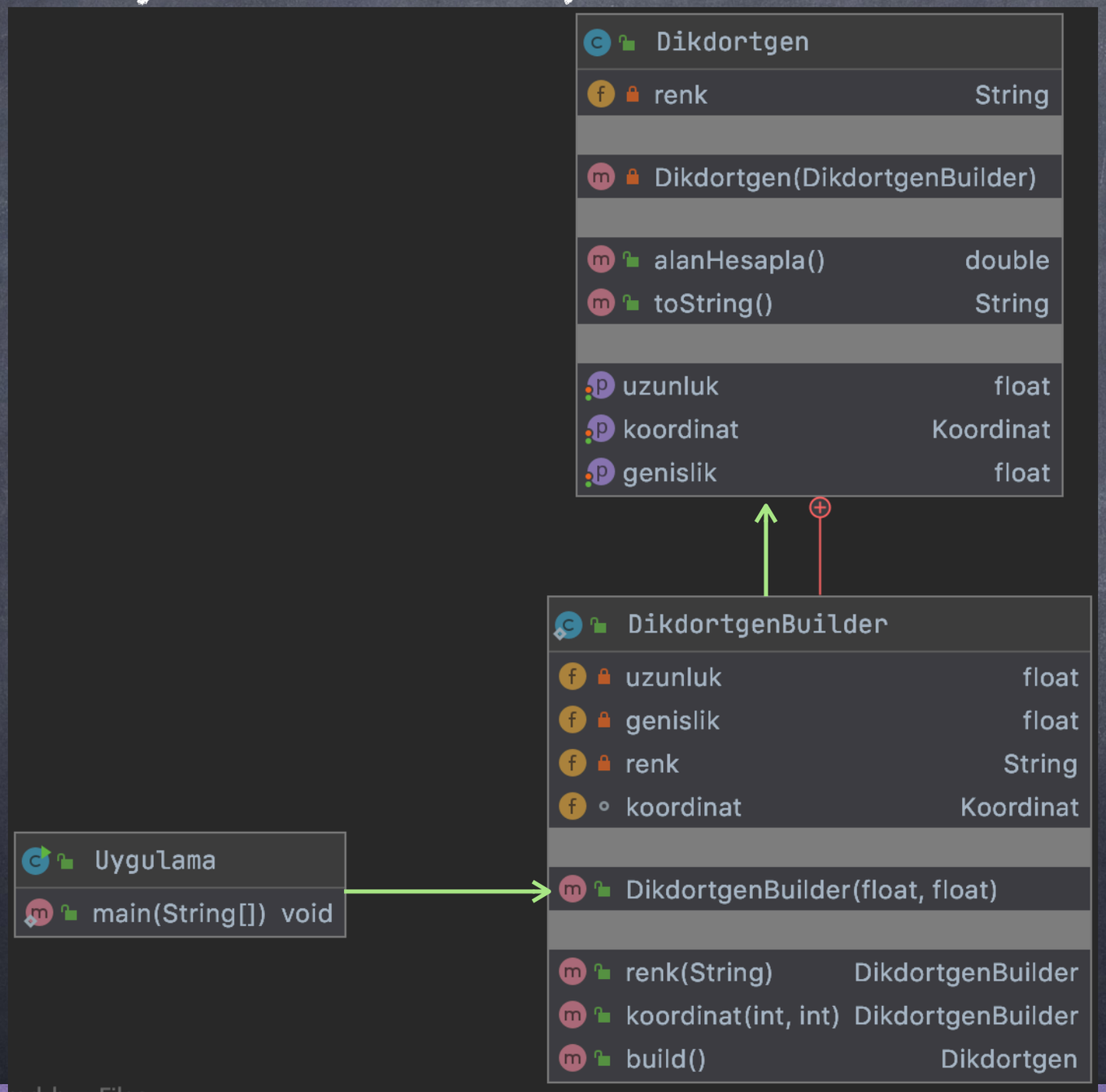

https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders12

# Tasarım Desenleri : Builder

* Nesne oluşturmayla (creational) ilgili desenlerden biridir.
* Karmaşık nesnelerin (içerisinde çok sayıda üye değişken ve üye nesne olan) oluşturulması için kullanılır.
* Karmaşık bir nesnenin yapımını, temsilinden (sunumundan) ayırın, böylece aynı yapım süreci farklı temsiller oluşturabilir.
* Nesnelerin farklı temsillerinin (sunumunlarının) her biri için ayrı ayrı yapıcı tanımlamak yerine, nesne oluşturma işini adım adım gerçekleştiren "builder" deseni kullanılabilir.
    * Böylece nesne oluşturma işi nesnenin kendisinden ayrılmış olur (SRP).
    * Nesne oluşturma işlemi istemci koddan ayrılmış olur (SRP, loosly coupling)

# Tasarım Desenleri : Builder

* Builder sınıfı nesnenin tüm üye değişkenlerini/nesnelerini içeriyor.
* Nesne oluşturmak gerektiğinde, builder sınıfının (static olmalı) nesnenin ilgili özelliklerine ilk değer ataması yapan yöntemleri sırasıyla çağrılıyor.
* Çağrılan son yöntem (build) Dikdörtgen nesnesini oluşturuyor. Nesne oluşturmak için, varsayılan yapıcısına, nesnenin ilgili üyelerinin değerlerini içeren DikdortgenBuilder sınıfını gönderiliyor.



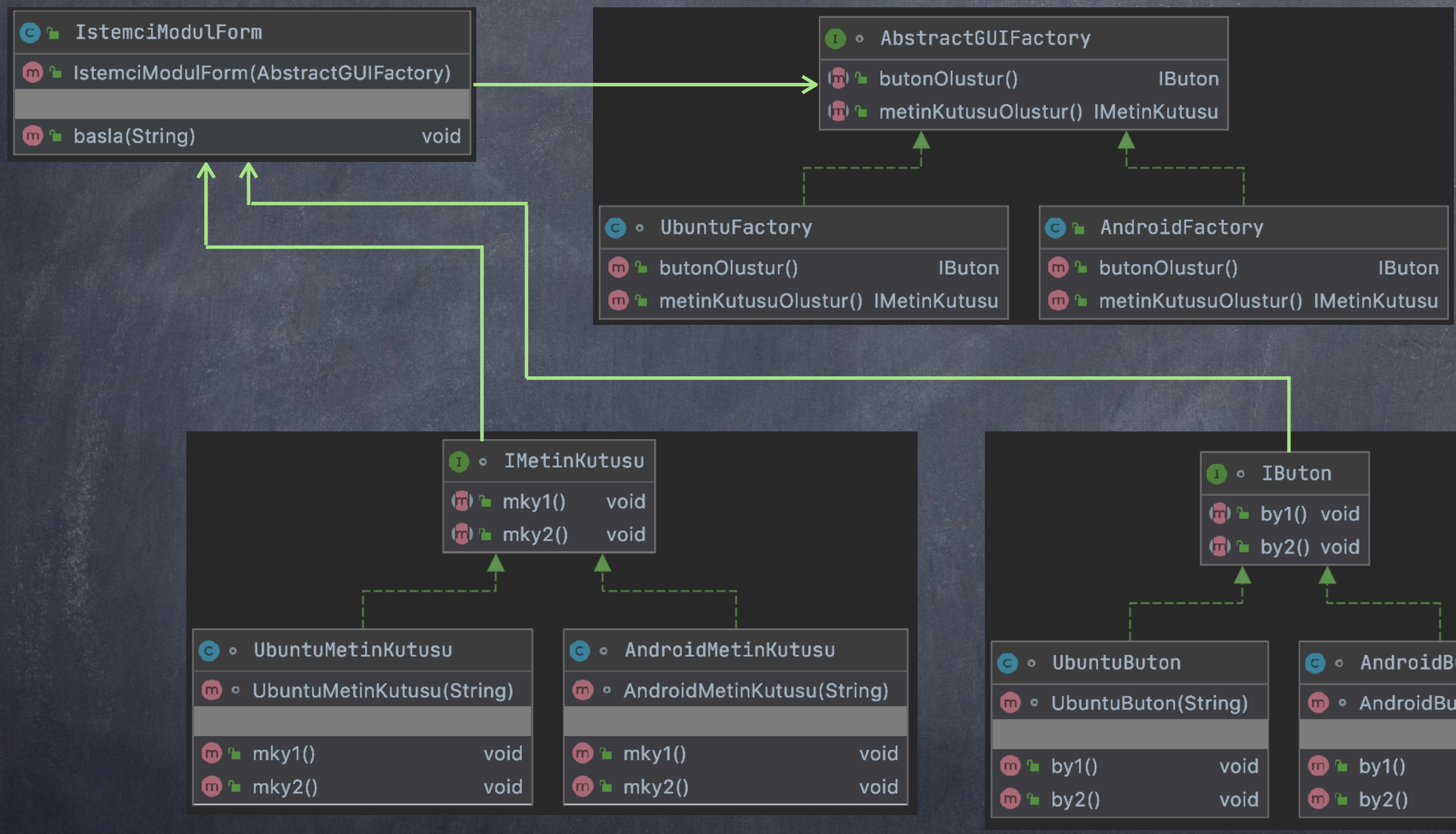

https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders12

# Tasarım Desenleri : Abstract Factory

* Nesne oluşturmayla (creational) ilgili desenlerden biridir.
* İstemci modül içerisinde, somut sınıflarını belirtmeden, birbirleriyle ilgili ya da birbirlerine bağlı nesne aileleri/ grupları oluşturmak için kullanılabilir.
* Örneğin; bir geliştirme ortamının görünümünü değiştirmek istediğimizde ya da bir yazılımın farklı platformlarda sorunsuz çalışabilmesini istediğimizde bu deseni kullanabiliriz.

https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders12

# Tasarım Desenleri : Abstract Factory



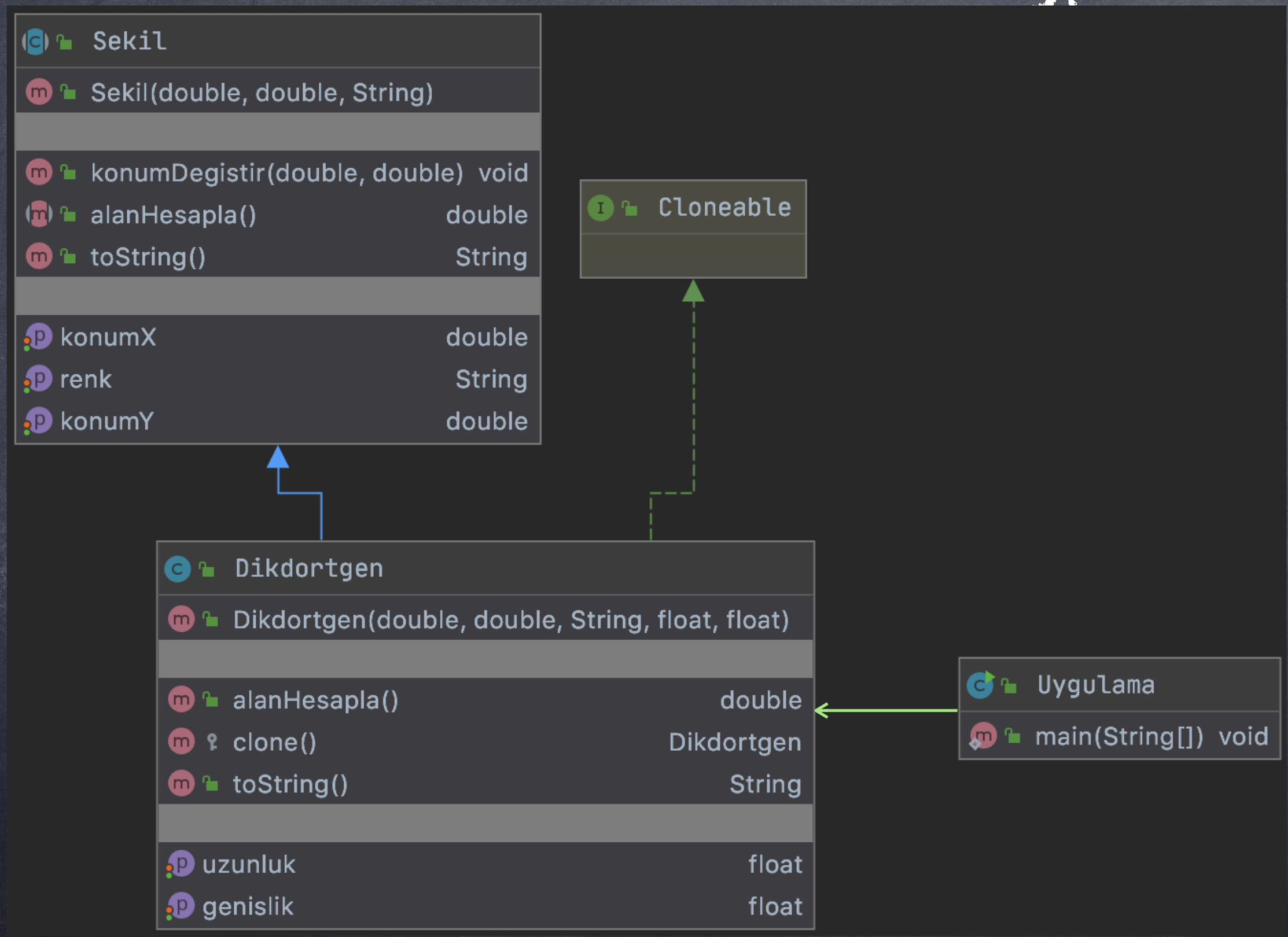

https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders12

# Tasarım Desenleri : Prototype

* Nesne oluşturmayla (creational) ilgili desenlerden biridir.
* Yeni nesne oluşturma işleminin maliyetli olduğu durumlarda, mevcut nesnenin kopyasını oluşturmak gerektiğinde kullanılabilir.
* Karmaşık veritabanı sorguları sonucu oluşan nesneler "cache" içerisinde saklanır ve bu nesnelere ihtiyaç duyulduğunda önce "cache" içerisinde aranır. Bulunursa kopyası oluşturulur ve veritabanı sorgusuna gerek kalmaz.
* Nesneler, prototype (cloneable) arayüzü içerisinde tanımlı clone() yöntemini gerçekleyerek, (new komutu ile) kendi kopyalarını oluştururlar.

https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders12

# Tasarım Desenleri : Prototype




https://github.com/celalceken/NesneYonelimliAnalizVeTasarimDersiUygulamalari/tree/master/Ders12