Yazılım Mühendisliği

Ders 5

Sistem Modelleme

Kaynak: https://www.tutorialspoint.com/uml/index.htm
https://sparxsystems.com/resources/tutorials/uml2/index.html

Sistem Modelleme

Sistem modelleme, her modelin o sistemin farklı bir görünümünü veya perspektifini sunduğu bir sistemin **soyut modellerini geliştirme** sürecidir.

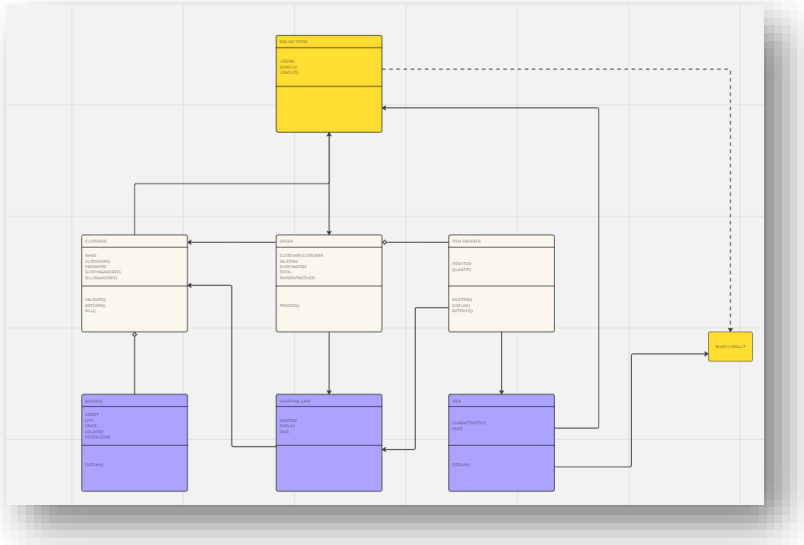
Sistem modelleme, şimdi neredeyse her zaman Birleşik Modelleme Dilindeki (UML)* gösterimlere dayanan bir tür grafiksel gösterim kullanan bir sistemi temsil etmek anlamına gelmiştir.

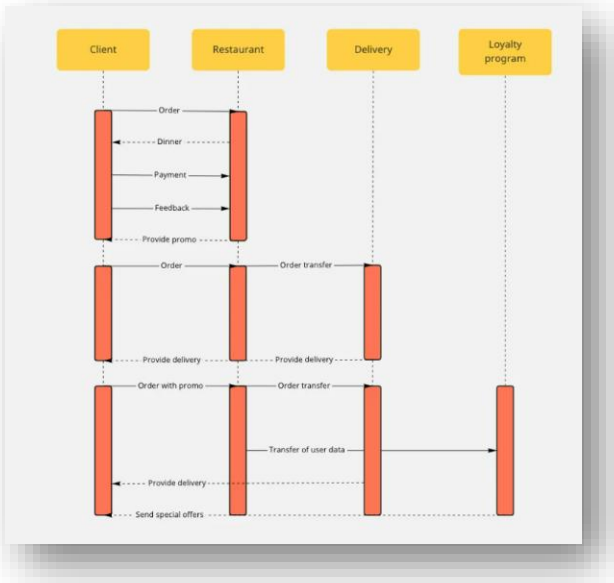
Sistem modelleme, **analistin sistemin işlevselliğini** anlamasına yardımcı olur ve **modeller müşterilerle iletişim kurmak** için kullanılır.

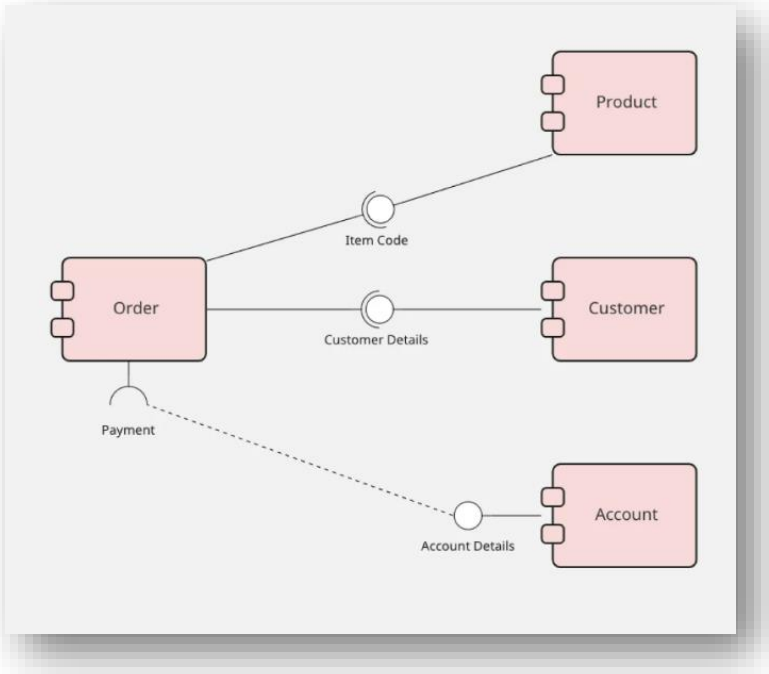
*UML (Unified Modeling Language)

UML

- UML, yazılım sistemlerinin eserlerini belirtmek, görselleştirmek, oluşturmak ve belgelemek için standart bir dildir.
- UML, Nesne Yönetim Grubu (OMG) tarafından oluşturulmuş ve Ocak 1997'de OMG'ye UML 1.0 belirtim taslağı önerilmiştir. OMG-UML
- OMG, gerçek bir endüstri standardı oluşturmak için sürekli olarak çaba göstermektedir.

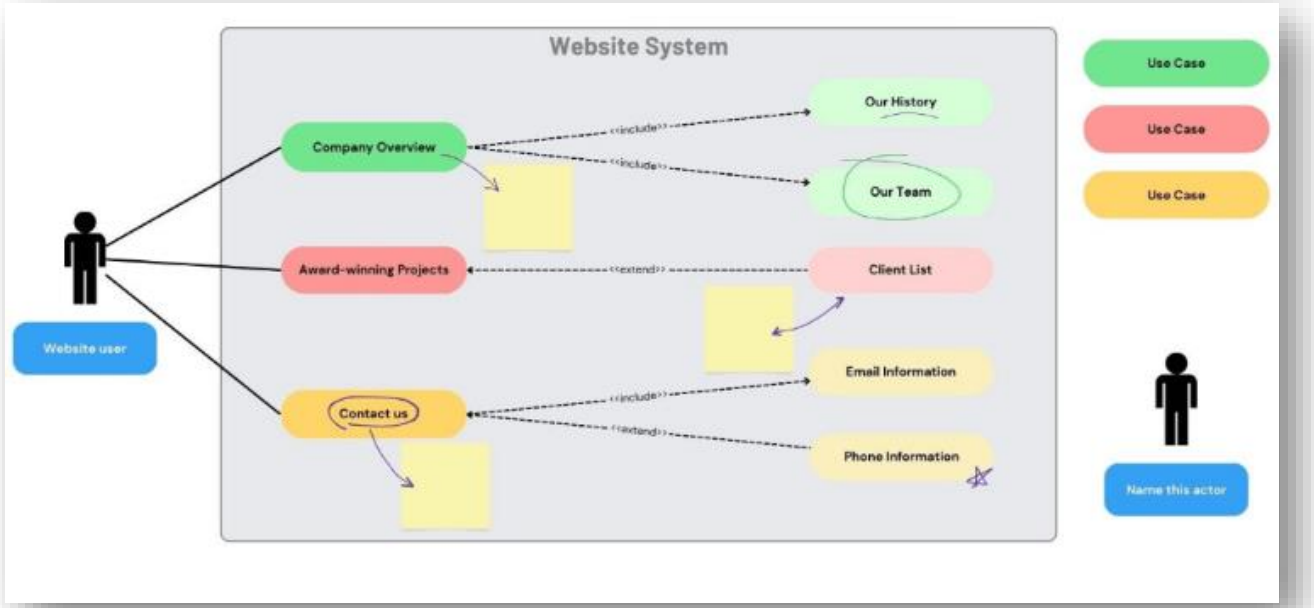






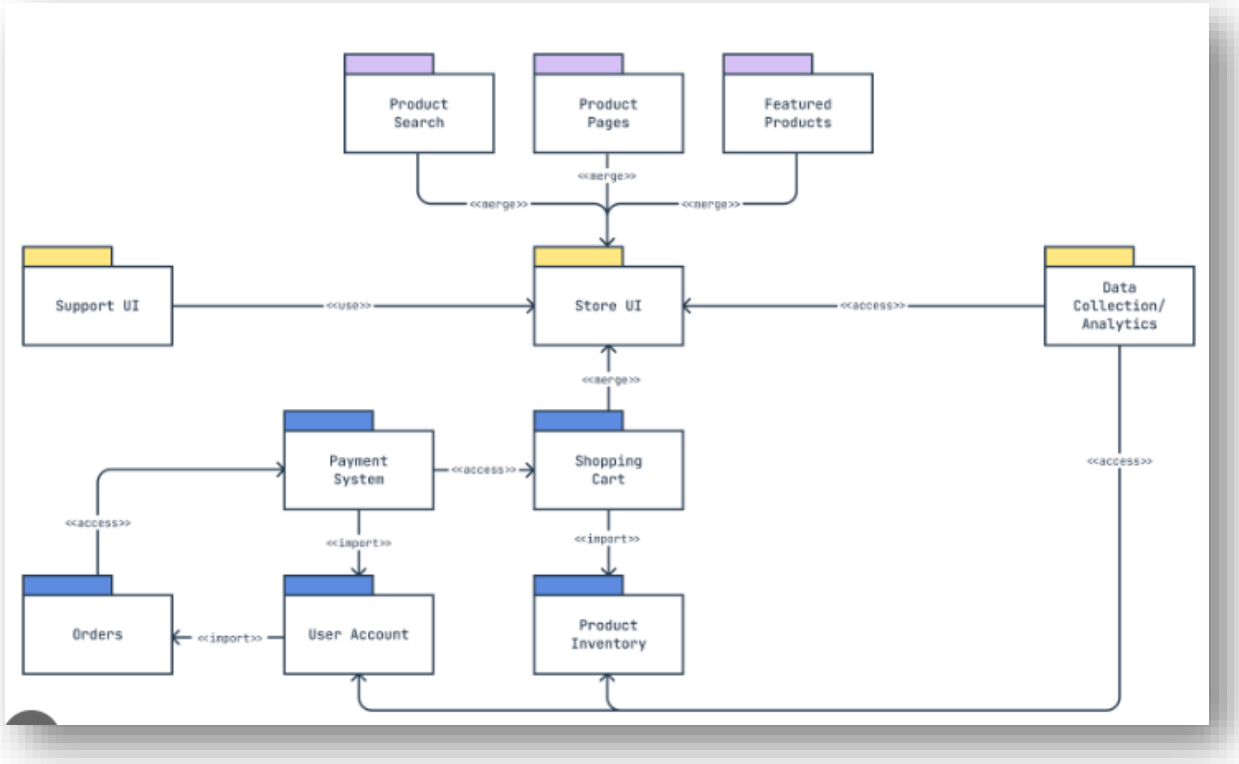
UML

- UML, Birleşik Modelleme Dili anlamına gelir .
- UML, C++, Java, COBOL gibi diğer yaygın programlama dillerinden farklıdır.
- UML, yazılım planları yapmak için kullanılan resimsel bir dildir.
- UML, yazılım sistemini görselleştirmek, belirlemek, oluşturmak ve belgelemek için genel amaçlı bir görsel modelleme dili olarak tanımlanabilir.
- UML genellikle yazılım sistemlerini modellemek için kullanılsa da bu sınırla sınırlı değildir. Ayrıca yazılım dışı sistemleri modellemek için de kullanılır. Örneğin, bir üretim birimindeki süreç akışı vb.



UML

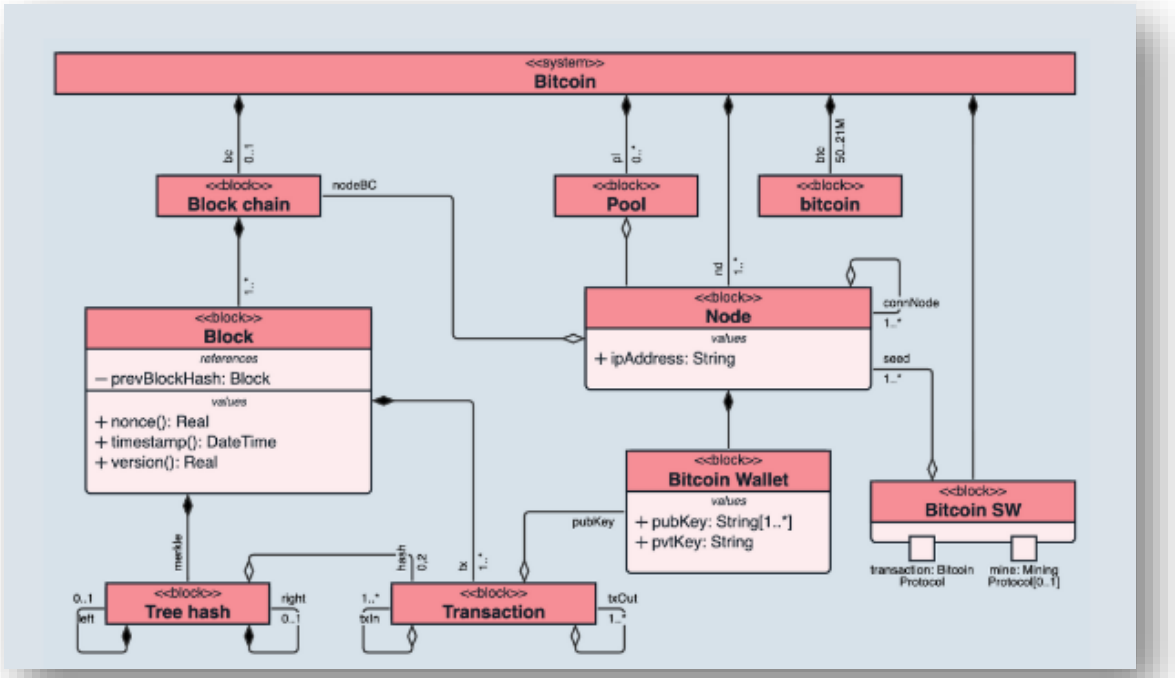
- UML bir programlama dili değildir, ancak UML diyagramlarını kullanarak çeşitli dillerde kod oluşturmak için araçlar kullanılabilir.
- UML'nin nesne yönelimli analiz ve tasarımla doğrudan bir ilişkisi vardır.
- Bazı standardizasyonlardan sonra UML, bir OMG standardı haline gelmiştir.



UML

UML'nin Hedefleri

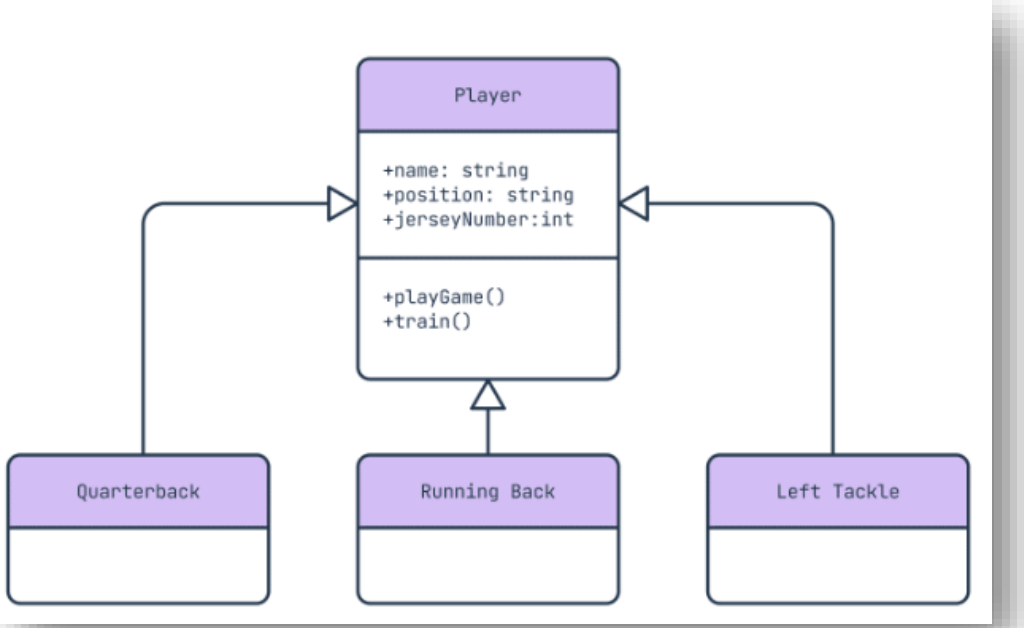
- Bir resim bin kelimeye bedeldir , bu deyim kesinlikle UML'yi tanımlamaya uyuyor.
- Nesne yönelimli kavramlar UML'den çok daha önce tanıtıldı. O zamanlar, nesne yönelimli geliştirmeyi organize etmek ve pekiştirmek için standart metodolojiler yoktu.
- UML geliştirmenin bir dizi hedefi vardır, ancak en önemlisi, **tüm modelleyicilerin kullanabileceği bazı genel amaçlı modelleme dili tanımlamaktır** ve ayrıca anlaşılması ve kullanılması **basit** hale getirilmelidir.



UML

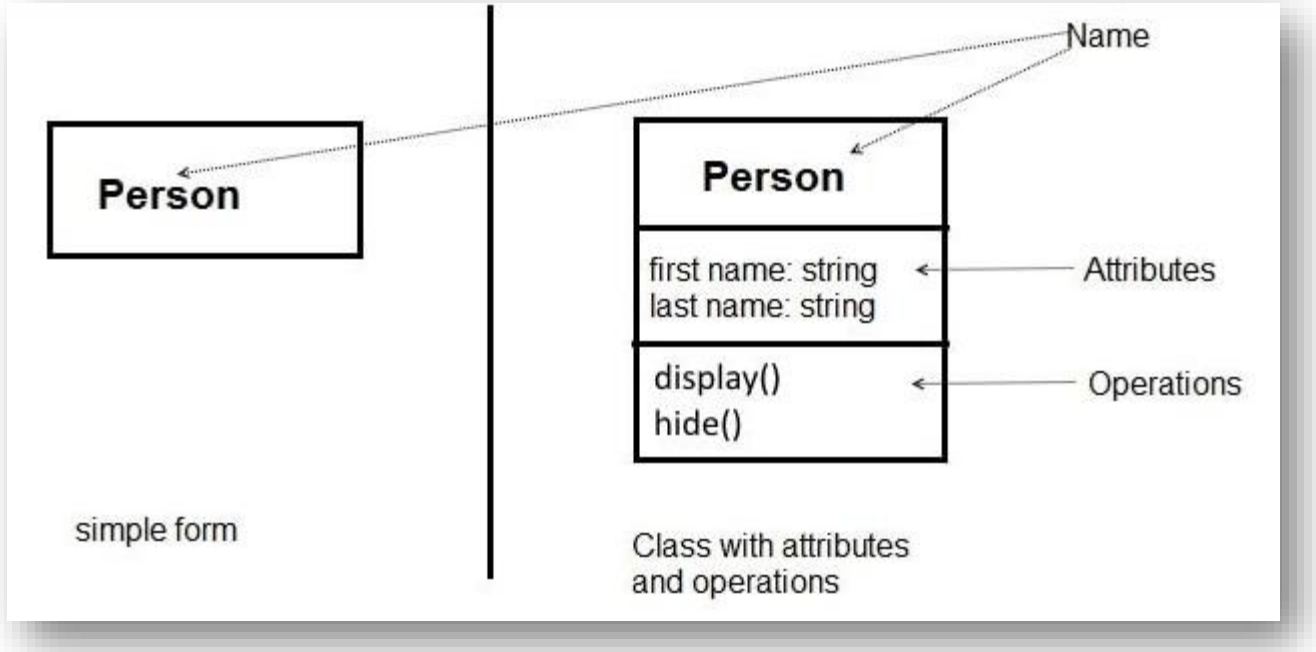
UML'nin Hedefleri

- UML diyagramları yalnızca geliştiriciler için değil, aynı zamanda **iş kullanıcıları, sıradan insanlar ve sistemi anlamak isteyen herkes** için yapılmıştır.
- Sistem bir yazılım veya yazılım dışı sistem olabilir. Bu nedenle, UML'nin bir geliştirme yöntemi olmadığı, onu başarılı bir sistem haline getirmek için süreçlere eşlik ettiği açık olmalıdır.
- Sonuç olarak, UML'nin amacı, günümüzün karmaşık ortamında olası tüm pratik sistemleri modellemek için **basit bir modelleme mekanizması** olarak tanımlanabilir.



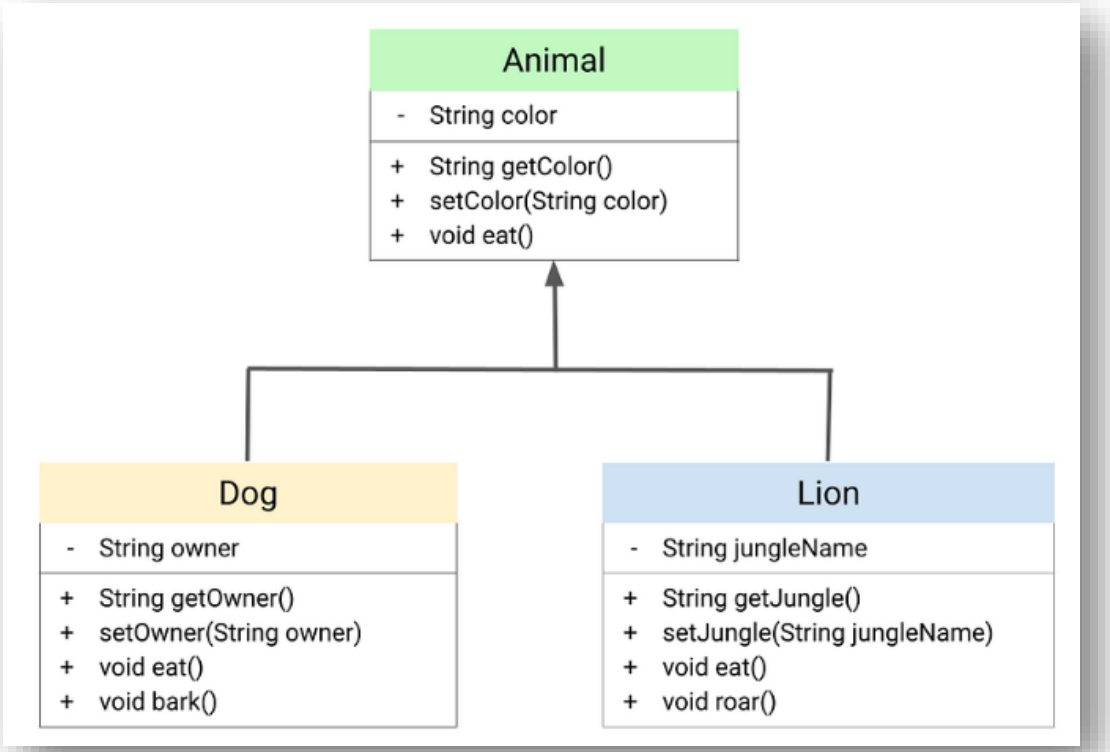
Nesne Yönelimli Kavramlar

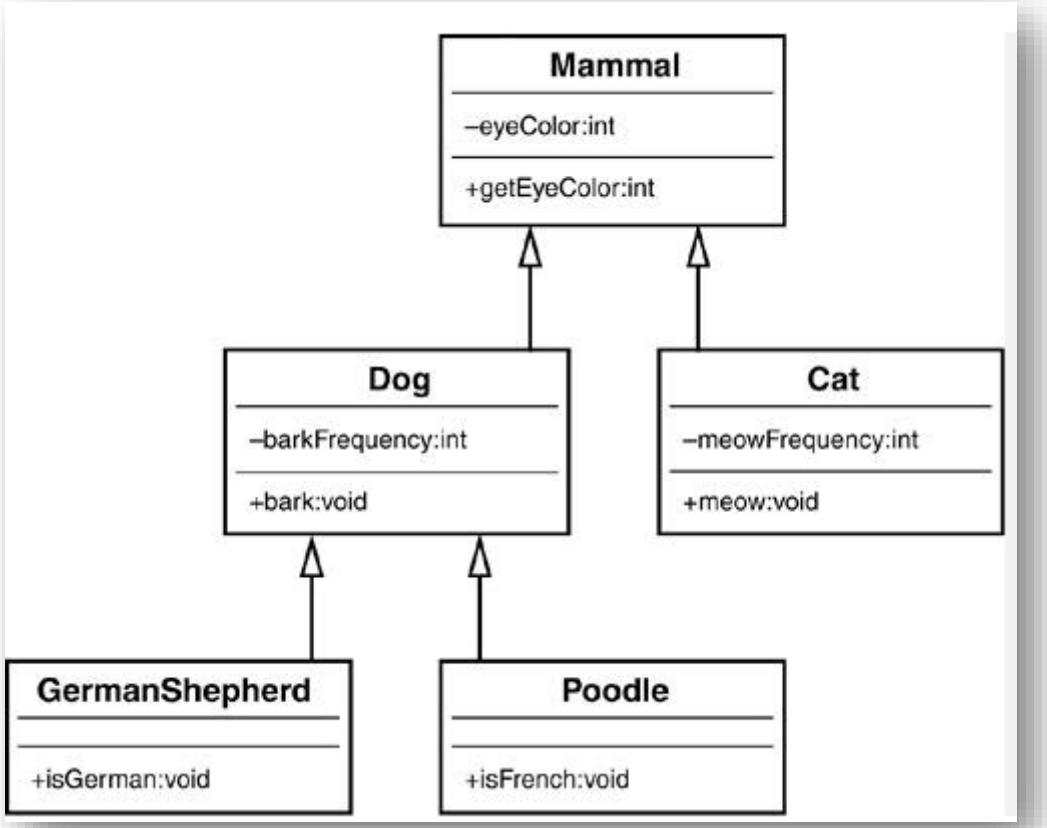
- UML, nesne yönelimli (OO) analiz ve tasarımın halefi olarak tanımlanabilir.
- Bir nesne hem verileri hem de verileri kontrol eden yöntemleri içerir.
- Veriler, nesnenin durumunu temsil eder.
- Bir sınıf, bir nesneyi tanımlar ve ayrıca gerçek dünya sistemini modellemek için bir hiyerarşi oluşturur.



Nesne Yönelimli Kavramlar

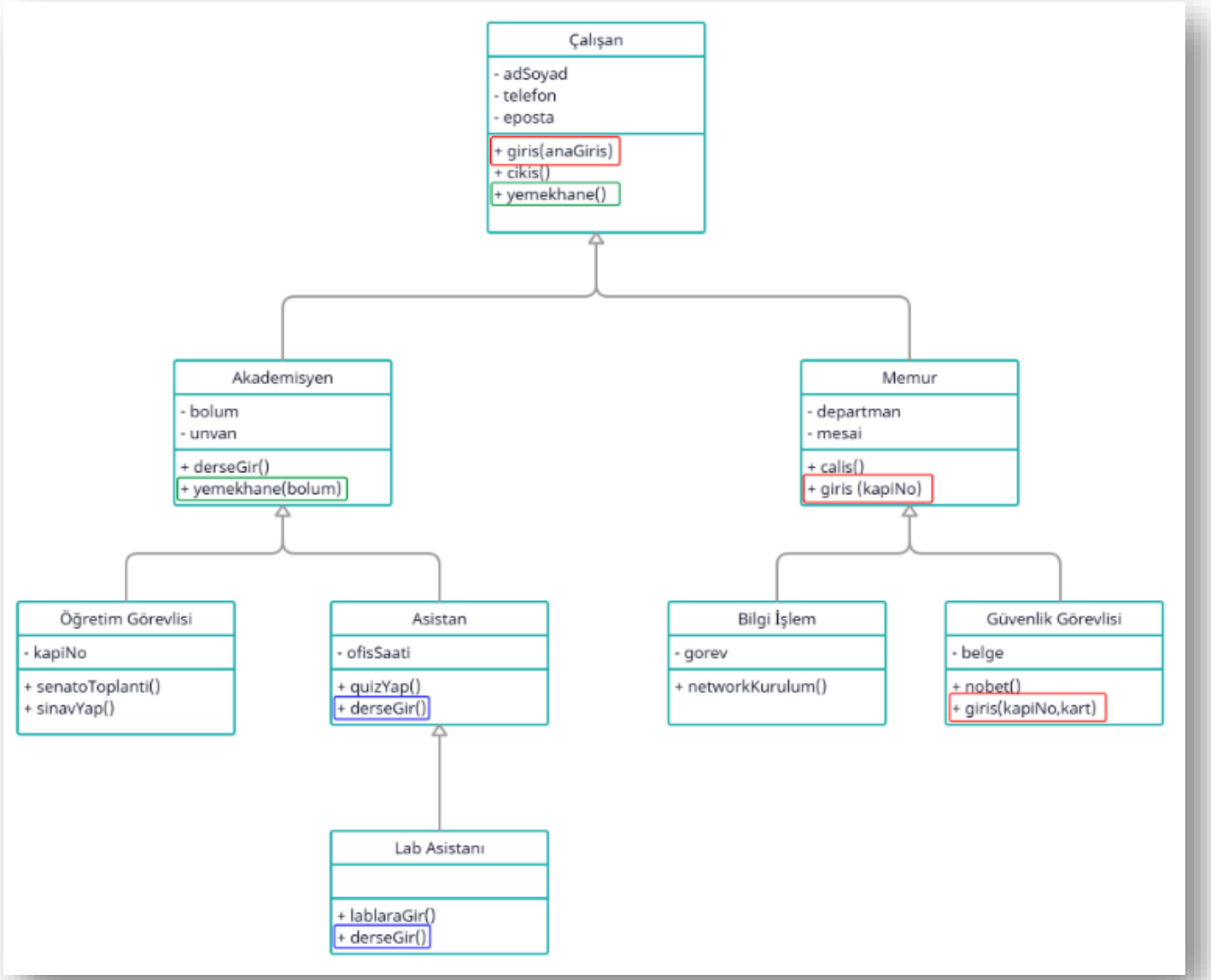
- Hiyerarşi, kalıtım olarak temsil edilir ve sınıflar, gereksinime göre farklı şekillerde de ilişkilendirilebilir.
- Nesneler, etrafımızda var olan gerçek dünya varlıklarıdır ve soyutlama, kapsülleme, kalıtım ve polimorfizm gibi temel kavramların tümü UML kullanılarak temsil edilebilir.





Nesne Yönelimli Kavramlar

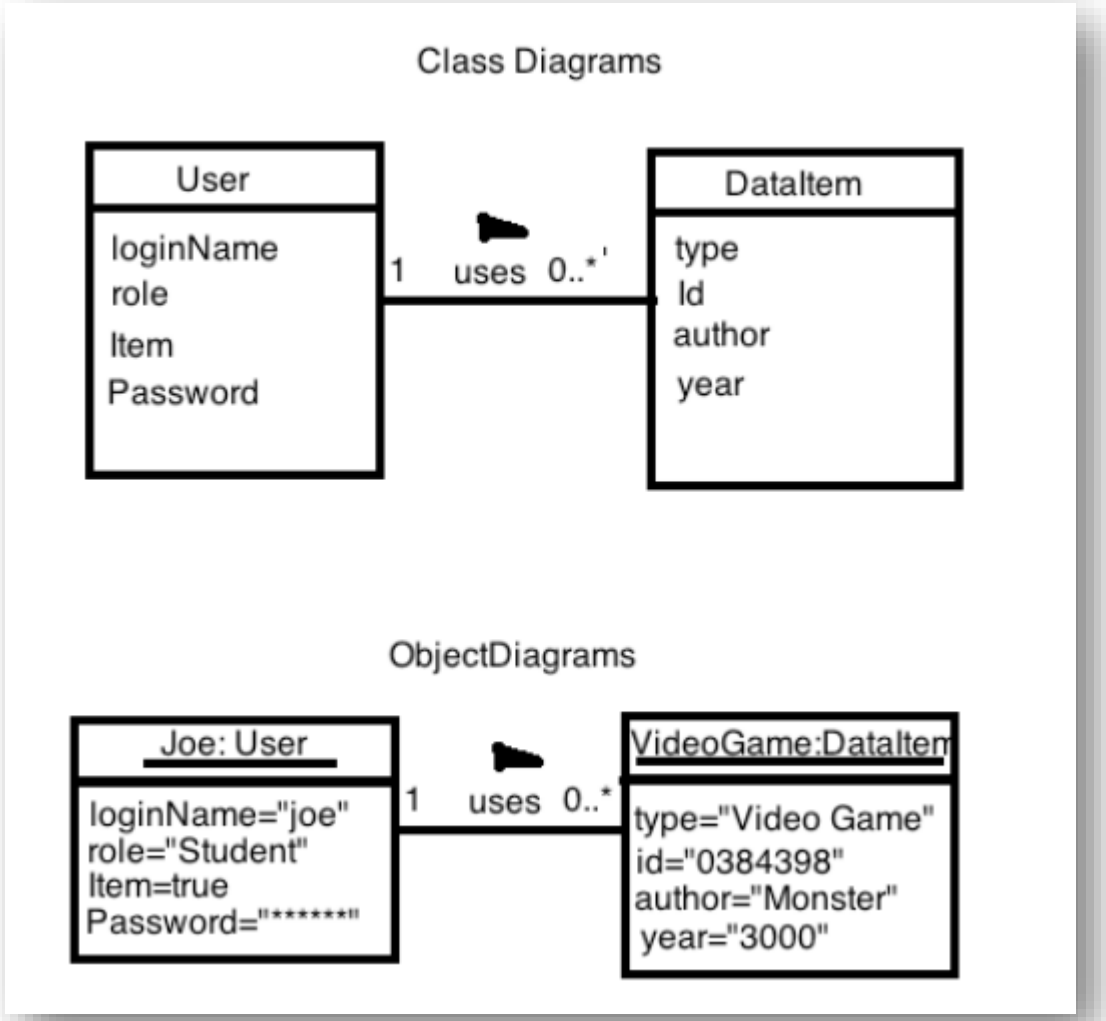
- UML, nesne yönelimli analiz ve tasarımda var olan tüm kavramları temsil edecek kadar güçlüdür.
- UML diyagramları yalnızca nesne yönelimli kavramların temsilidir.
- Bu nedenle, UML'yi öğrenmeden önce, OO kavramını ayrıntılı olarak anlamak önemli hale gelir.



Nesne Yönelimli Kavramlar

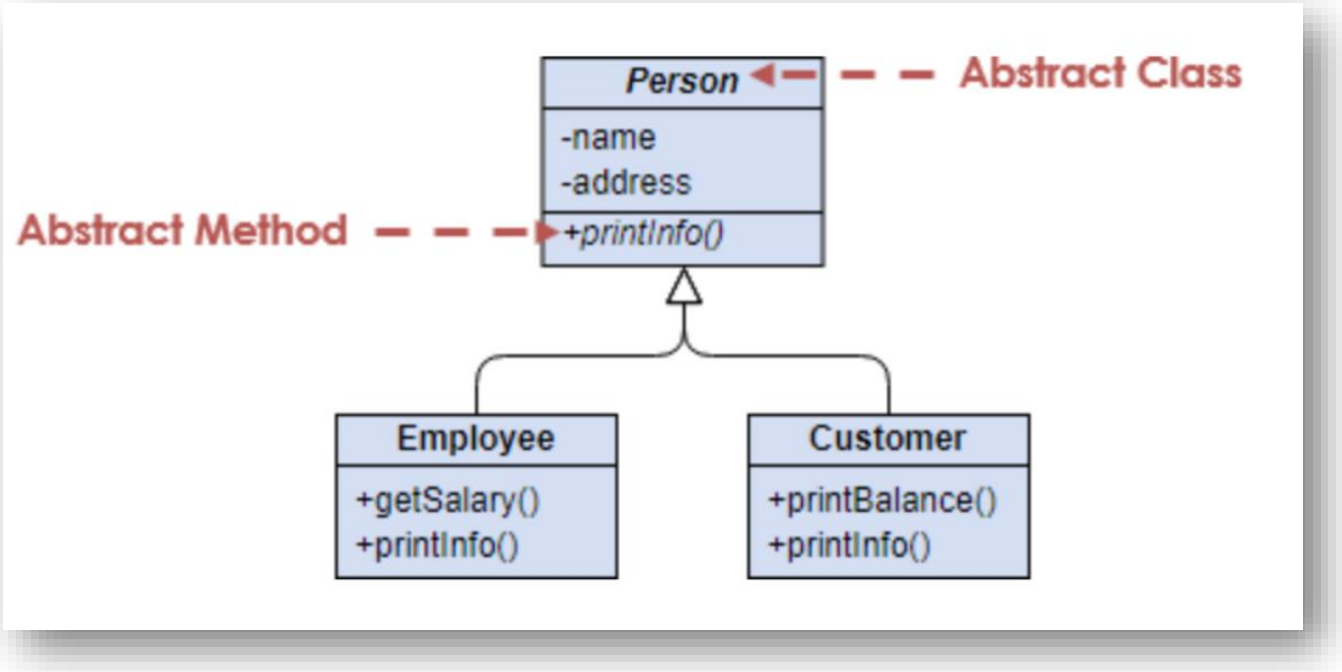
- ***Nesneler (Objects):*** Nesneler, bir varlığı veya temel yapı taşını temsil eder. Gerçek dünyadaki varlıkları (araba, insan, kitap vb.) veya soyut kavramları (hesap, müşteri, işlem vb.) temsil edebilirler. Nesnelerin özellikleri (alanlar) ve davranışları (metodlar) olabilir.

- ***Sınıf (Class):*** Bir sınıf, nesnelerin "taslağını" veya "şablonunu" tanımlayan bir yapıdır. Nesneler, bir sınıfın örneğidir. Örneğin, "Araba" bir sınıftır ve bu sınıfın örnekleri farklı arabalardır. Sınıflar, nesnelerin ortak özelliklerini (alanlar) ve davranışlarını (metodlarını) tanımlar.



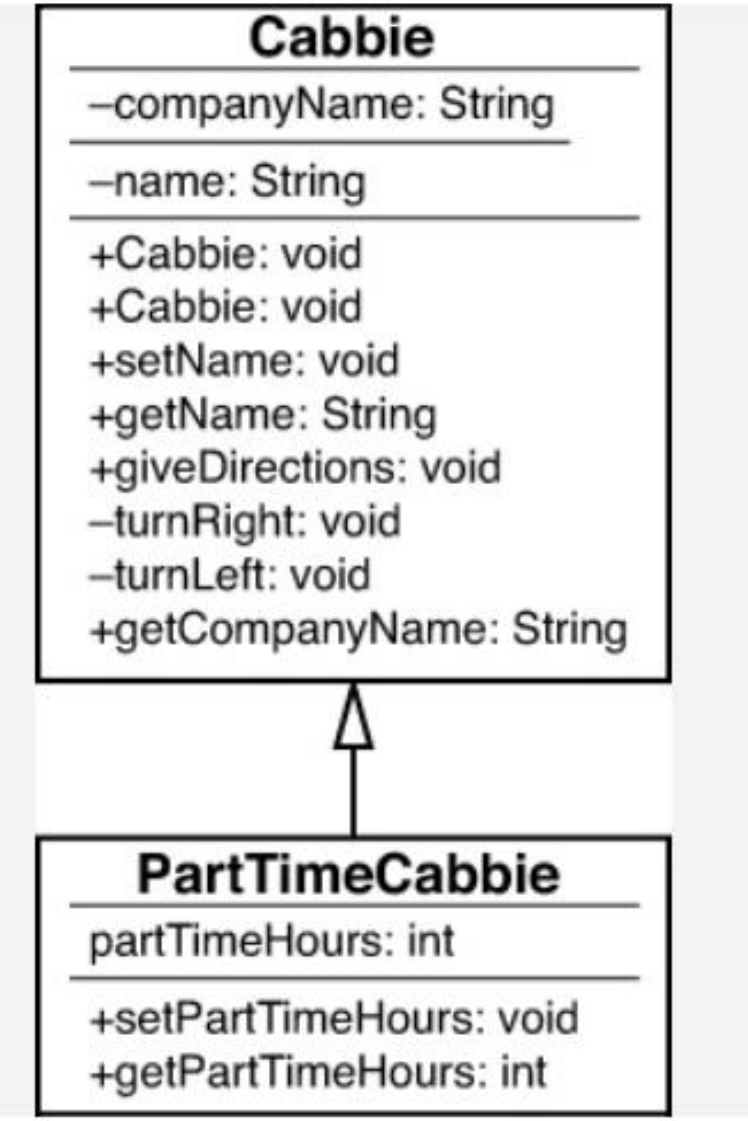
Nesne Yönelimli Kavramlar

- ***Soyutlama (Abstraction):*** Soyutlama, gerçek dünyadaki varlıkların veya süreçlerin önemli özelliklerini vurgulayarak karmaşıklığı azaltma sürecidir. Yani, sınıflar ve nesneler, gerçek dünyadaki varlıkların veya kavramların özelliklerini ve davranışlarını temsil ederken, gereksiz detaylardan kaçınarak soyutlama yaparlar.



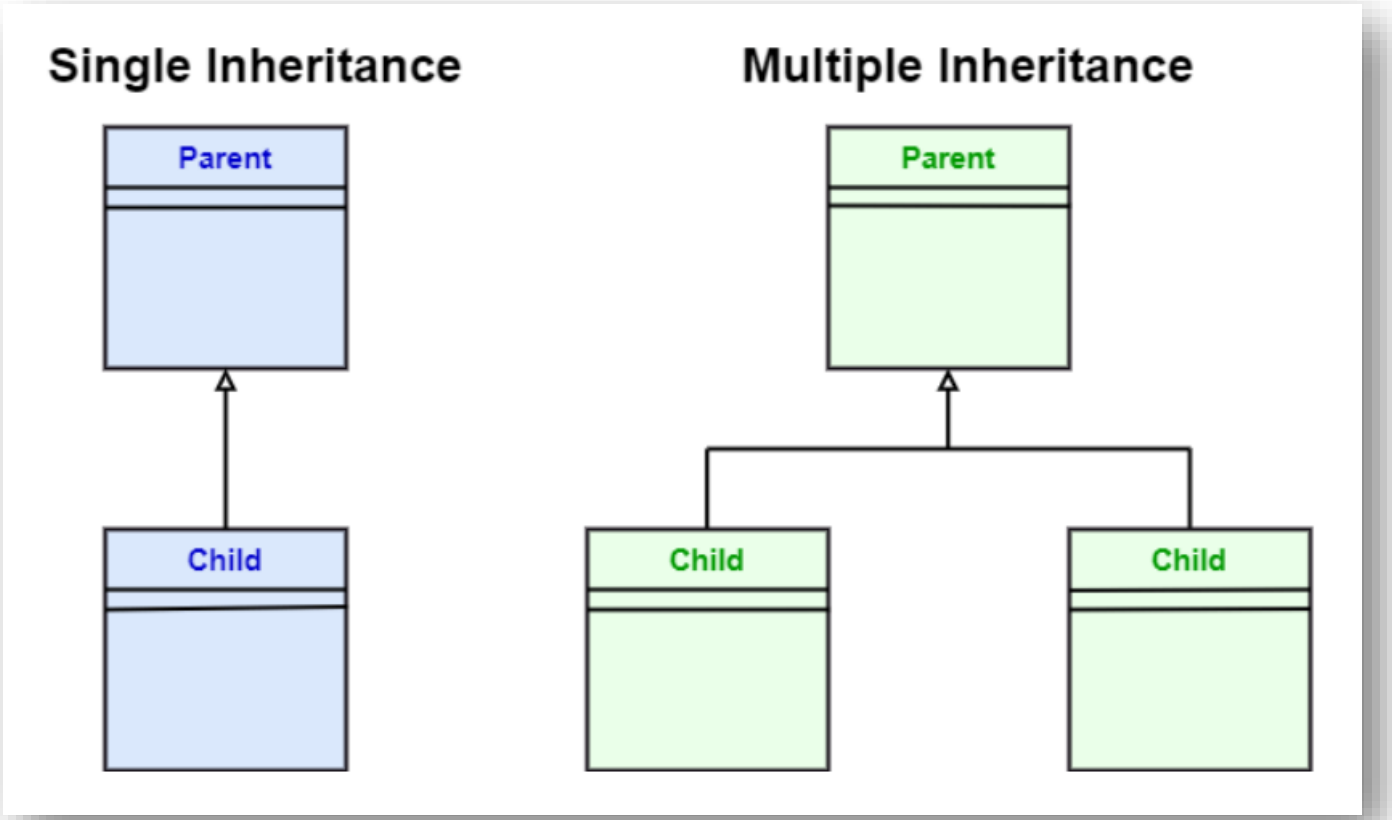
Nesne Yönelimli Kavramlar

- ***Kapsülleme (Encapsulation***): Kapsülleme, verileri (alanlar) ve onların işlevlerini (metodlar) bir araya getirme ve dış dünyadan gizleme sürecidir. Yani, bir nesnenin iç yapısını koruyarak, dışarıdan erişimi kontrol etme yeteneğidir. Bu, verilerin doğrudan değiştirilmesini önleyerek programın daha güvenli ve düzenli olmasını sağlar.



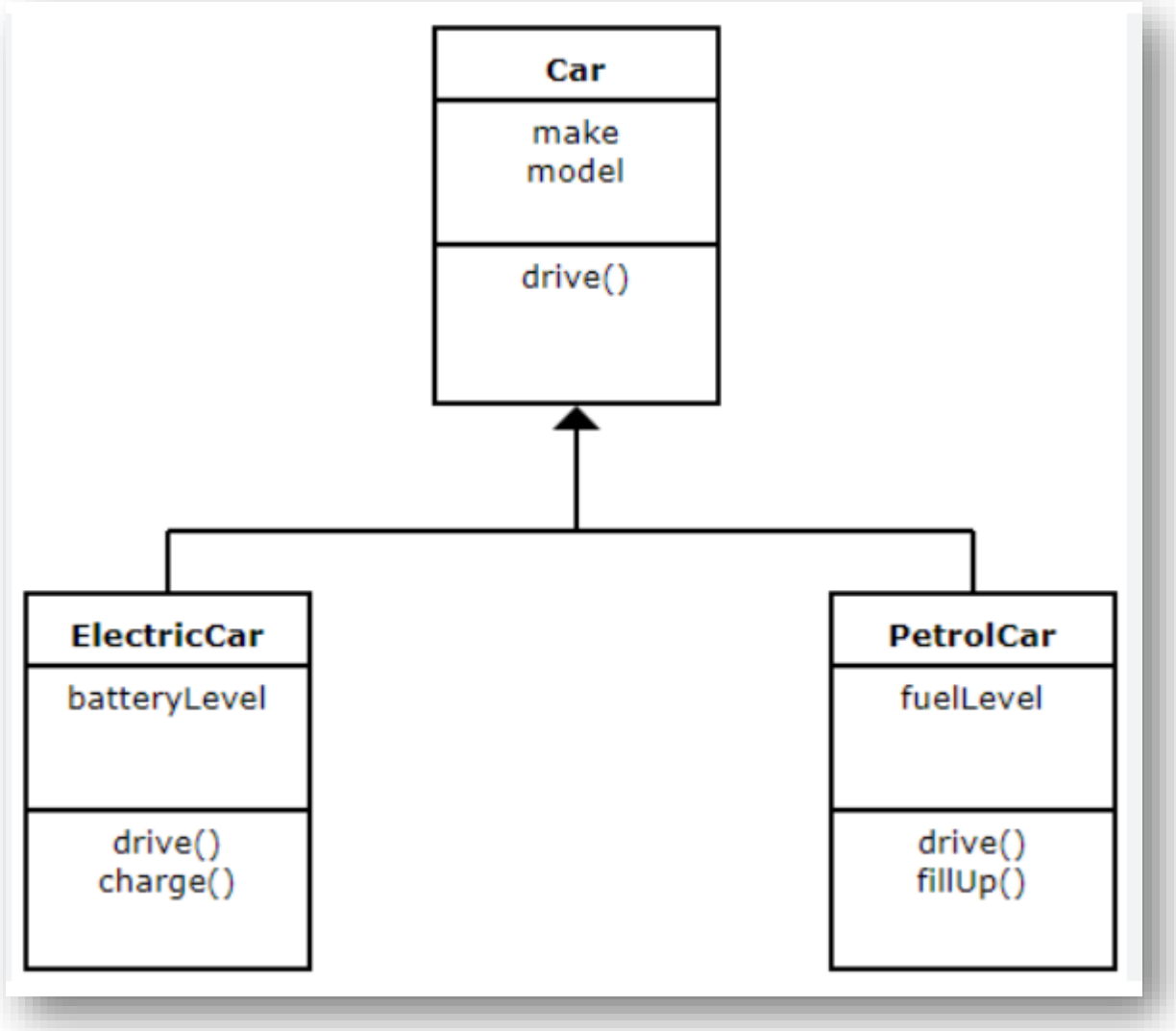
Nesne Yönelimli Kavramlar

- ***Kalıtım (Inheritance):*** Kalıtım, mevcut sınıflardan yeni sınıflar oluşturma mekanizmasıdır. Bir sınıf, başka bir sınıftan türetilerek (miras alınarak) yeni özellikler ve davranışlar ekleyebilir veya mevcut olanları değiştirebilir. Bu, kodun yeniden kullanılabilirliğini artırır ve kodun düzenliğini sağlar.



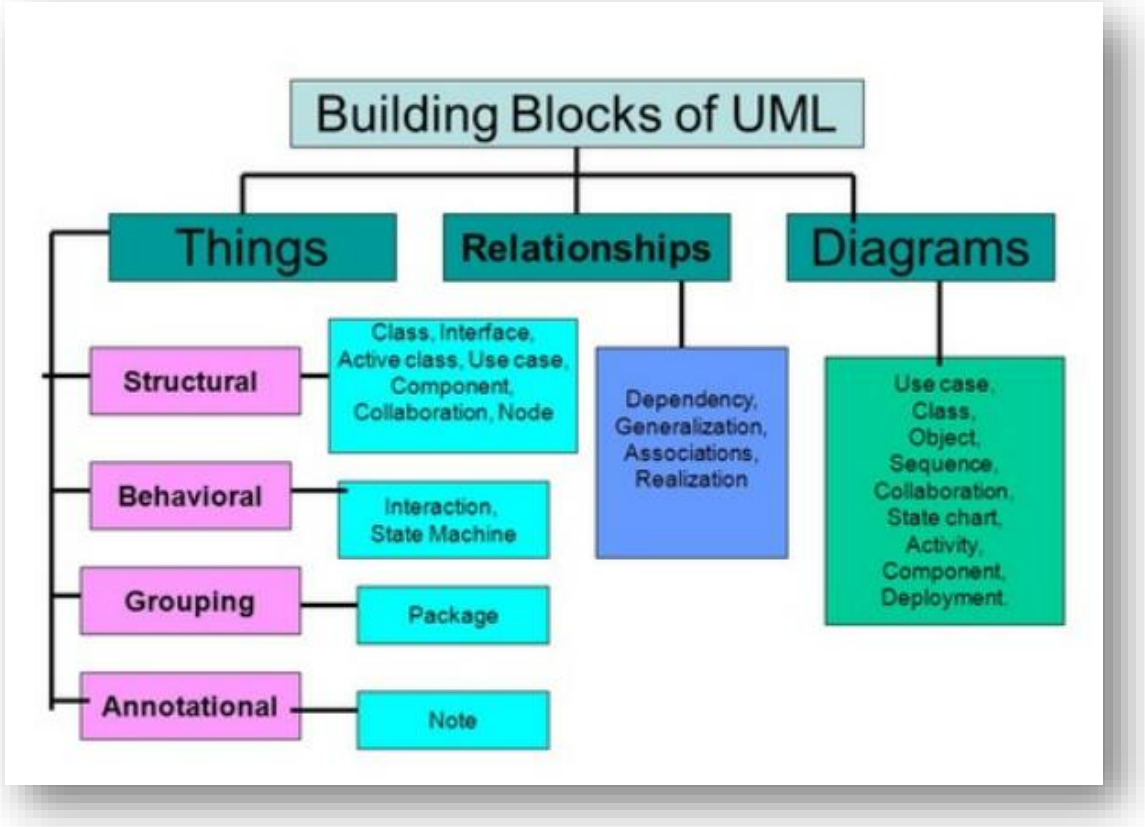
Nesne Yönelimli Kavramlar

- ***Polimorfizm (Polymorphism):*** Polimorfizm, farklı şekillerde var olma mekanizmasını tanımlar. Bu, aynı metodun farklı sınıflar tarafından farklı şekillerde uygulanabilmesini ifade eder. Yani, aynı isimdeki metot farklı davranışlar gösterebilir. Bu, kodun daha esnek ve modüler olmasını sağlar ve genellikle kalıtım ve soyutlama ile birlikte kullanılır.



UML'nin Yapıtaşları

1. Nesneler
2. İlişkiler
3. Diyagramlar

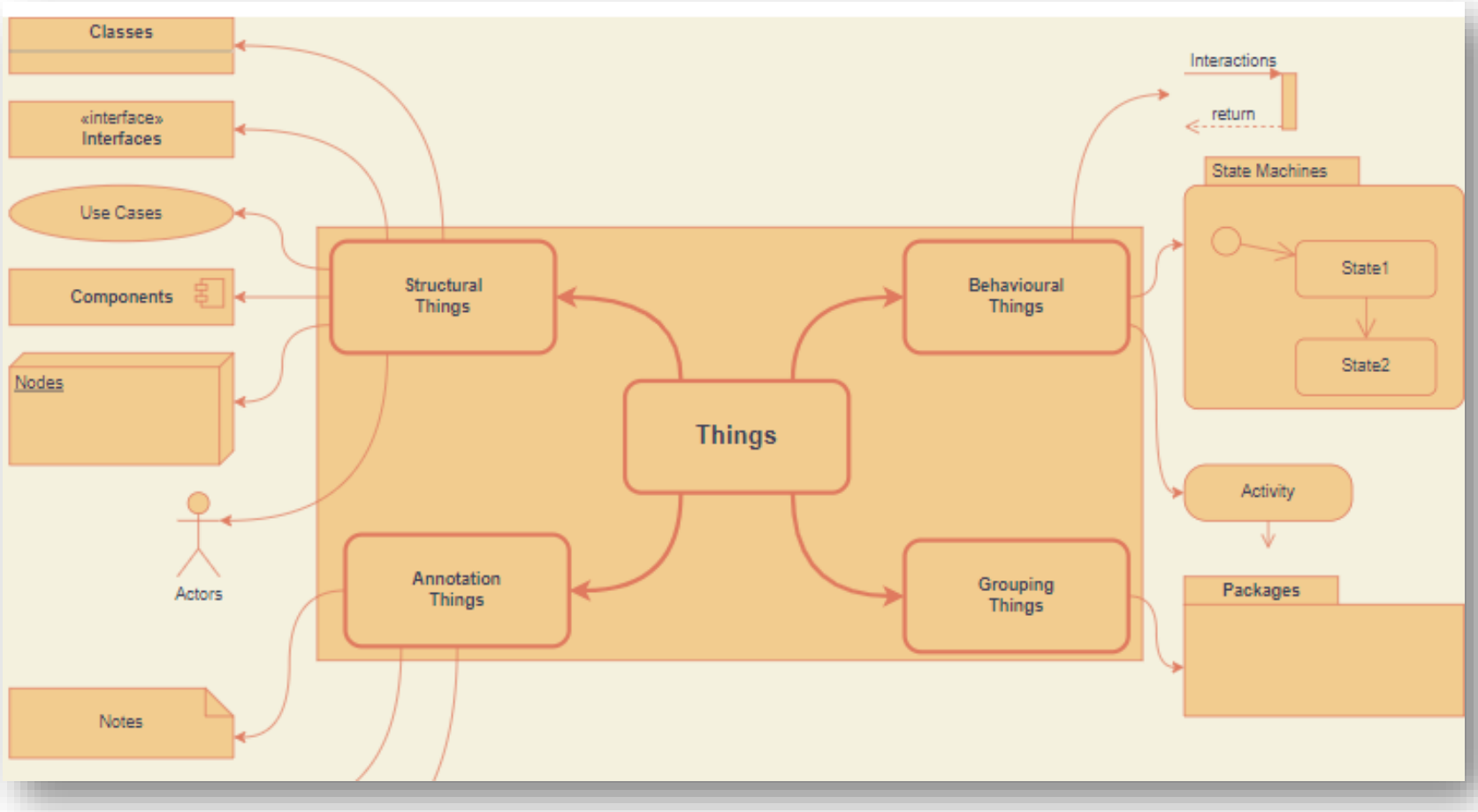


UML’nin Yapıtaşları

1- Nesneler

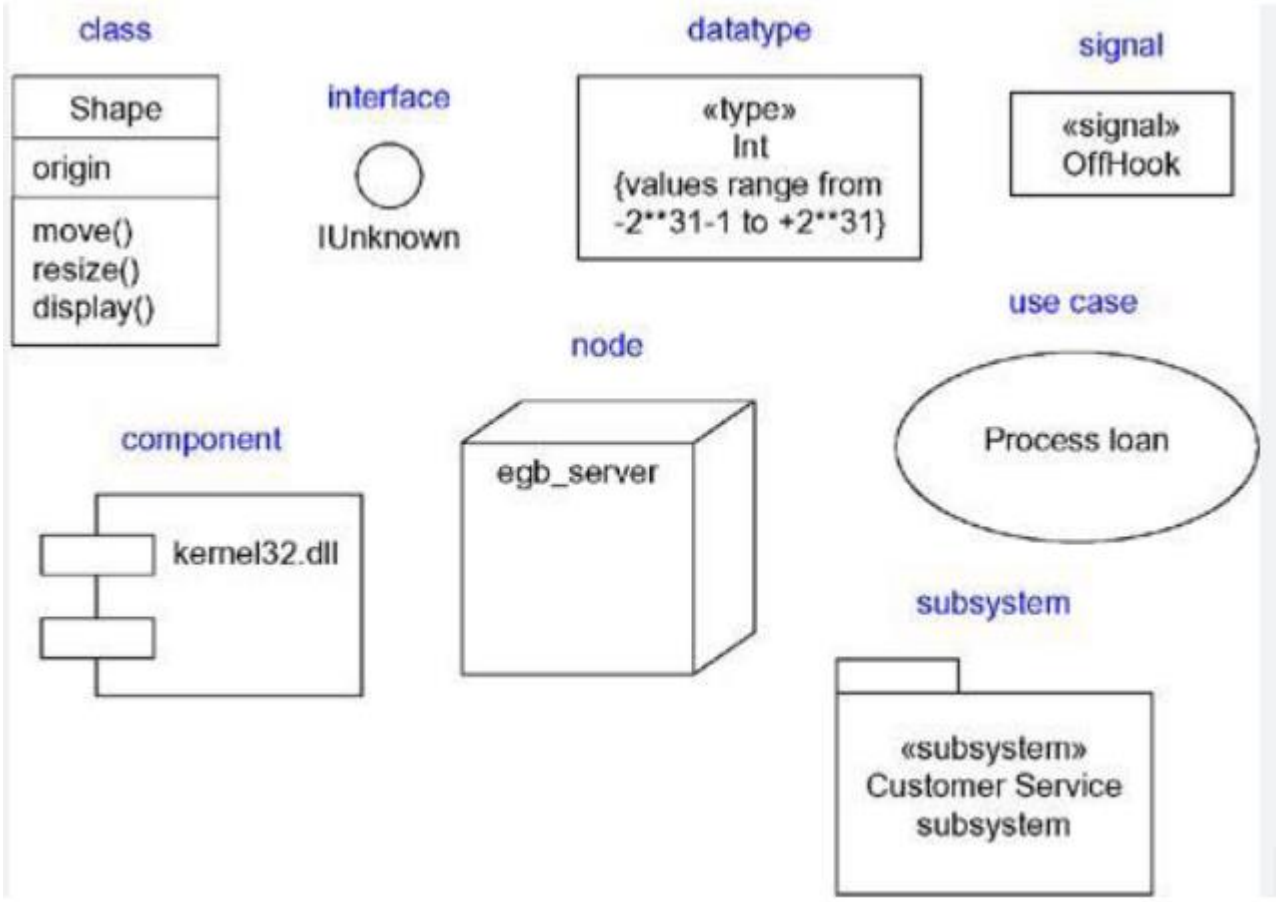
Nesneler , UML'nin en önemli yapı taşlarıdır. Bunlar;

- Yapısal
- Davranışsal
- Gruplamalı
- Açıklamalı olabilir



Yapısal (Structural Things)

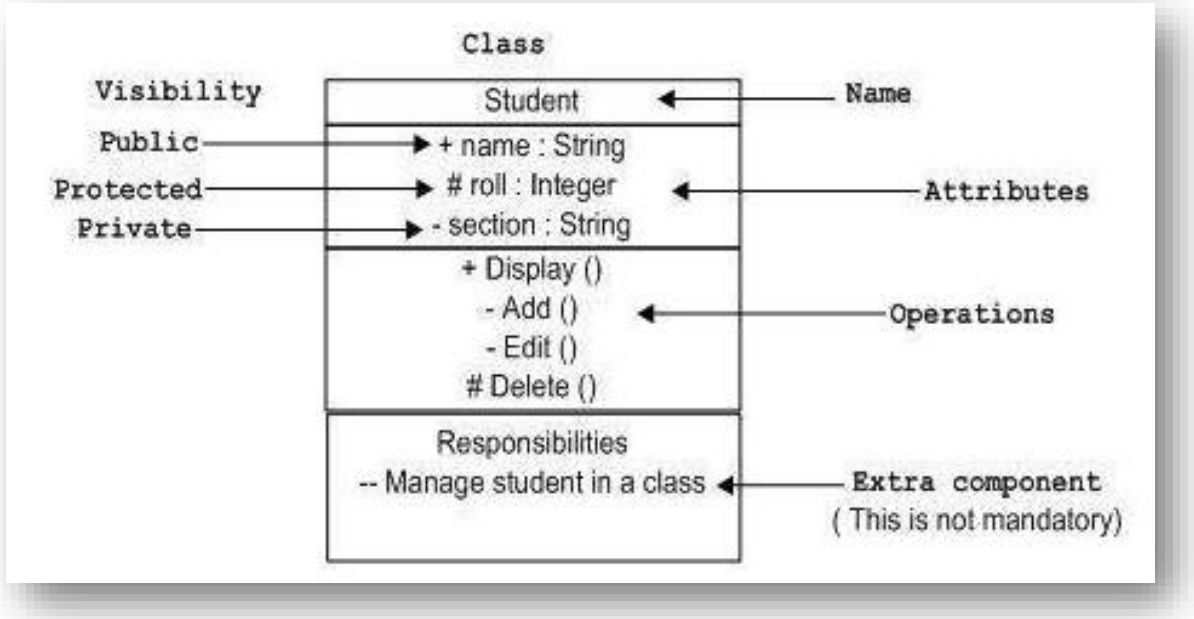
Yapısal şeyler, modelin statik kısmını tanımlar. Fiziksel ve kavramsal öğeleri temsil ederler.

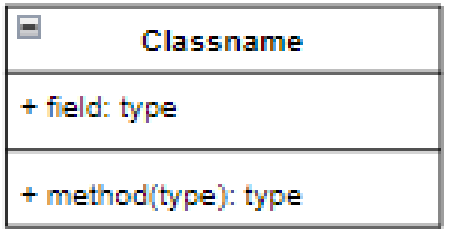


Yapısal (Structural Things)

Sınıf - Sınıflar nesneleri temsil etmek için kullanılır. Nesneler, özellikleri ve sorumluluğu olan herhangi bir şey olabilir. UML sınıfı aşağıdaki şekil ile temsil edilmektedir. Diyagram dört bölüme ayrılmıştır.

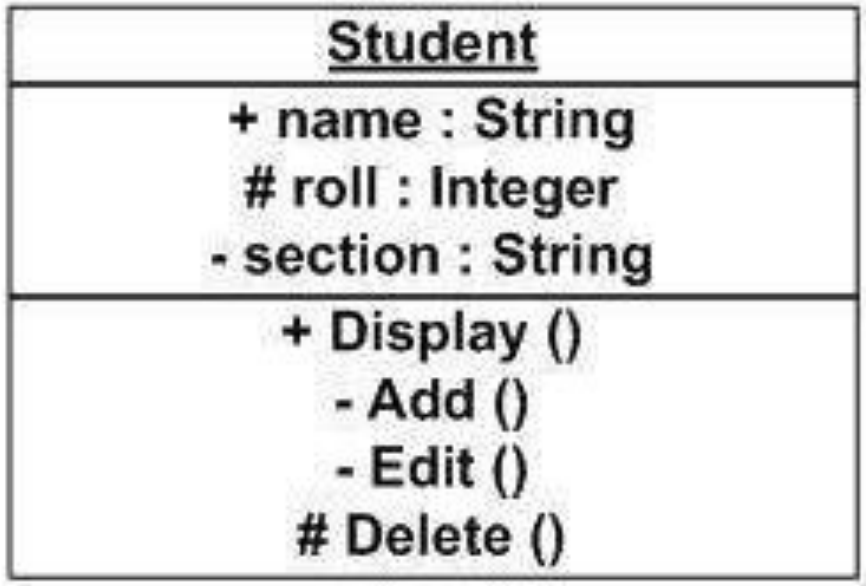
1. Üst kısım sınıfı adlandırmak için kullanılır.
2. İkincisi, sınıfın niteliklerini göstermek için kullanılır.
3. Üçüncü bölüm, sınıf tarafından gerçekleştirilen işlemleri açıklamak için kullanılır.
4. Dördüncü bölüm, herhangi bir ek bileşeni göstermek için isteğe bağlıdır.





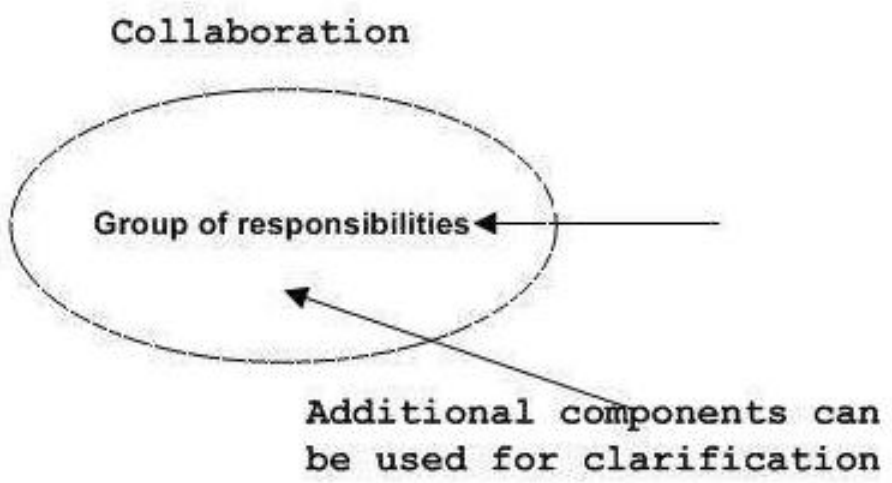
Yapısal (Structural Things)

Nesne -Nesne , sınıfla aynı şekilde temsil edilir. Tek fark, aşağıdaki şekilde gösterildiği gibi altı çizili olan *addır* .



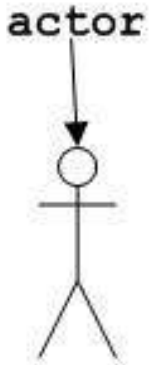
Yapısal (Structural Things)

İşbirliği (Collaboration) -İşbirliği, aşağıdaki şekilde gösterildiği gibi noktalı bir tutulma ile temsil edilir. Tutulmanın içine yazılmış bir adı vardır.



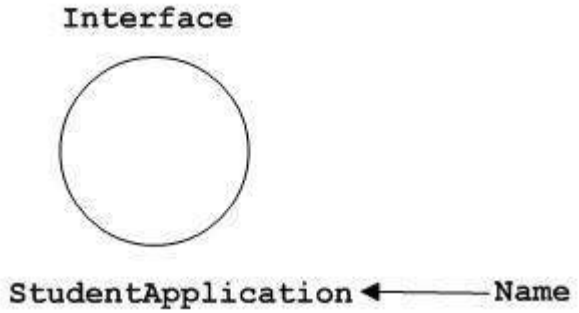
Yapısal (Structural Things)

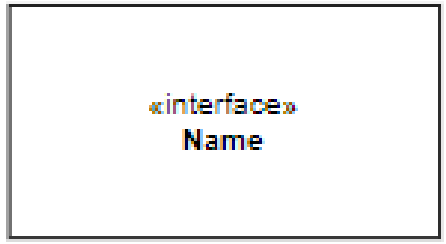
Aktör -Bir aktör, sistemle etkileşime giren bazı dahili veya harici varlıklar olarak tanımlanabilir

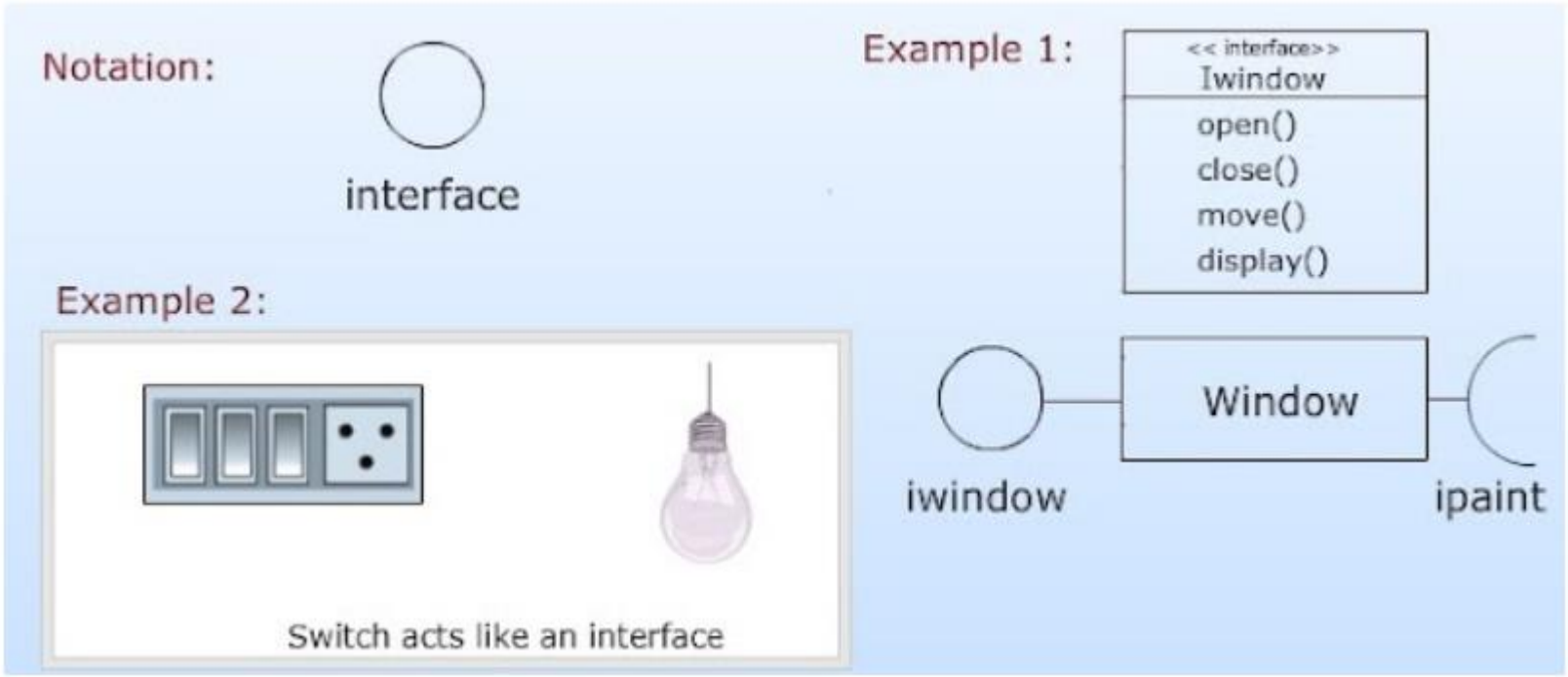


Yapısal (Structural Things)

Arayüz - Arayüz, bir sınıfın sorumluluğunu belirleyen bir dizi işlemi tanımlar. Arayüz, aşağıdaki şekilde gösterildiği gibi bir daire ile temsil edilir. Genellikle dairenin altına yazılan bir adı vardır.

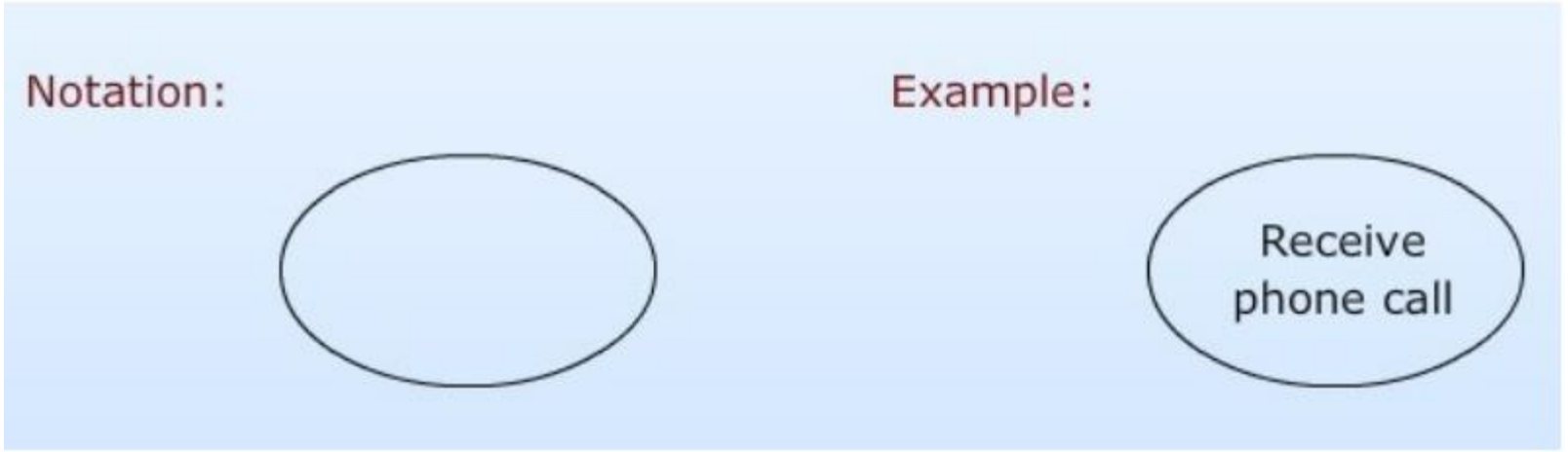


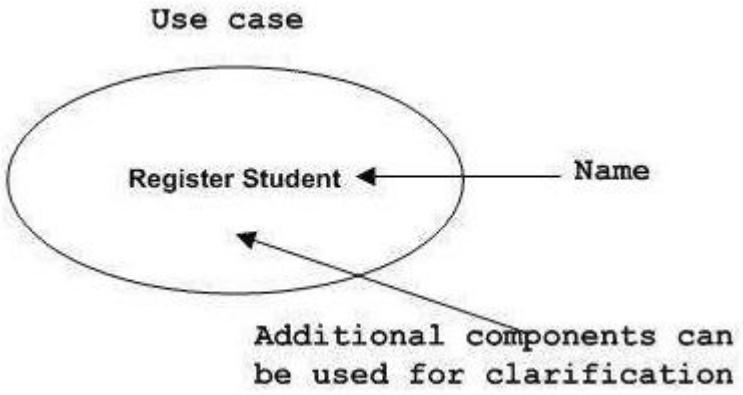




Yapısal (Structural Things)

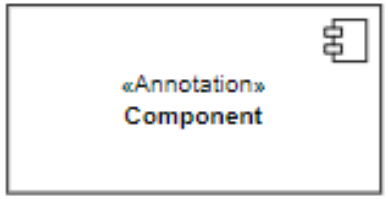
Kullanım durumu - Kullanım durumu, belirli bir hedef için bir sistem tarafından gerçekleştirilen bir dizi eylemi temsil eder.

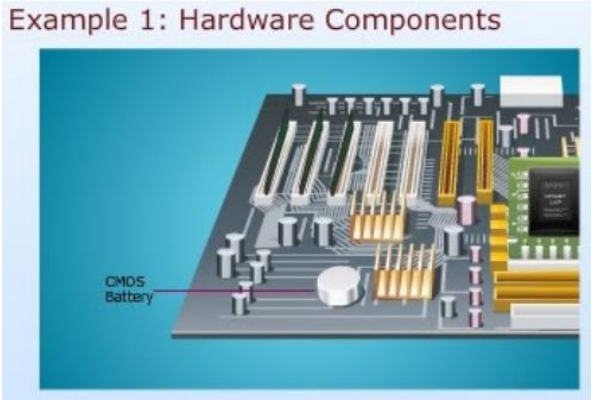




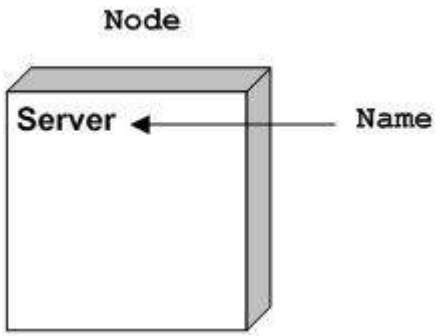
Yapısal (Structural Things)

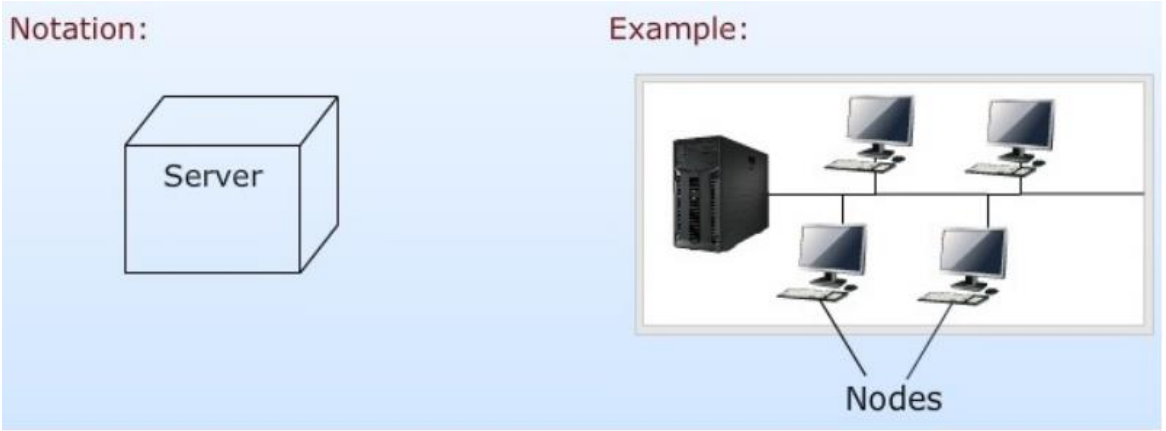
Bileşen - Bileşen, bir sistemin fiziksel bölümünü tanımlar.





Düğüm - Bir düğüm, çalışma zamanında var olan **fiziksel bir öğe** olarak tanımlanabilir. Düğüm, bilgisayar, sunucu, ağ vb. gibi bir sistemin fiziksel bölümünü temsil etmek için kullanılır.

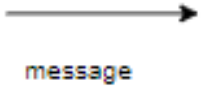




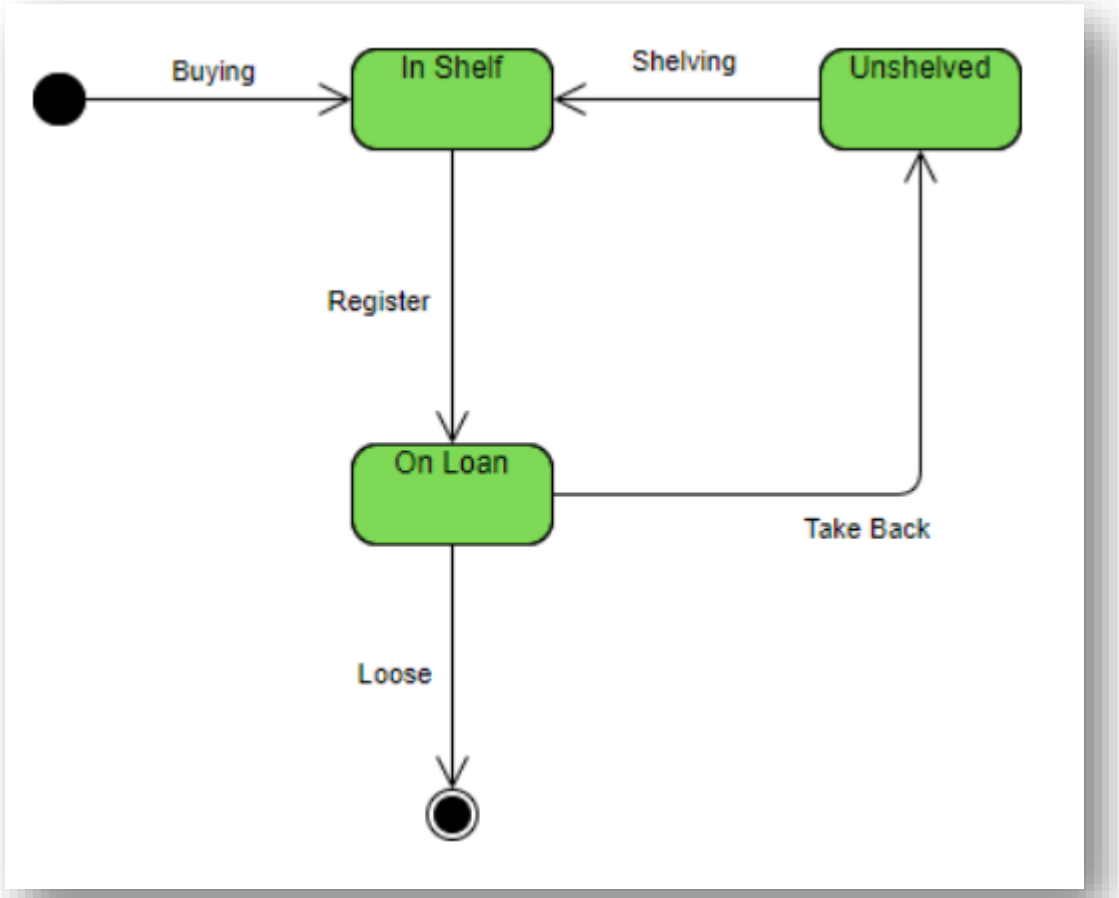
Davranışsal (Behavioral Things)

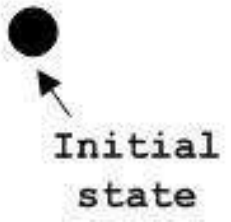
Davranışsal bir şey, UML modellerinin dinamik kısımlarından oluşur.

Etkileşim - Etkileşim, belirli bir görevi gerçekleştirmek için öğeler arasında değiş tokuş edilen bir grup mesajdan oluşan bir davranış olarak tanımlanır.



Durum makinesi - Durum makinesi, bir nesnenin yaşam döngüsündeki durumu önemli olduğunda kullanışlıdır. Bir nesnenin olaylara yanıt olarak geçtiği durumların sırasını tanımlar. Olaylar, durum değişikliğinden sorumlu dış faktörlerdir.

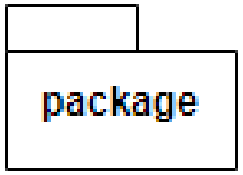


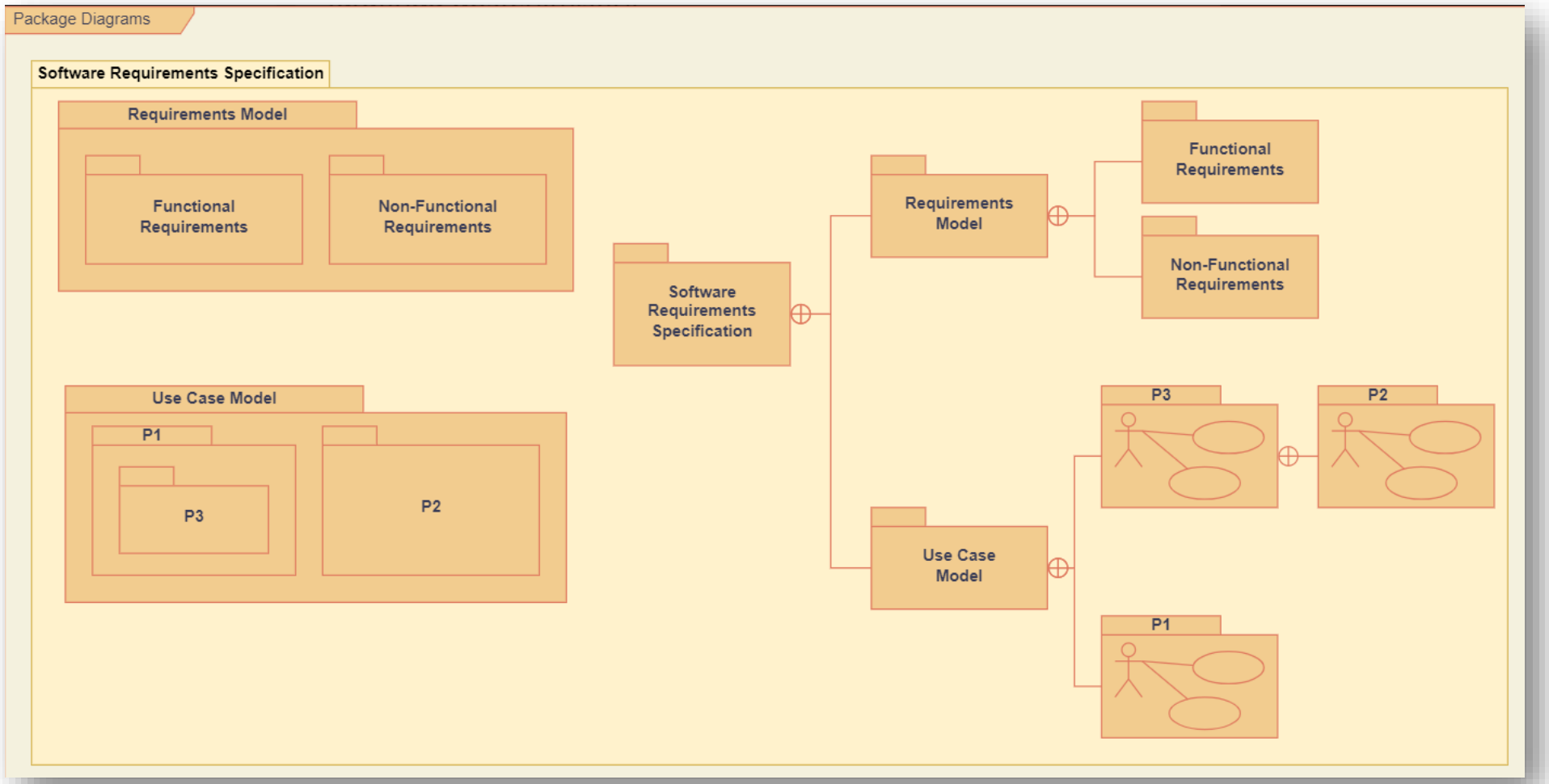


Gruplamalı (Grouping Things)

Öğeleri gruplamak, bir UML modelinin öğelerini birlikte gruplamak için bir mekanizma olarak tanımlanabilir.

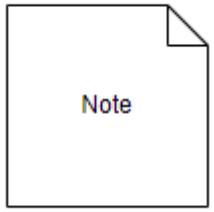
Group - Paket, yapısal ve davranışsal şeyleri toplamak için mevcut olan tek gruplama şeydir.

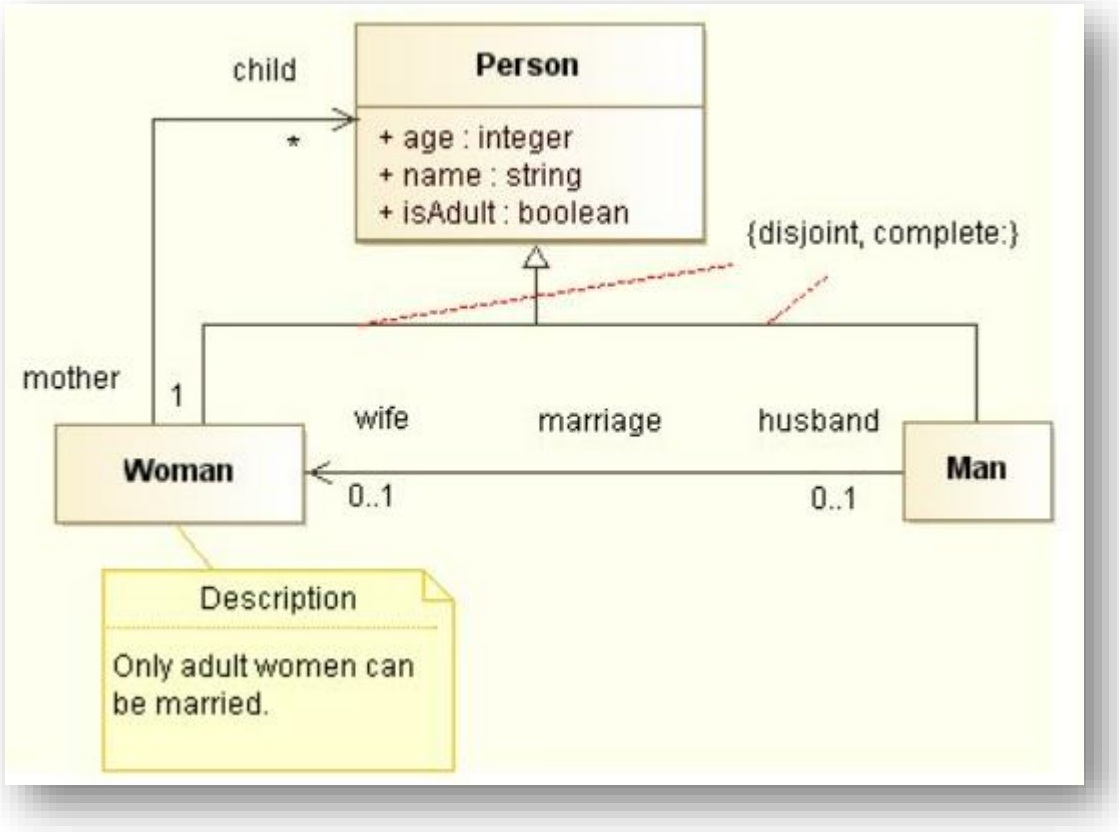




Açıklamalı(Annotational Things)

Açıklamalı şeyler, UML model öğelerinin açıklamalarını, açıklamalarını ve yorumlarını yakalamak için bir mekanizma olarak tanımlanabilir.





UML’nin Yapıtaşları

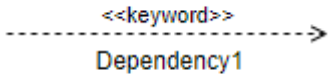
1. Nesneler
2. İlişkiler
3. Diyagramlar

İlişki (Relationship)

İlişki , UML'nin bir başka en önemli yapı taşıdır. Öğelerin birbirleriyle nasıl ilişkilendirildiğini gösterir ve bu ilişkilendirme, bir uygulamanın işlevselliğini açıklar.

Bağımlılık (Dependency):

Bağımlılık, bir öğedeki değişikliğin diğerini de etkilediği iki şey arasındaki ilişkidir.

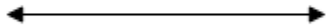


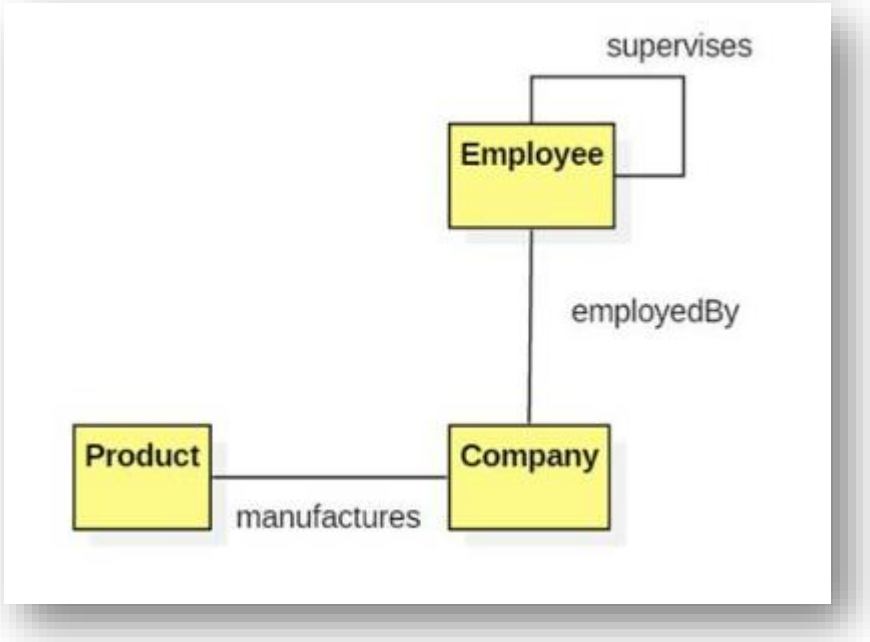
İstemcinin(clint) sağlayıcı(provider) tarafından sağlanan hizmetleri kullandığı anlamına gelir. Daha genel olarak bu, sağlayıcıda yapılan değişikliklerin istemcide de değişiklik yapılmasını gerektirebileceği anlamına gelir

İlişki (Relationship)

<u>İlişkilendirme (Association):</u>

İlişkilendirme, temel olarak bir UML modelinin öğelerini birbirine bağlayan bir dizi bağlantıdır. Ayrıca, bu ilişkide kaç nesnenin yer aldığını da açıklar.



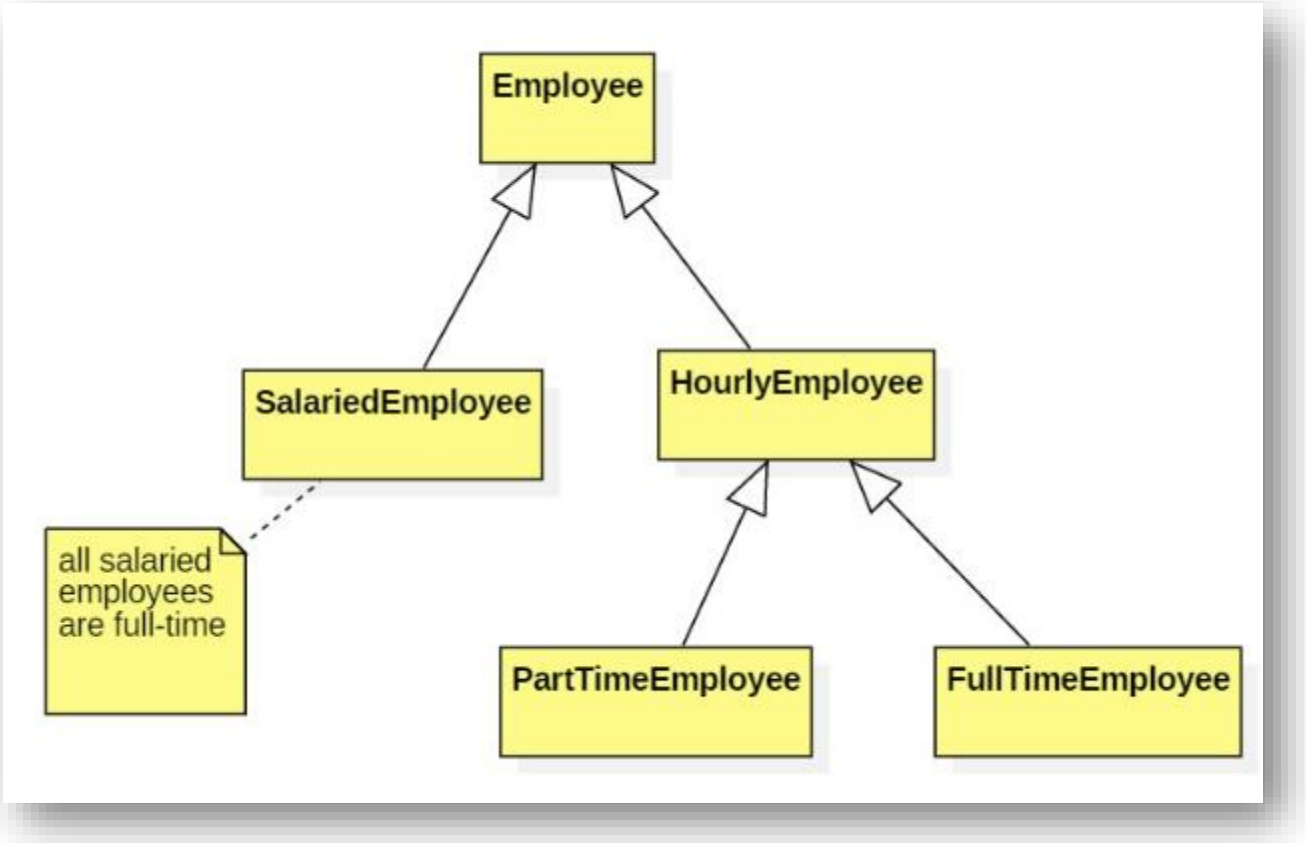


Çalışan X , Çalışan Y'yi ***denetler*** *,*
Çalışan X , Z Şirketi ***tarafından istihdam edilir*** *,*
Z Şirketi, P Ürününü ***üretir***

İlişki (Relationship)

Genelleme (Generalization):

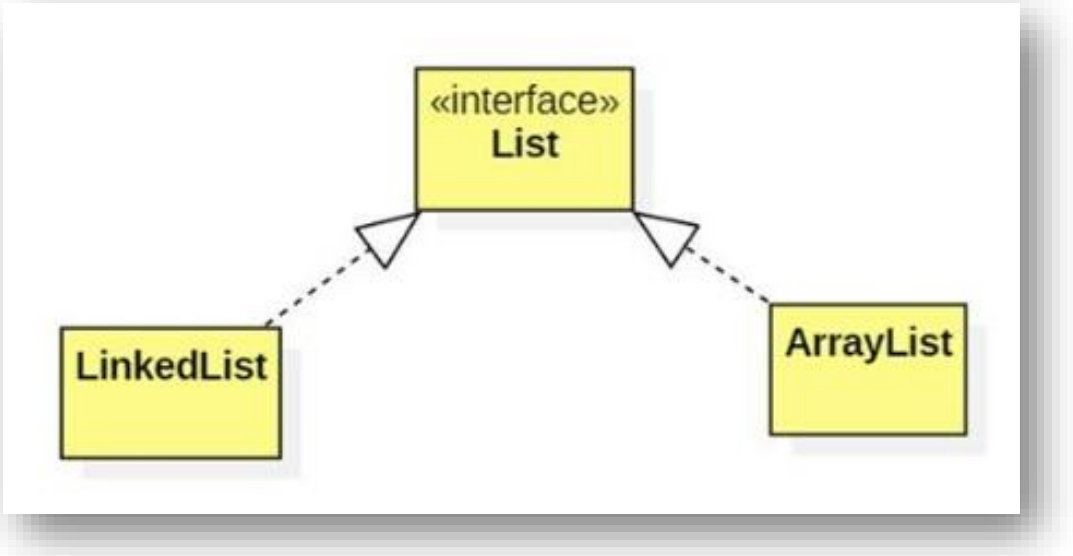
Genelleme, daha genel bir sınıflandırıcı (sınıf, kullanım durumu, bileşen vb.) ile daha spesifik bir sınıflandırıcı arasındaki ilişkidir. Temelde nesneler dünyasındaki kalıtım ilişkisini tanımlar.



İlişki (Relationship)

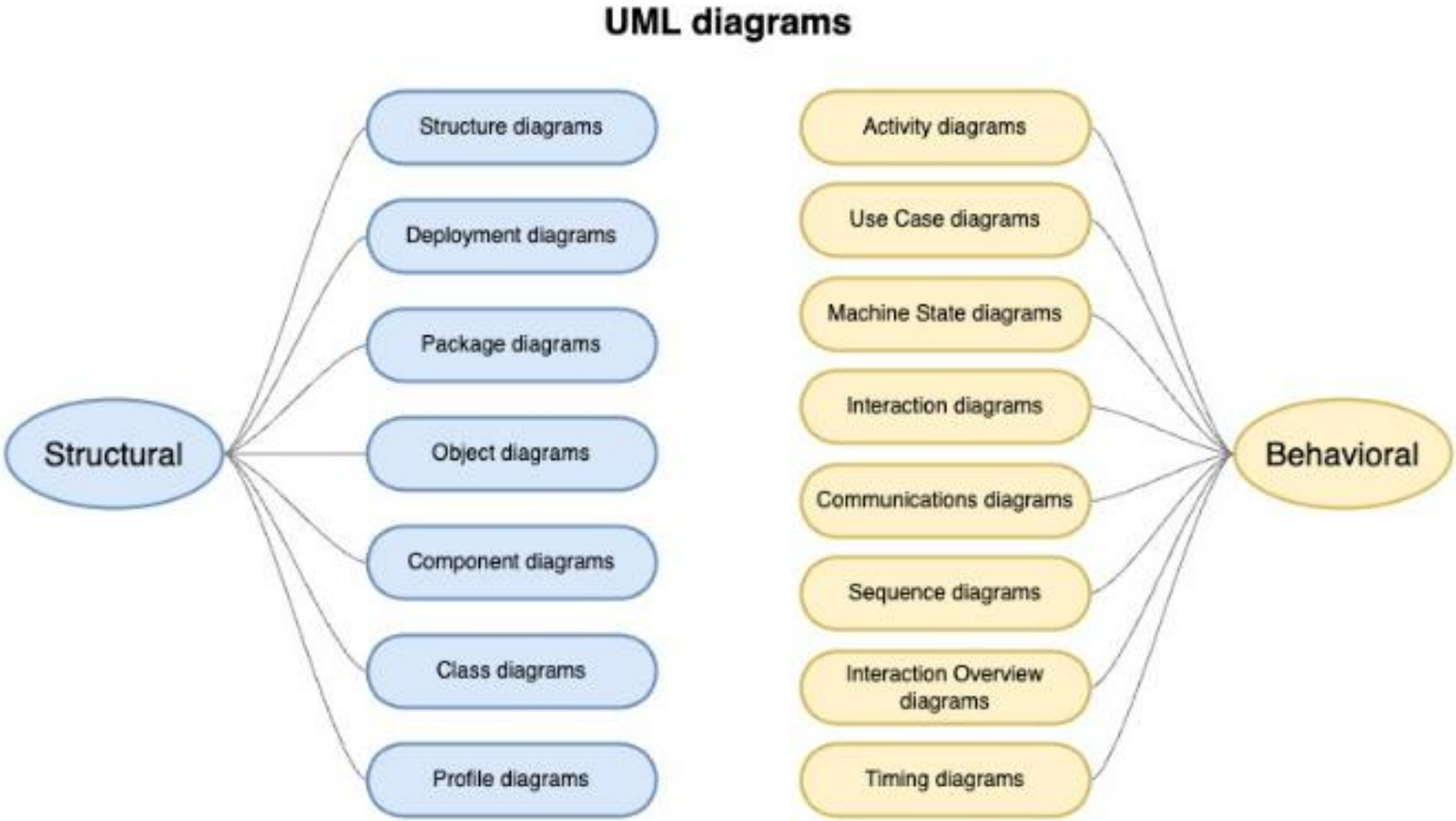
Gerçekleştirme (Realization):

Gerçekleşme, iki unsurun birbirine bağlı olduğu bir ilişki olarak tanımlanabilir. Öğelerden biri, uygulanmayan bazı sorumlulukları tanımlarken, diğeri bunları uygular. Bu ilişki, arayüzler durumunda mevcuttur.



UML’nin Yapıtaşları

1. Nesneler
2. İlişkiler
3. Diyagramlar

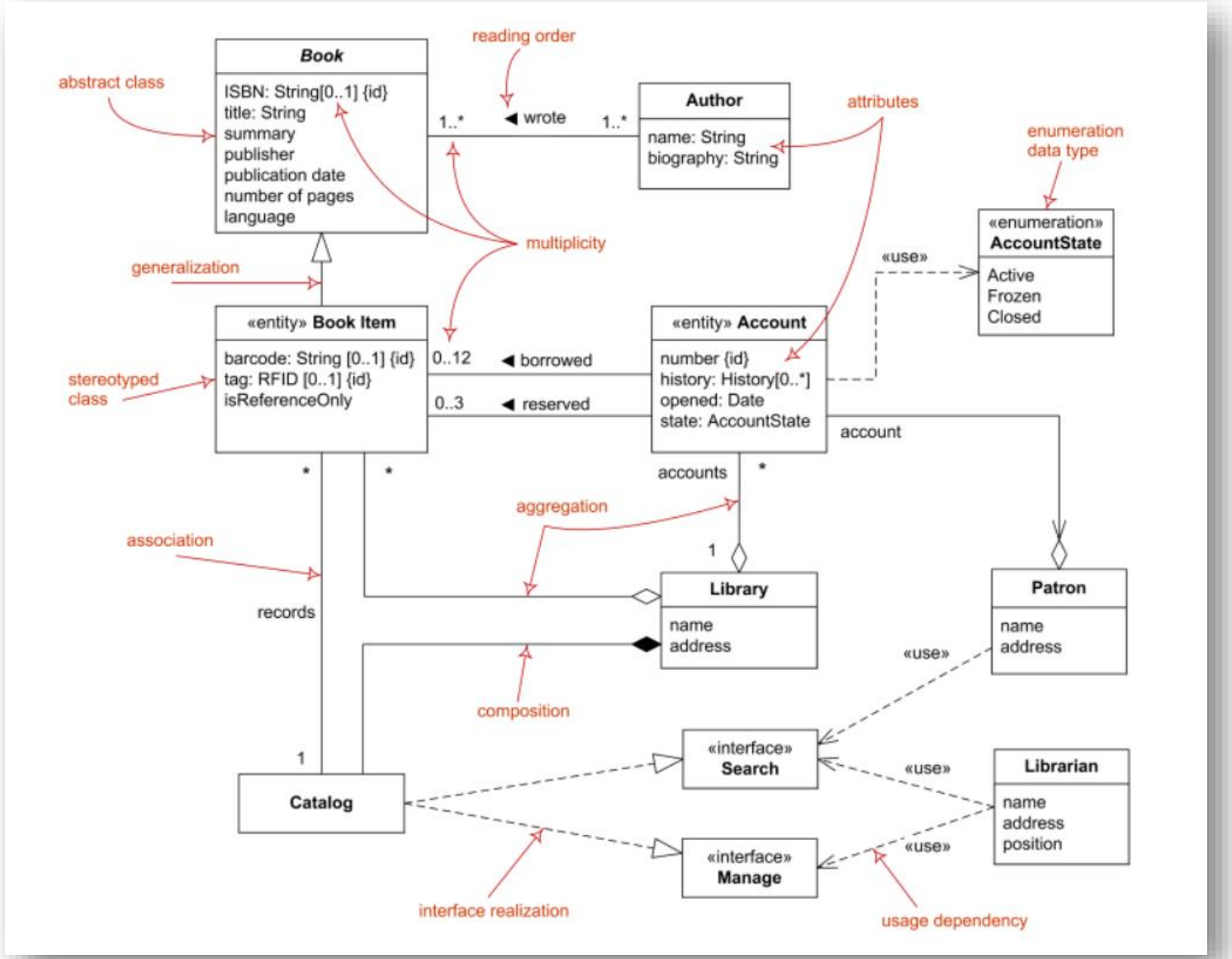


Kaynak: https://drawio-app.com/blog/i-%f0%9f%a7%a1-draw-io-three-ways-to-improve-your-uml-diagrams/
https://www.uml-diagrams.org/uml-24-diagrams.html

Yapısal Modelleme

Sınıf diyagramları:

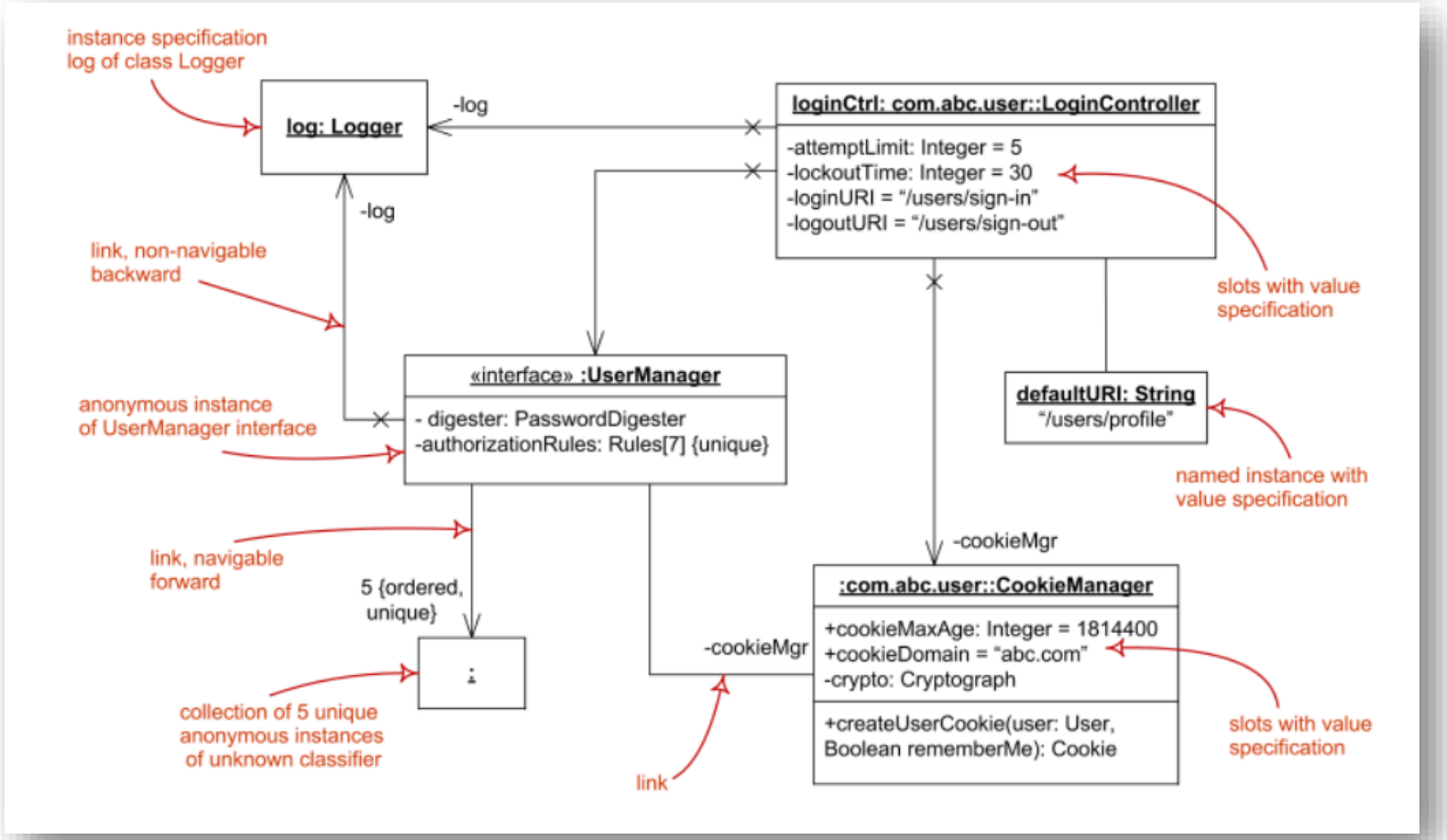
- Sınıf diyagramları, bir sistemin sınıflarını, özelliklerini, metotlarını ve aralarındaki ilişkileri göstererek sistemin yapısını temsil eden bir diyagram türüdür.
- Sınıf diyagramları genellikle nesne yönelimli programlama ve yazılım geliştirme alanında kullanılır.



Yapısal Modelleme

Nesne diyagramları:

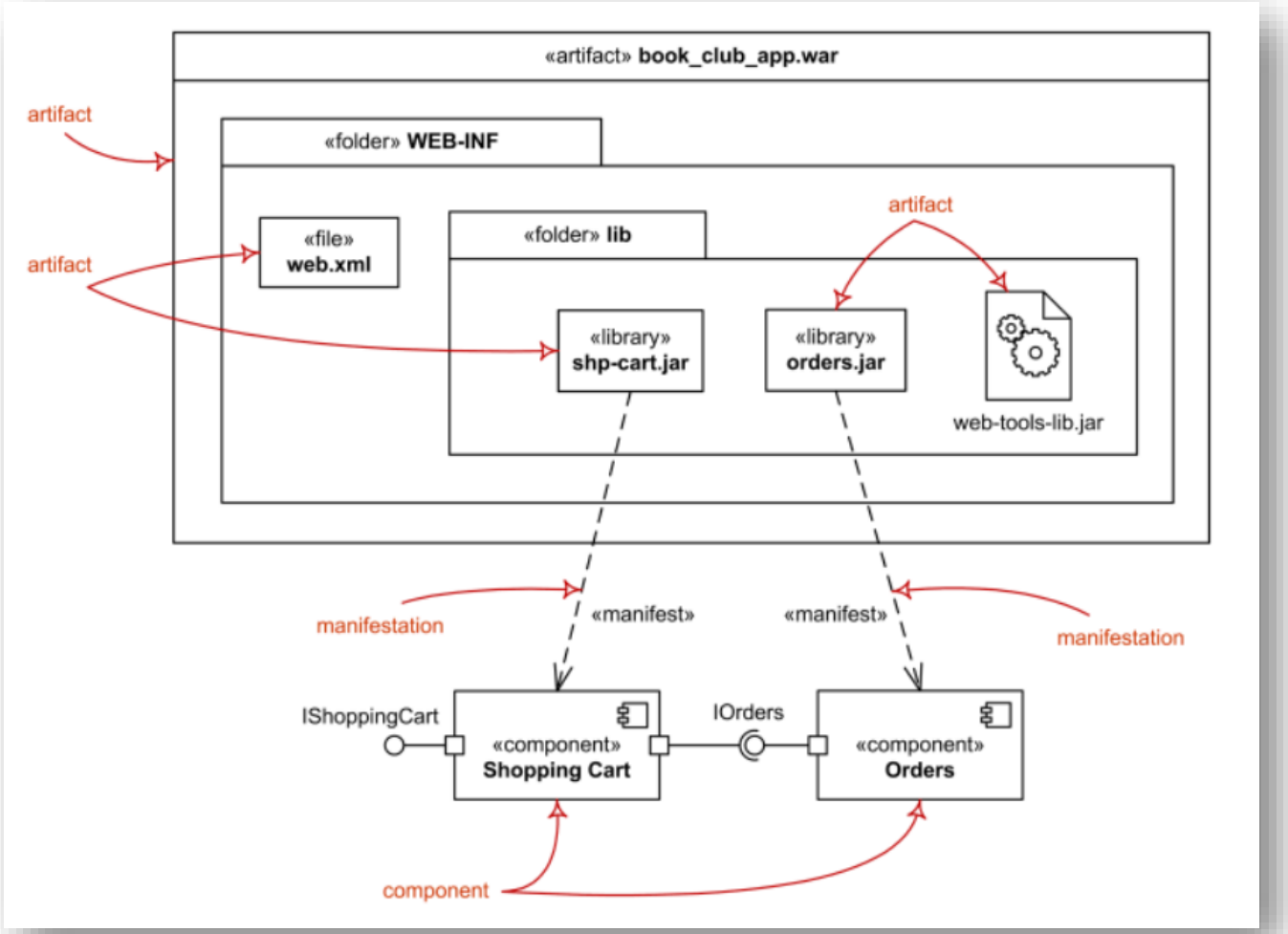
- Nesne diyagramları, bir sistemin belirli bir anda nesnelerinin ve aralarındaki ilişkilerin bir anlık görüntüsünü göstermek için kullanılır.
- Sınıf diyagramlarına benzerler, ancak sınıfları ve aralarındaki ilişkileri temsil etmek yerine, nesne diyagramları sınıfların örneklerini ve aralarındaki ilişkileri gösterir.



Yapısal Modelleme

Dağıtım diyagramları:

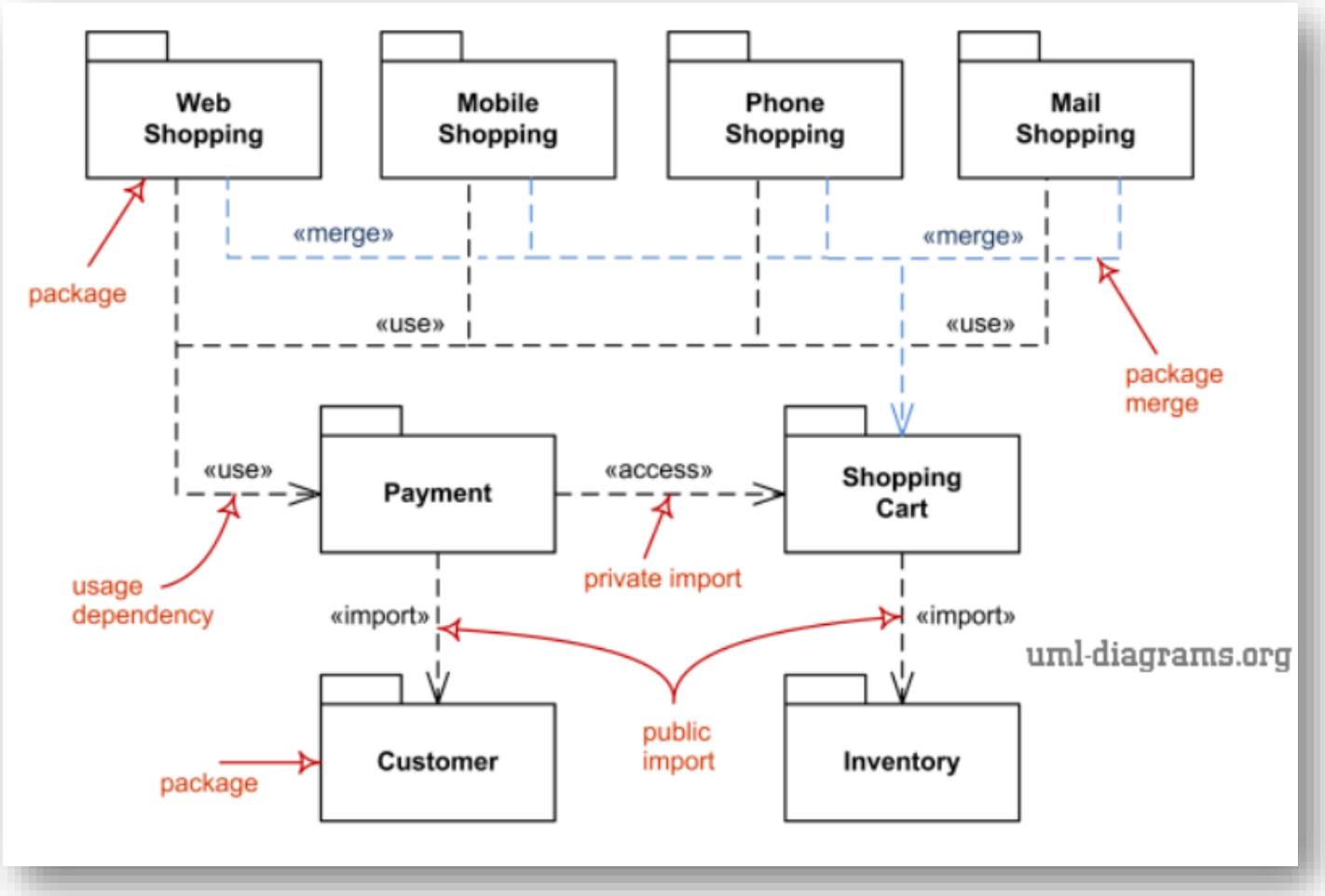
- Dağıtım diyagramları, bir sistemin bileşenlerinin fiziksel dağıtımını göstermek için kullanılır.
- Sunucular, işlemciler ve iletişim kanalları dahil olmak üzere sistemin işlemesi için gereken donanım ve yazılım altyapısını gösterirler.



Yapısal Modelleme

Paket diyagramları:

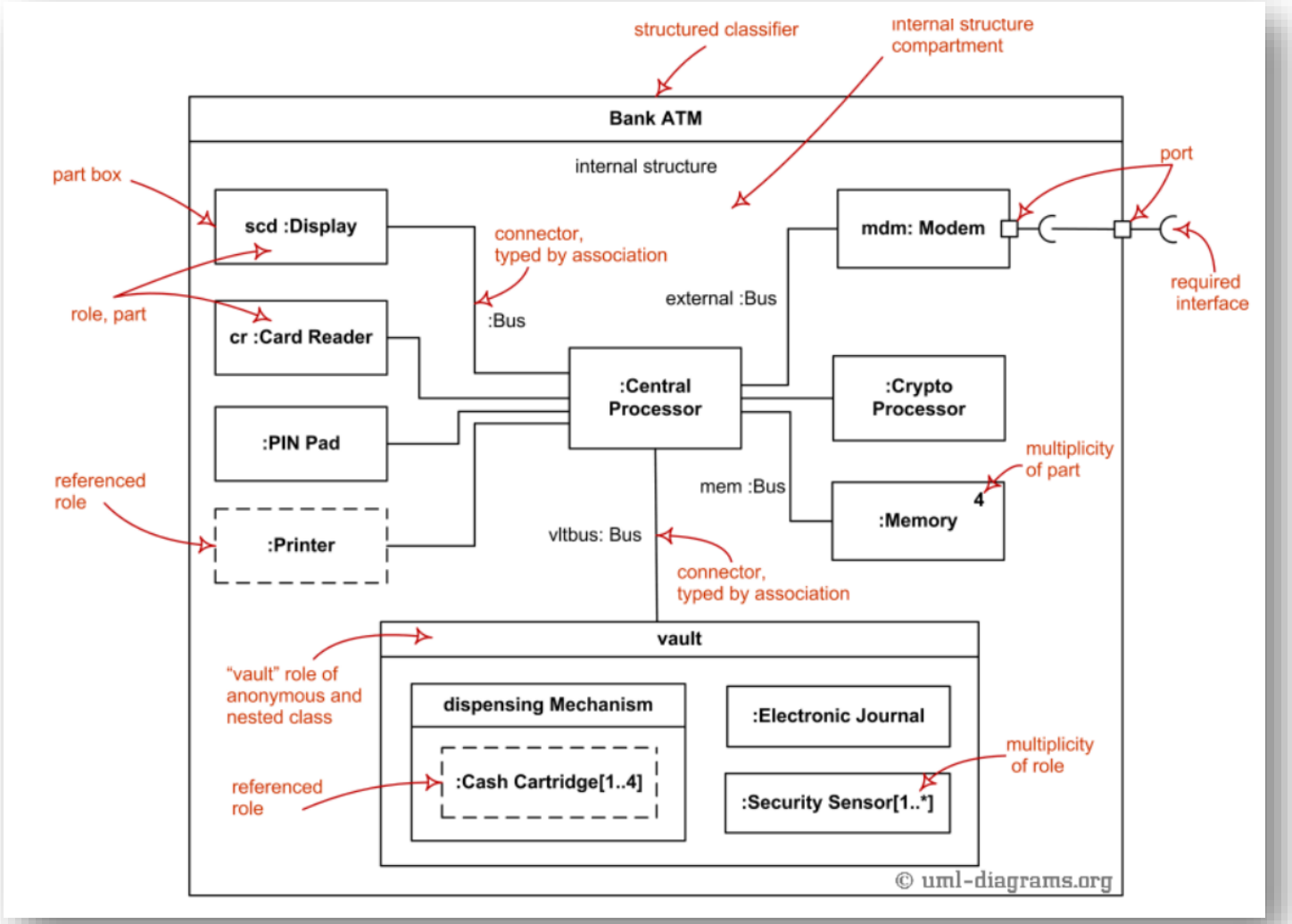
- Paket diyagramları, paketler ve içerikleri arasındaki ilişkileri göstermek için kullanılır. Bir paket, ilgili sınıfların, bileşenlerin ve diğer unsurların bir koleksiyonudur.
- Paket diyagramları, sınıf diyagramlarından daha yüksek bir seviyede bir sistemin yapısını göstermek için kullanılabilir, paketlerin ve alt sistemlerin nasıl düzenlendiğini gösterirler.



Yapısal Modelleme

Kompozit yapı diyagramları:

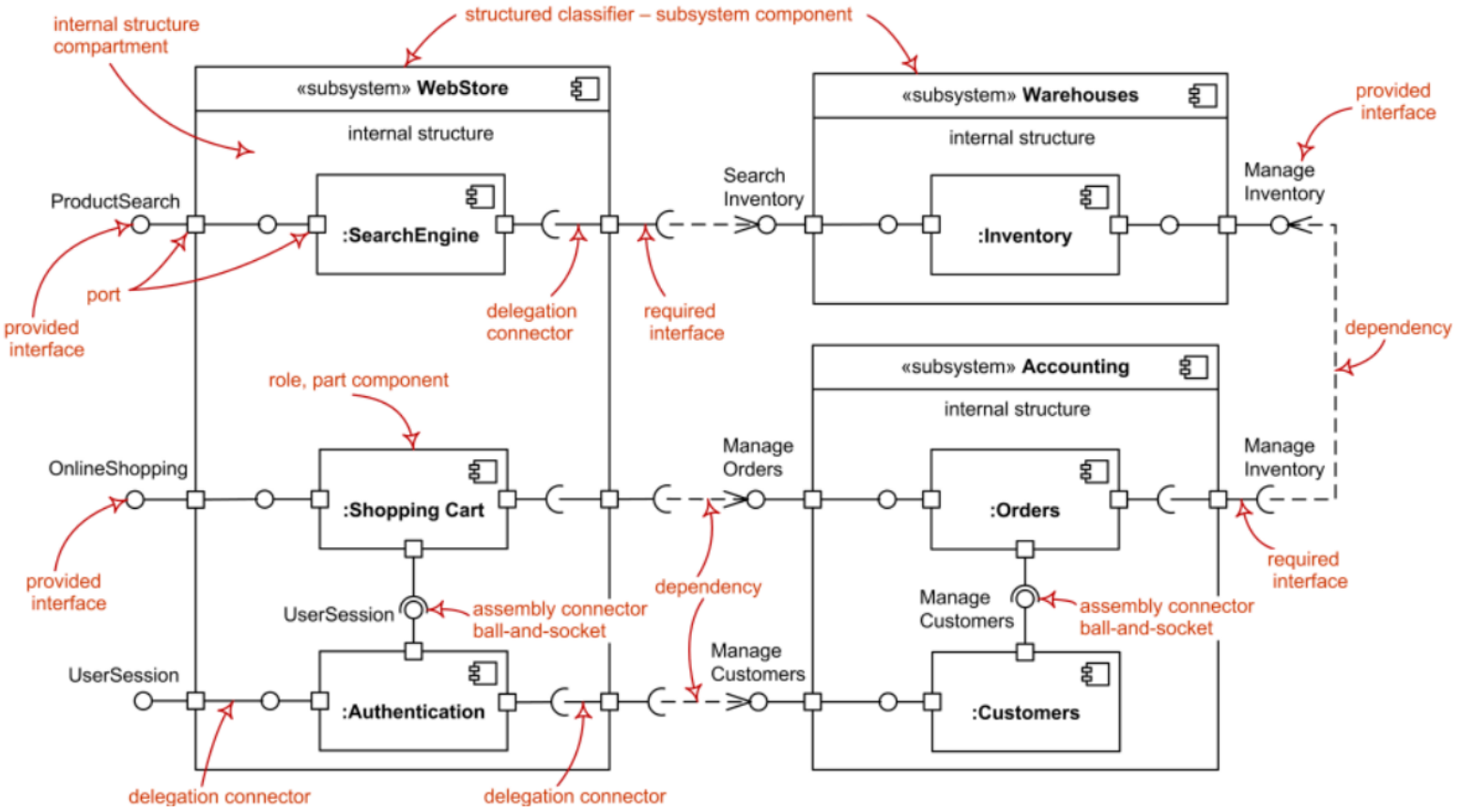
- Kompozit yapı diyagramları, bir sınıfın veya bileşenin iç yapısını, parçalarını, bağlantı noktalarını ve bağlayıcılarını göstermek için kullanılır.
- Karmaşık bir sistemin daha küçük, daha yönetilebilir bileşenlere nasıl ayrılabileceğini gösterirler.



Yapısal Modelleme

Bileşen diyagramları:

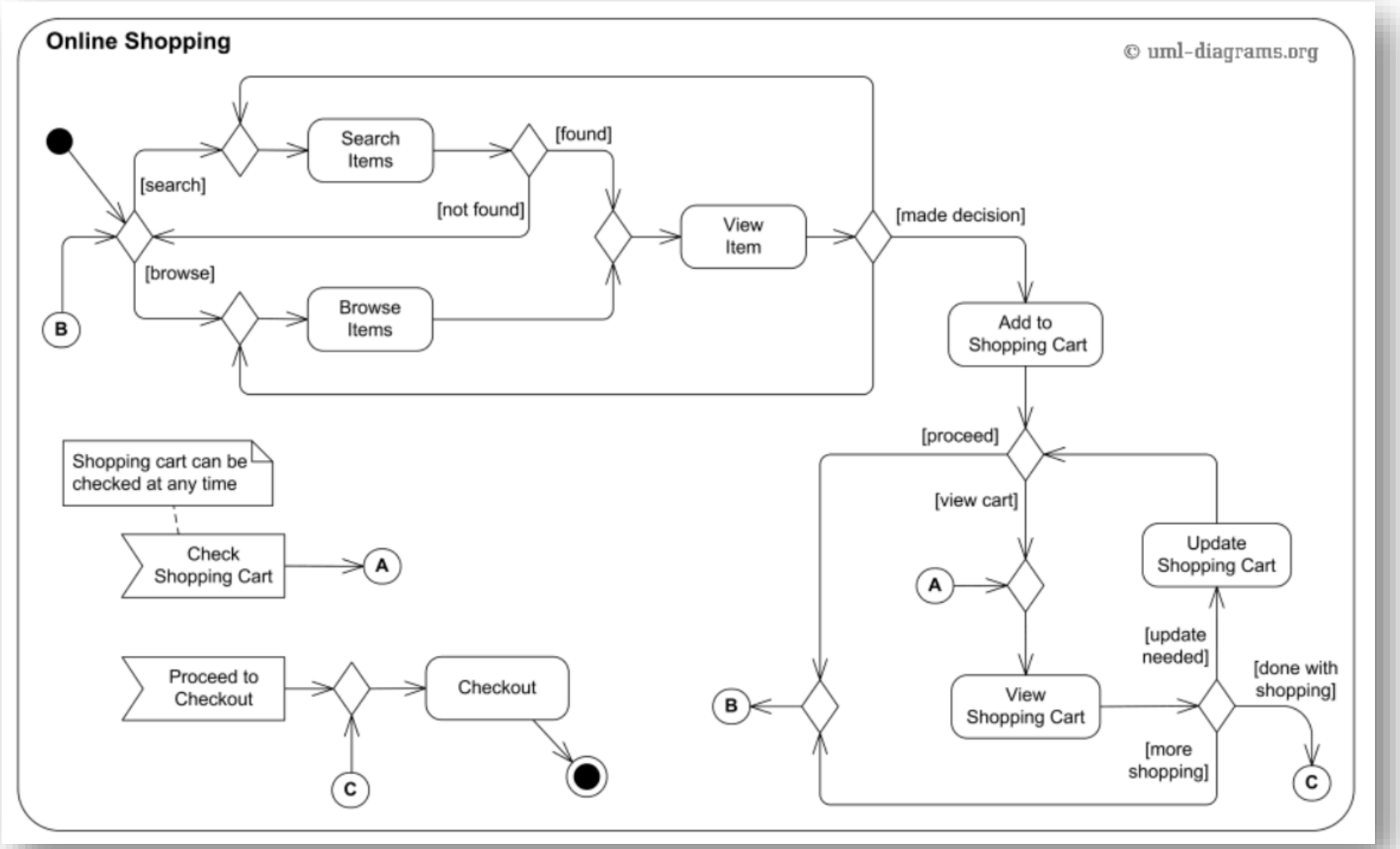
- Bileşen diyagramları, bir sistemin yazılım bileşenlerinin yapısını ve bağımlılıklarını göstermek için kullanılır.
- Bileşenlerin nasıl bağlandığı ve birbirleriyle nasıl iletişim kurduğu gösterilir ve modüler ve ölçeklenebilir bir şekilde karmaşık sistemler tasarlamak ve geliştirmek için kullanılabilirler.



Davranışsal Modelleme

Activity diagramları:

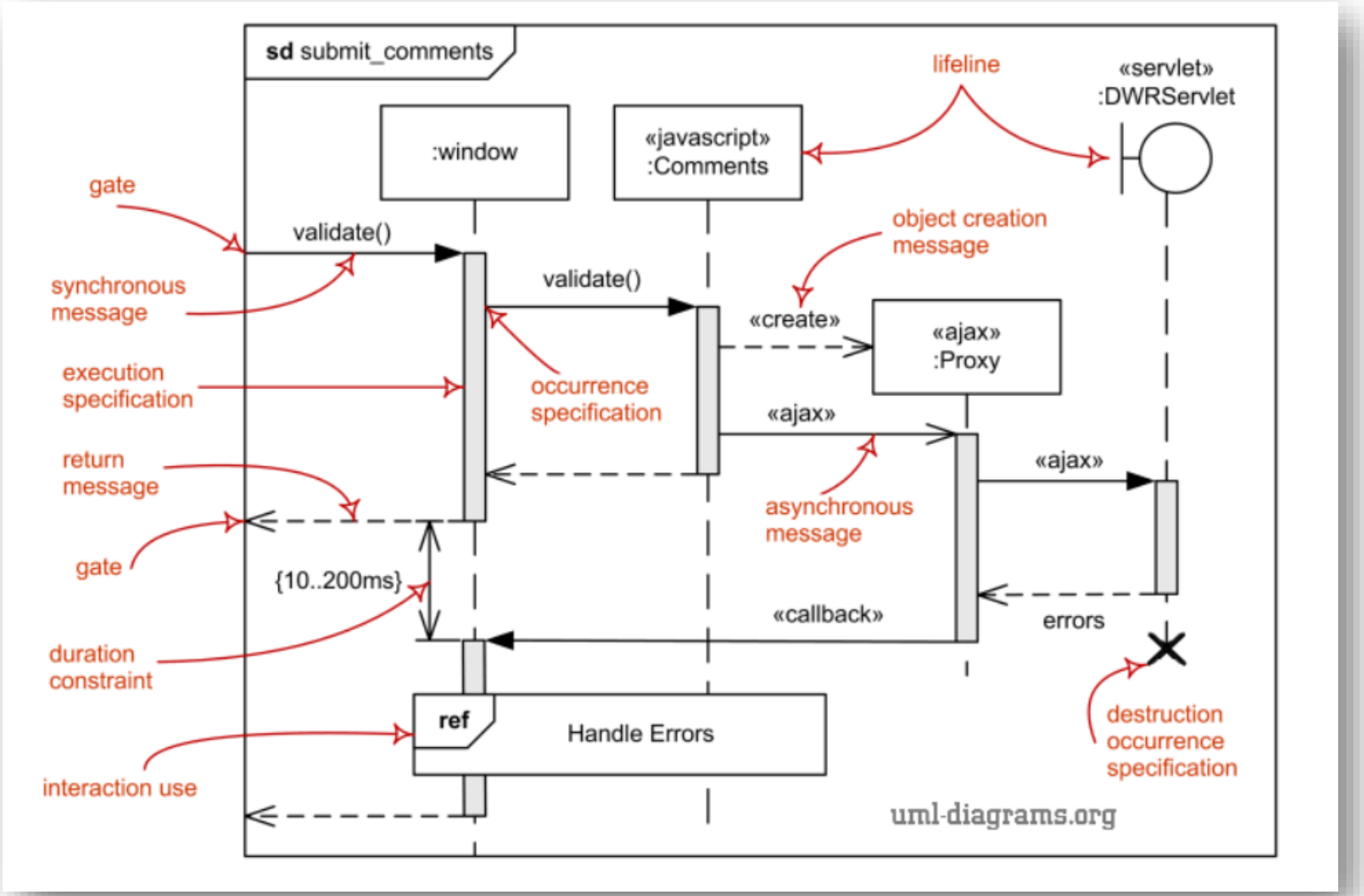
- Activity diyagramları, bir sürecin veya aktivitenin akışını göstermek için kullanılan bir diyagram türüdür.
- Adımları ve adımlar arasındaki ilişkileri gösterirler.
- İş akışı modellemesi, işletme analizi, yazılım geliştirme ve diğer alanlarda kullanılabilirler.



Davranışsal Modelleme

Sıra diyagramları:

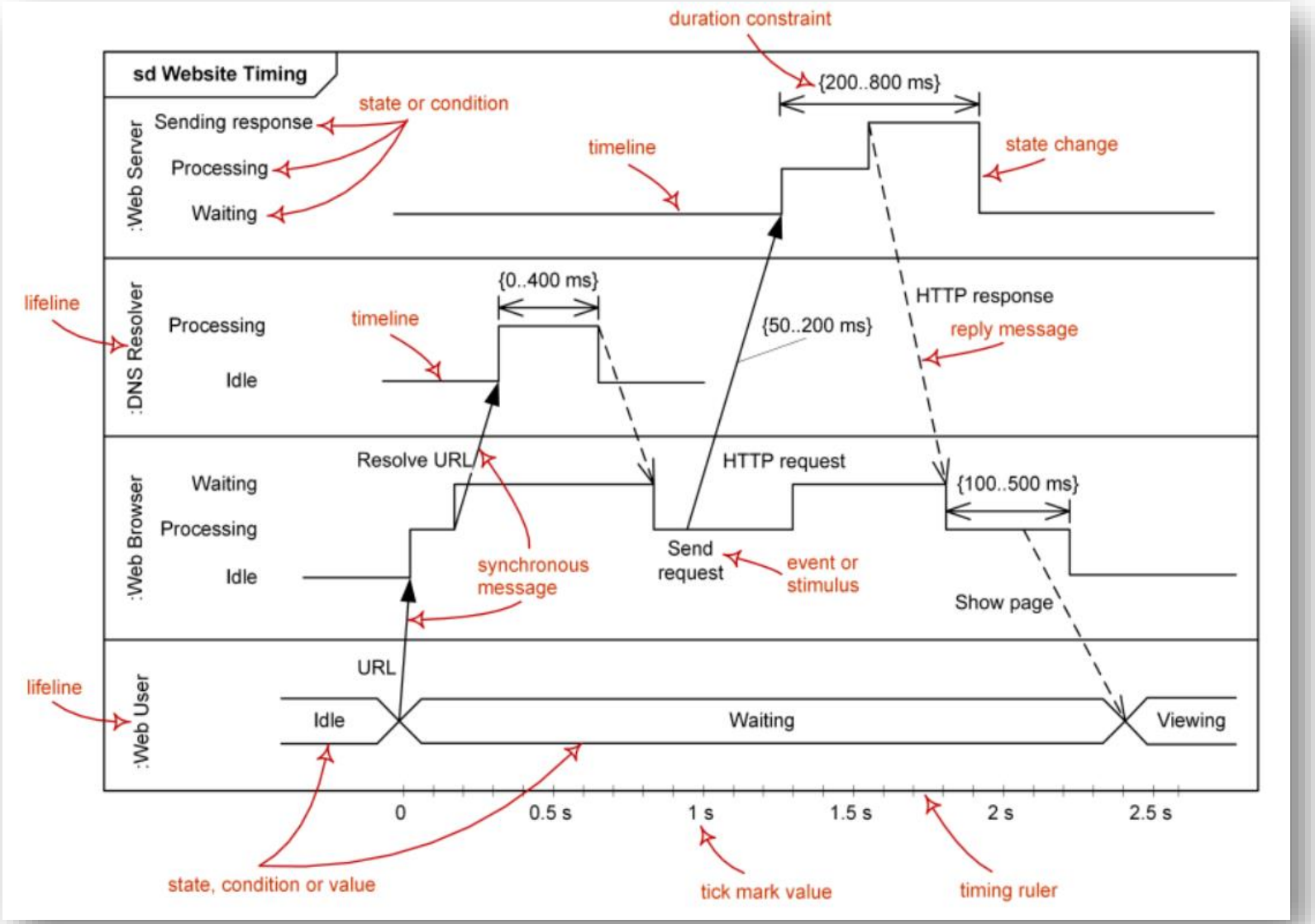
- Bir dizi yaşam çizgisi arasındaki mesaj alışverişine odaklanan en yaygın etkileşim diyagramıdır .
- Sıra diyagramı, değiştirilen mesajların sırasına ve bunların yaşam hatlarında karşılık gelen oluşum özelliklerine odaklanarak bir etkileşimi tanımlar.



Davranışsal Modelleme

Zamanlama Diyagramları:

- Zamanlama diyagramları, diyagramın temel amacının zaman hakkında akıl yürütmek olduğu durumlarda etkileşimleri göstermek için kullanılan UML etkileşim diyagramlarıdır.
- Zamanlama diyagramları, doğrusal buir zaman ekseni boyunca yaşam çizgileri içinde ve arasında değişen koşullara odaklanır.
- Zamanlama diyagramları , yaşam hatlarının modellenen koşullarında değişikliklere neden olan olayların zamanına dikkat çekerek, hem bireysel sınıflandırıcıların davranışlarını hem de sınıflandırıcıların etkileşimlerini tanımlar.



Davranışsal Modelleme

Use case diagrams:

- Kullanım durumu diyagramları, bir sistem veya yazılımın kullanım senaryolarını ve aktörleri gösteren bir diyagram türüdür.
- Kullanım durumları, bir kullanıcının veya sistemin diğer bileşenlerinin sistemi nasıl kullanabileceğini gösterir.
- Aktörler, bir sistemle etkileşime giren insanlar, diğer sistemler veya bileşenlerdir. Kullanım durumu diyagramları, sistem gereksinimlerini belirlemek, tasarlamak ve doğrulamak için kullanılır.