

11. Hafta İçeriği

► Modelleme Nedir?

- Avantajları

► UML Nedir?

► Sınıf Diyagramları

► Kullanım Senaryusu Diyagramları (Use Case)

Modelleme Nedir?

- ▶ Gerçek hayatı karşılaşılan sistemler genellikle karmaşıktır ve bu sistemlerin kişiler tarafından tüm yönleriyle bir defada kavranabilmesi için, sistemin parçalar halinde incelenmesini ve tasarlamanı sağlayacak başka yöntemlerin kullanılması gereklidir. Modelleme kavramı, insanlığın sistemlerin karmaşıklığı ile baş etmede kullandığı bilinen en eski ve en etkin yöntemdir.
- ▶ Bir sistem modellenirken, sistemin özelliklerinden o an için ilgilenilen kısımlar öne çıkarılırken, diğerlerini arka planda saklayan soyut yapılar kurulur.
- ▶ "Hiçbir model, tanımı gereği, sistemin tüm özelliklerini gerçekçe özdeş biçimde içermez. Her model asının yalınlaştırılmış bir kopyasıdır ve tüm ayrıntıları içinde barındırması olanaklı değildir."

Modelleme Nedir?

- ▶ Yazılımcı tümüyle soyut yapılar üzerinde çalışır. Tasarım ve programla süresince kullandığı listeler, döngüler, nesneler, veritabanı tabloları gibi yapıların hiçbirisi elle tutulabilir somut nesneler değildir. Geliştirdiği sistemin, örneğin mimarların yaptığı gibi, bir maketinin yapılması ve masanın üzerine konarak incelenmesi de mümkün değildir. Bu nedenle yazılımcı soyut yapılarla çalışabilme ve soyut düşünübilme yeteneğini olabildiğince geliştirmek zorundadır.
- ▶ Modeller, sistem karmaşıklığını yönetilir boyutlara indirgemeyenin yanı sıra, tasarımcılar arasında bir etkileşim biçimi olarak da hizmet veriler. Bir tasarımın temel özelliklerinin açıklanması ve çeşitli seçeneklerin değerlendirilmesi bir model üzerinde daha etkin olarak yapılabilir. Bunun yapılabilmesi için modellemede ortak bir gösterim biçiminin, bir başka deyişle ortak bir modelleme dilinin kullanılması zorunludur.

UML Nedir?

- ▶ **UML (Unified Modelling Language)** Türkçe olarak "Birleşik Modelleme Dili" şeklinde adlandırılabilir. UML bir programlama (ya da yazılım geliştirme) dili olmaktan ziyade iş sistemlerinin nasıl modellenebileceğini belirleyen ve açıklayan yöntemlerin bir araya toplanmış halidir. Daha çok yazılım geliştiriciler tarafından kullanılıyor olsa da UML ile yapılan modellemeler sadece yazılım projelerinde kullanılmak zorunda değildir: **Resmi UML dokumantasyonlarında UML'in yazılımın yanısıra "İş Sistemleri Modellenmesi"**nde de kullanılabilir.
- ▶ Örneğin bir iş sistemin yapısını sade ve anlaşılır şekilde ortaya çıkarmak için Paket Diyagramı ("Package Diagram") kullanılabilir. Sınıf Diyagramı ("Class Diagram") vasıtası ile Nesnel Yönelimli Programlamada temel teşkil eden sınıflar net şekilde gösterilebilir ve böylece sağlanan ek görsellik ile yazılım tasarlamanın ilerleyen aşamalarında daha yüksek verimlilik sağlanabilir.

UML Diyagramları

- UML 2.0, 3 ana bölümde 13 çeşit diyagram içerir. Yapısal diyagamlarda modellenen sistemde nelerin var olması gereği vurgulanır. Davranış diyagamlarında modellenen sistemde nelerin meydana gelmesi gerektiğini belirtir. Davranış diyagamlarının bir alt kümesi olan Etkileşim diyagamlarında ise modellenen sistemdeki elemanlar arasındaki veri ve komut akışı gösterilir. UML Diyagamları 3 başlık altında toplanabilir.

► **Yapısal Diyagamlar**

- 1. Sınıf (Class) diyagamı**, sistemin yapısını anlatmak için sistemde var olan sınıfları, sınıfların özelliklerini ve sınıflar arası ilişkileri kullanır. Nesneye yönelik sistemleri modellemede kullanılan en yaygın diyagramdır.
- 2. Nesne (Object) diyagamı**, modellenen sistemin yapısının belirli bir andaki bütün ya da kısmi görünüşü tarif edilir.
- 3. Bileşen (Component) diyagamı**, bir yazılım sisteminin hangi tür bileşenlere ayrıldığını ve bu bileşenlerin nasıl birbiriyle ilişkili olduğunu betimler. Bir bileşen genellikle bir veya birden fazla sınıf, arayüz ve iletişime karşılık gelir.

UML Diyagramları

- 4. Paket (Package) diyagramı**, bir sistemin hangi mantıksal grplara bölündüğünü ve bu gruplar arasındaki bağımlılıkları betimler.
- 5. Dağılım (Deployment) diyagramı**, sistemde kullanılan donanımları, bu donanımların içinde yer alan bileşenleri ve bu bileşenlerin arasındaki bağlantıları gösterir.
- 6. Birleşik Yapı (Composite Structure) diyagramı**, bir sınıfın iç yapısını ve bu yapının mümkün kıldığı iletişimleri tarif eder.

► Davranış Diyagramları

- 1. Kullanım Senaryosu (Use-Case) diyagramı**, modellenen sistem tarafından sağlanan işlevselligi sistemde yer alan aktörleri, aktörlerin sahip olduğu kullanım senaryolarını ve bu senaryolar arasındaki bağımlılıkları göstererek açıklar.
- 2. Durum (Statechart) diyagramı**, bilgisayar programlarından iş süreçlerine kadar birçok sistemi tarif eden standartlaşmış bir gösterimdir. Durumlar, geçişler, olaylar ve faaliyetler gösterilir.
- 3. Faaliyet (Activity) diyagramı**, modellenen sistemdeki iş akışını adım adım gösterir. Faaliyet diyagramı kapsamlı bir komut akışını tarif eder. Faaliyetler arası akışı gösteren Durum diyagramıdır.

UML Diyagramları

► Etkileşim Diyagramları

- 1. Sıralama (Sequence) diyagramı**, nesnelerin birbiriyle nasıl iletişim sağladıklarını sıralı iletiler şeklinde gösterir. Ayrıca nesnelerin yaşam süreleri de gösterilir.
- 2. İletişim (Communication) diyagramı**, nesneler ve parçalar arasındaki etkileşimi sıralı iletiler olarak gösterir. Sınıf, Sıralama ve Kullanım Senaryoları diyagramlarındaki bilgileri kullanarak sistemin hem statik yapısını hem de dinamik davranışını gösterir.
- 3. Etkileşime Bakış (Interaction Overview) diyagramı**, farklı etkileşim diyagramları kullanarak, bunlar arasındaki komut akışını gösterir. Bir başka deyişle, elemanları etkileşim diyagramları olan faaliyet diyagramlarıdır.
- 4. Zaman Aķış (Timing) diyagramı**, odağın zaman kısıtlamaları olduğu etkileşim diyagramıdır.

UML'in yararları

- ▶ Takım çalışmasında yardımcı olur, UML standartlaşmış uluslararası bir dildir ve bu dili bilen herkes diyagamlardan aynı şeyleri anlar. Müşteri ve teknik sorumlular diyagramlar üzerinden rahatça iletişim kurabilirler. Ekibinizde yer alan çalışma arkadaşlarınızla uyumlu bir şekilde çalışabilirsiniz ve ekibe yeni giren bir çalışan da projeye rahatlıkla dahil edilebilir.
- ▶ Kodlamayı kolaylaştırır, UML ile uygulamanızın tasarımlı analiz aşamasında yapıldığı için, modellemeniz bittikten hemen sonra kod yazmaya başlayabilirsiniz.
- ▶ Hataları en aza indirir, UML ile bütün sistem tasarlandığı için sistemde hata çıkma olasılığı azdır. Çıkan hataları düzeltmek ise çok daha kolaydır.

UML'in yararları

- ▶ Tekrar kullanılabılır bileşenleriniz artar, UML ile tüm sistem ve sistemin bileşenleri daha baştan belirlendiği için, o bölümler tekrar tekrar yazılmayacaktır.
- ▶ Program kararlılığı artar, UML ile ayrıntılı gereksinim analizleri yapıldıktan sonra senaryolar belirlenir. Senaryoların baştan belirli olması programınızı daha kararlı hale getirmenizde size yardımcı olur.

UML Class Diyagramları

- ▶ Bir sınıf, ortak yapısı, ortak davranışları, ortak ilişkileri ve ortak semantiği bulunan nesneler koleksiyonudur.
- ▶ UML' de üç farklı alanı olan bir dikdörtgen şeklinde gösterilirler.
- ▶ Bu üç bölümden ilki sınıf ismini, ikinci kısım yapısını(attributes), ve üçüncü bölüm ise davranışını(operations) gösterir.
- ▶ Sınıfların gösteriminde sadece sınıf ismini, yapısını ya da davranışlarını veya her üçünü de birden görebilirsiniz.
- ▶ Sınıflar isimlendirilirken bir standartizasyon olması amacıyla bütün isimler büyük harf ile başlarlar.

UML Class Diyagramları

Sınıf Adı Bölgesi: Sınıfların ve Interfacelerin adının yazıldığı bölgedir. Sınıf abstract ise sınıf adı italik yazılır. Interface olması durumunda ise bu sınıfın arayüz olduğu belirtebilmek amacıyla <<interface>> şeklinde stereotypes olarak belirtilir.

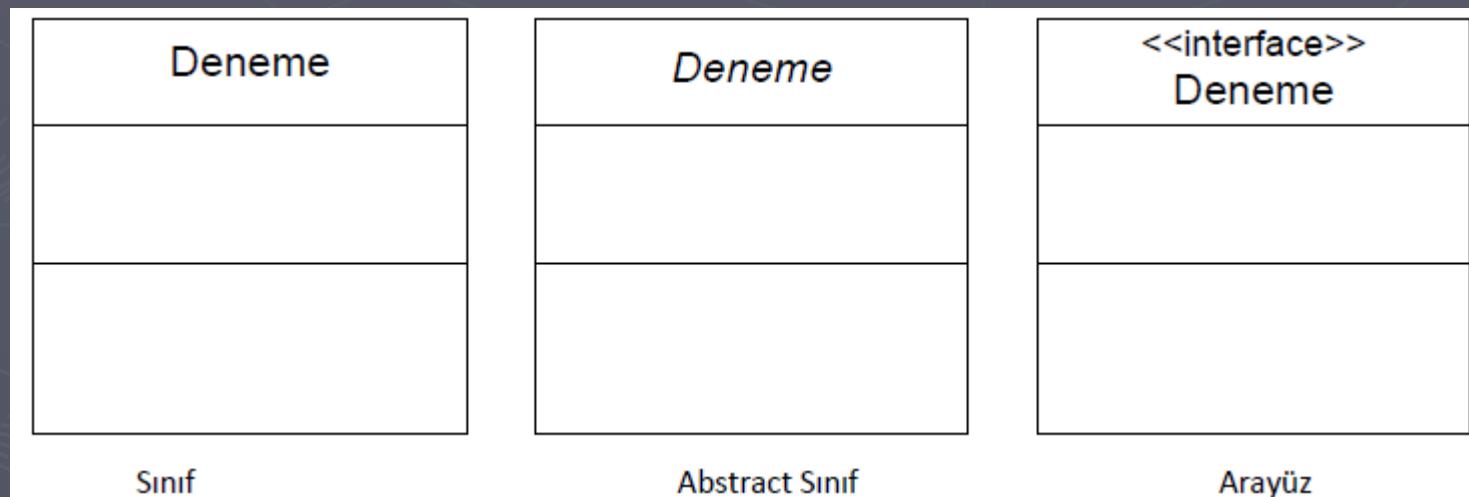
Attribute Bölgesi: Bu bölgede sınıf üyelerinden alanlar yer alır. Öncelikle erişim belirtice yazılır . Üyenin adından sonra : (iki nokta üst üste) yazılır ve tipi belirtilir.

Operation Bölgesi: Bu bölgede sınıf üyelerinden metodlar vb yer alır. Öncelikle erişim belirtice yazılır. Üyenin adından sonra varsa metod parametrelerinin türleri parantez içersinde yazılır. : (iki nokta üst üste) yazılır ve geri dönüş tipi belirtilir



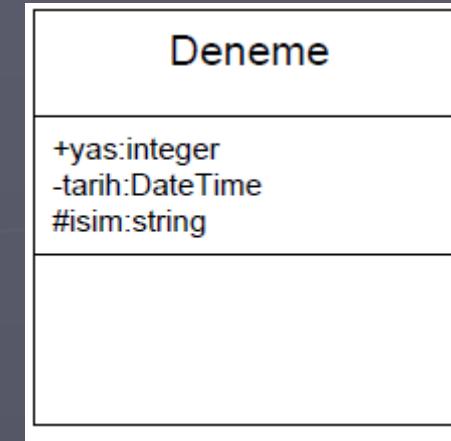
Sınıf Adı Bölgesi

- ▶ İlk harf büyük olmalı
- ▶ Sınıf abstract sınıf ise sınıf ismi italic
- ▶ Interface ise << interface >> stereotype ifadesi ismin üzerine yazılmalı



Attribute Bölgesi

- ▶ Bu bölgede sınıf üyelerinden alanlar yer alır.



▶ Görünürlük(Visibility)

Public: UML'de + simbolü ile gösterilir.

Protected: UML'de # simbolü ile gösterilir.

Package: aynı paketteki diğer sınıflar tarafından erişilebilir. UML'de ~ simbolü ile gösterilir.

Private: UML'de - simbolü ile gösterilir.

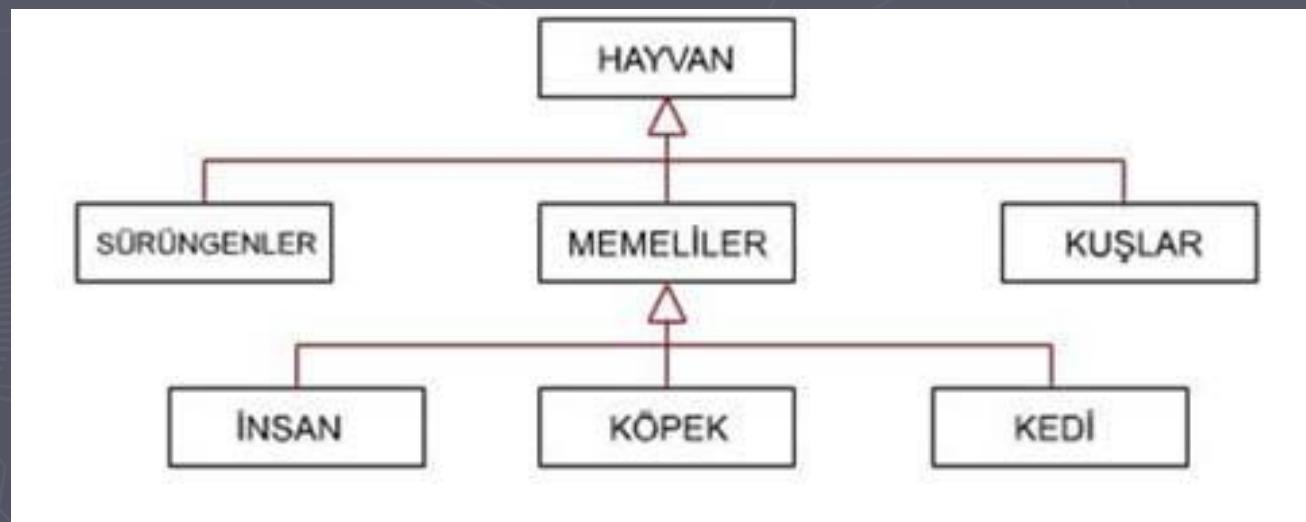
Operation Bölgesi

- ▶ Bu bölgede sınıf üyelerinden metodlar yer alır. Öncelikle erişim belirtice yazılır. Üyenin adından sonra varsa metod parametrelerinin türleri parantez içersinde yazılır. : (iki nokta üst üste) yazılır ve geri dönüş tipi belirtilir.

Deneme
+yas:integer -tarih:DateTime #isim:string
+YasHesapla(DateTime):integer -Topla(double,double):double #EkranaYazdir():void

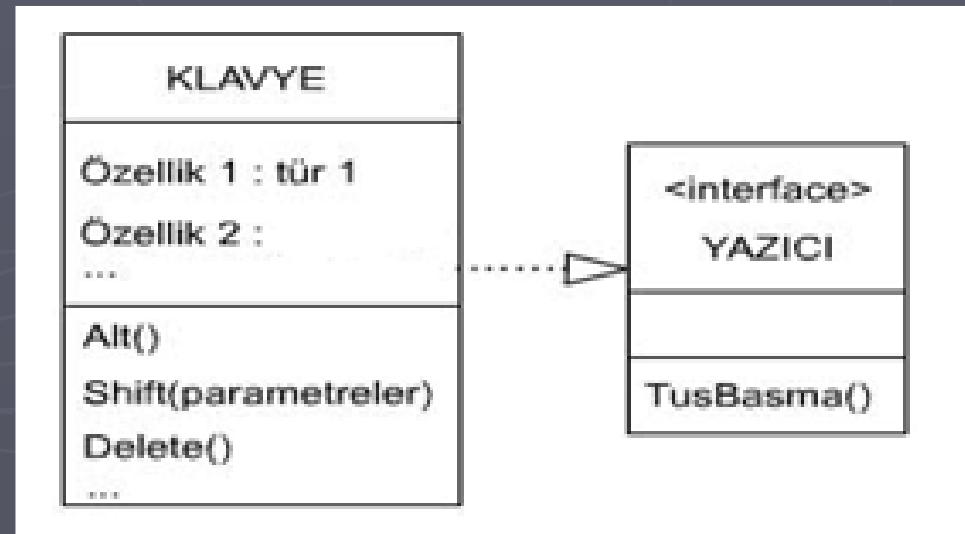
Sınıflar Arası İlişki 1: Kalıtım

- ▶ Sınıflar arası türetme işlemi ucu açık üçgen ve düz bir çizgiyle yapılır.
- ▶ Türetme sınıflar arası ilişki açısından türetmenin “is kind of”(bir çeşit) ilişkisinin olduğu görülür(Bird is a kind of Animal) gibi



Sınıflar Arası İlişki 2: Gerçekleme

- ▶ **Gerçekleme(Realization)** Bir sınıfın bir arayüze(interface) erişerek, arayüzün fonksiyonlarını gerçekleştirmesine denir.
- ▶ Kesik çizgilerle ve çizginin ucunda boş bir üçgen olacak şekilde gösterilir.



Sınıflar Arası İlişki 3: Association

Bağıntı İlişkisi (Association)

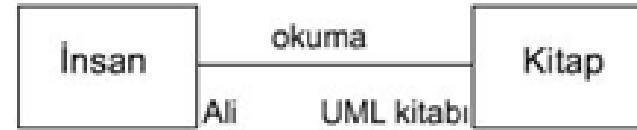
Bağıntı ilişkisi, sınıf diyagramlarında en çok kullanılan ve en basit ilişki türündür. Çoğu zaman referans tutma biçimindedir. İki nesne arasında varolan bağıntının çökluları (n:m), ilişki bağıntı sınıfı ile ifade edilebilir. Bağıntı ilişkisi iki nesne arasına çizilen düz çizgi ile belirtilir. Bağıntı ilişkileri için tanımlanmış bilgiler aşağıdaki gibidir;

- Bağıntının Adı
- Sınıfın bağıntıdaki rolü
- Bağıntının çöklüğü

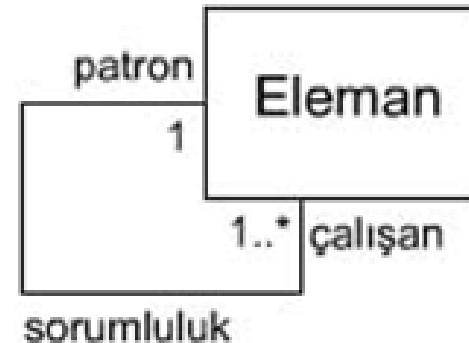
Bağıntı adı, iki sınıf arasındaki ilişkinin küçük bir açıklamasıdır. Bu açıklama ile birlikte yön bilgisi de belirtilebilir.

Sınıflar Arası İlişki 3: Association

- Bire-bir
- Bire-çok
- Bire –bir veya daha çok
- Bire –sıfır veya bir
- Bire-sınırlı aralık
- Bire-n (*)



Reflexive(Kendine dönen)ilişki: Bir sınıfın sistemde birden fazla rolü vardır.



Sınıflar Arası İlişki 3: Association



- Sınıf diyagramlarında sınıflar arasında bire-n ilişki kurulabilir. Bir sınıf, n tane başka bir sınıf ile ilişkiliyse buna bire-çok (1-n) ilişki denir.



- Sınıflar arasında yönlü bağlantıda olabilir. Normal ok ile gösterilir.

Sınıflar Arası İlişki 4: Dependency

Bağımlilik İlişkisi (Dependency)

- ▶ Birliktelik nesneler arası uzun süreli ilişkidir.
- ▶ • Gerçek hayatı , örneğin , insanlar ve arabaları bir ilişki oluştururlar. Bu ilişki bir birlikteliktir, bir yerden başka bir yere gitme olayında , ne kullanıcı arabasız düşünülebilir nede araba kullanıcısız düşünülebilir.

İki tür birliktelik vardır:

- ▶ 1. İçerme (Aggregation)
- ▶ 2. Kompozisyon - Oluşum (Composition)

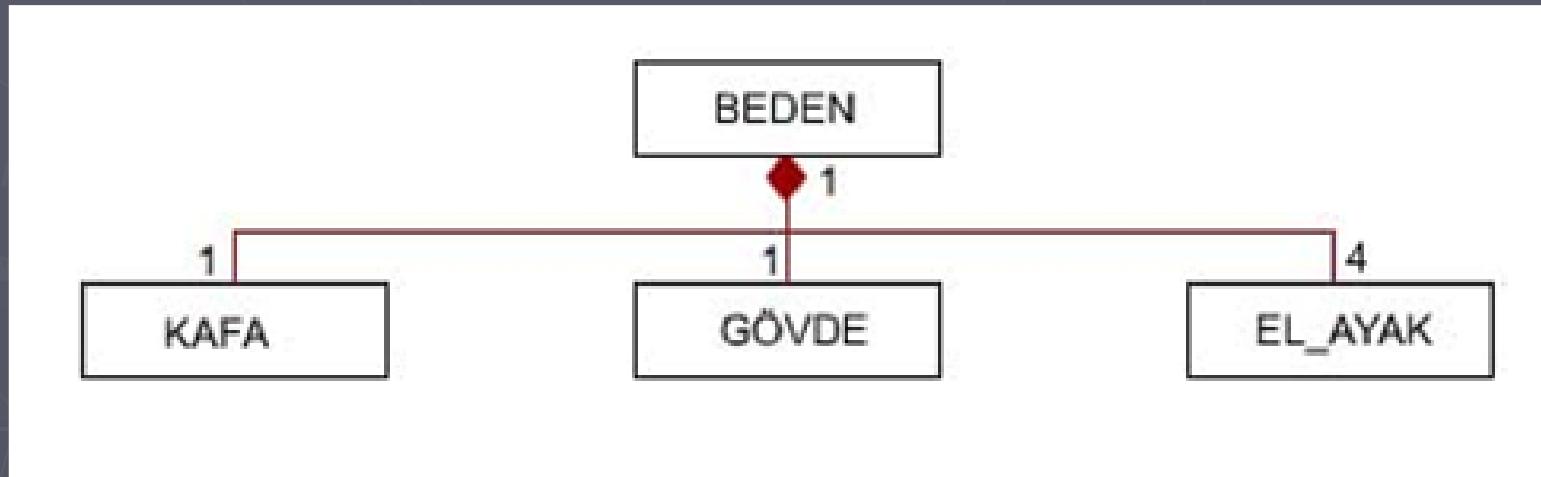
Sınıflar Arası İlişki 4.1: Aggregation

- ▶ Bütün parça yukarıda olacak şekilde ve bütün parçanın ucuna içi boş elmas yerleştirilecek şekilde gösteririz.
- ▶ İçi boş elmas ile gösterilen ilişkilerde herbir parça ayrı bir sınıfıtır ve tekbaşlarına anlam ifade eder. Parça bütün arasında sıkı bir ilişki yoktur.



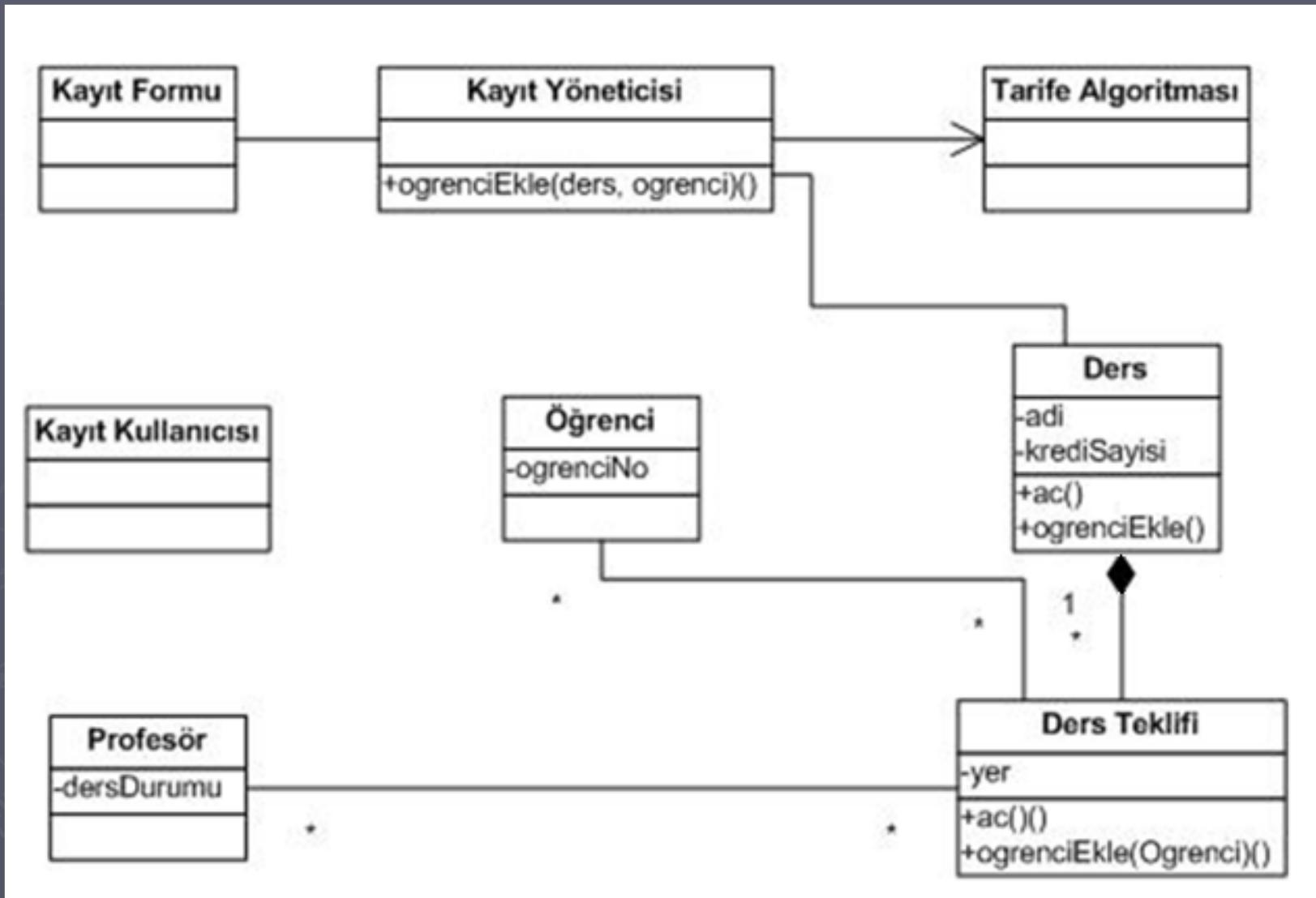
Sınıflar Arası İlişki 4: Composition

- ▶ Bazı durumlarda bütün nesne yaratıldığında parçalarının da yaratılmasını isteriz.
- ▶ Bu ilişki daha sıkıdır. İçi dolu elmas ile gösterilir.



- ▶ Beden olmadan gövdenin tek başına bulunmasının anlamı yoktur.

Örnek



UML Use Case Diyagramları

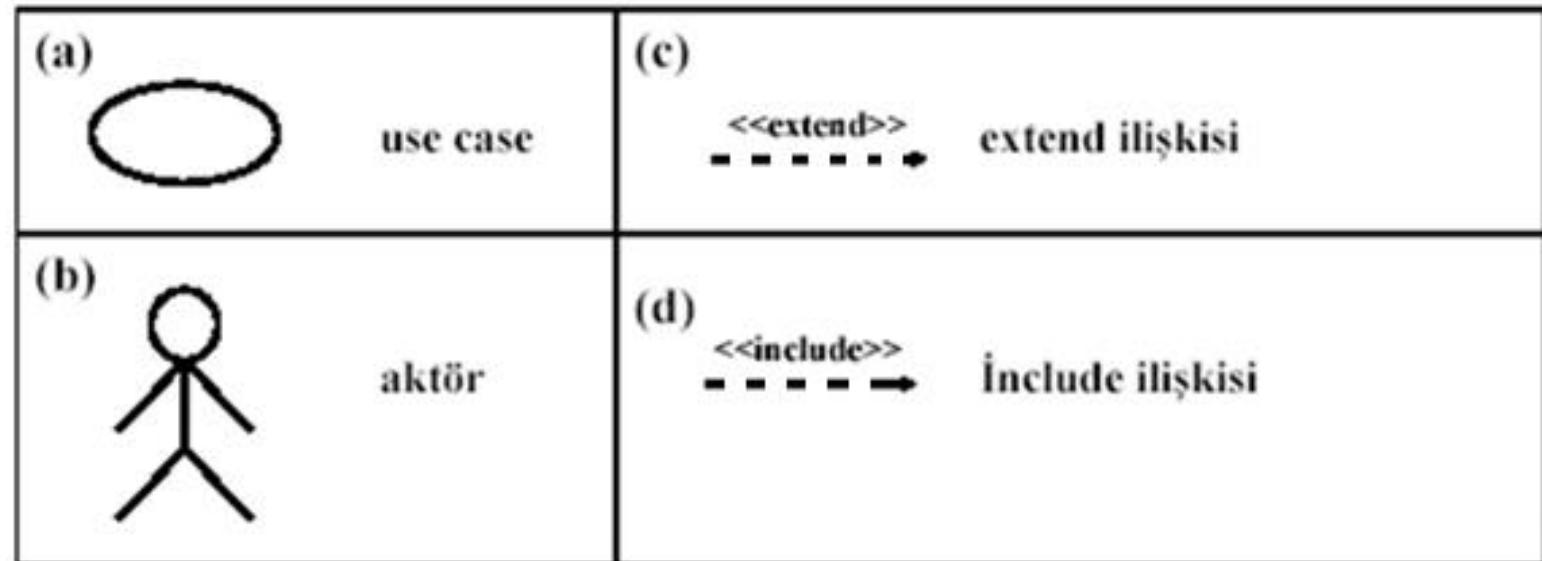
- ▶ **Kullanım Senaryosu (Use Case)** sistemin kullanıcılarla sunacağı bir hizmetin senaryo şeklinde anlatımıdır. Sistem gereksinimleri Use Case diyagramları ile belirtilir.
- ▶ Yazılım geliştirme için gerekli değildir, fakat gereksinimler ve nesnesel modeller arasında en önemli bağlantıdır.
- ▶ Bu diyagramlar ilk olarak aktörlere bakılarak oluşturulurlar. Aktör sistemin sunduğu hizmetleri kullanan bir kişi veya başka bir sistemdir. Aktörler sistemin dışında olan ve sistemle etkileşimde bulunması olası bir şahıs veya farklı bir sistem olarak belirtilirler. İlk olarak sorulacak soru sistemle kim iletişimde bulunacak sorusudur.

UML Use Case Diyagramları

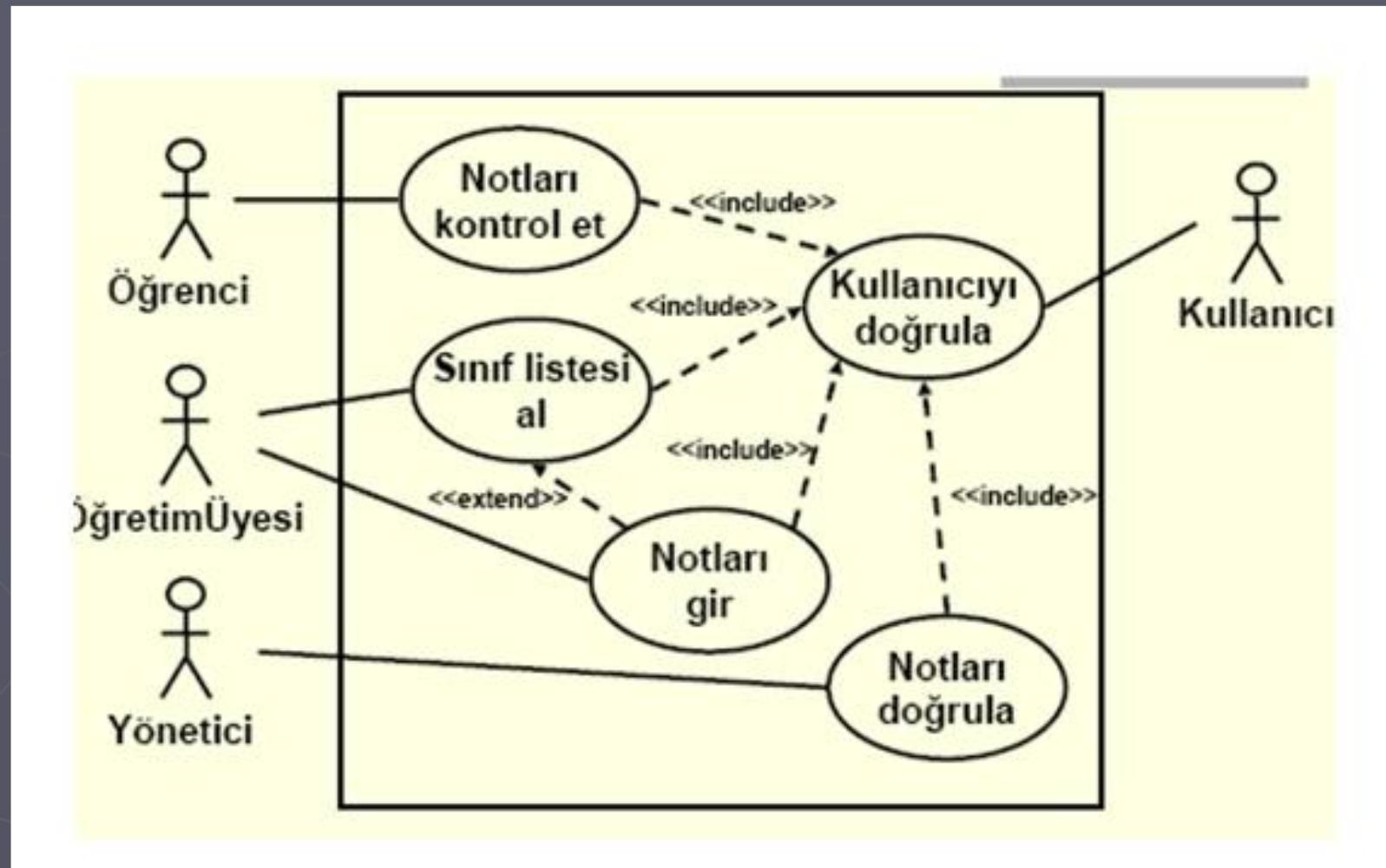
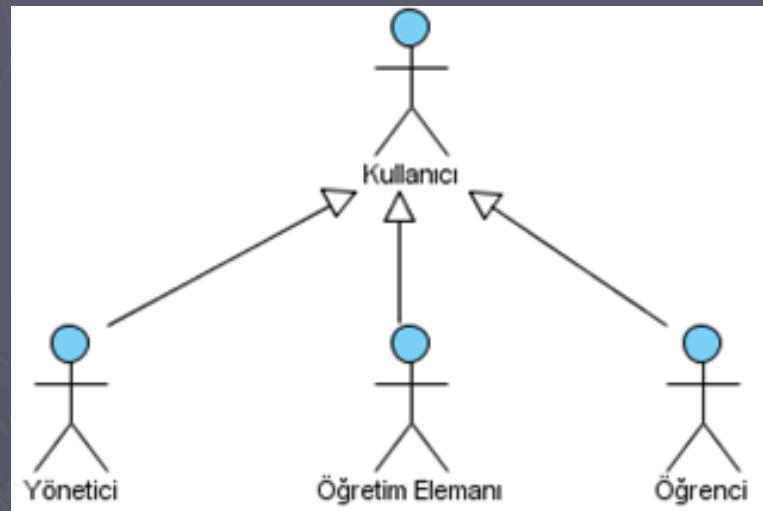
- ▶ Diyagramı hazırlayanlar Sistem Analisti'dir.
- ▶ Diyagramı kullananlar ise Müşteri, Proje Yöneticisi, Tasarımcı, Veritabanı analisti, tasarımcı
 - ❖ Ders seçim modeli için aktör olarak şu hususlardan bahsedebiliriz. Kayıt memuru, öğrenci, profesör ve de dış bir ödeme sistemini ele alabiliriz. (aktör illaki bir insan olmak zorunda değildir farklı bir sistem de aktör olabilir)
 - ❖ Kullanıcı durumu (use case) sistemdeki aktör tarafından sistem ile bir etkileşim esnasında gerçekleştirilecek işlemler dizisinin referansıdır. Kullanım durumları birer yazılım modülü degillerdir. Sistem ile etkileşimde bulunacak aktörlerin sistem için önemini gösterirler.
 - ❖ Sistemdeki kullanım durumlarını bulmak için o sisteme kısaca aktörlerin sistemi ne amaçla kullanmak istediklerini sormak yeterli olacaktır.

UML Use Case Diyagramları

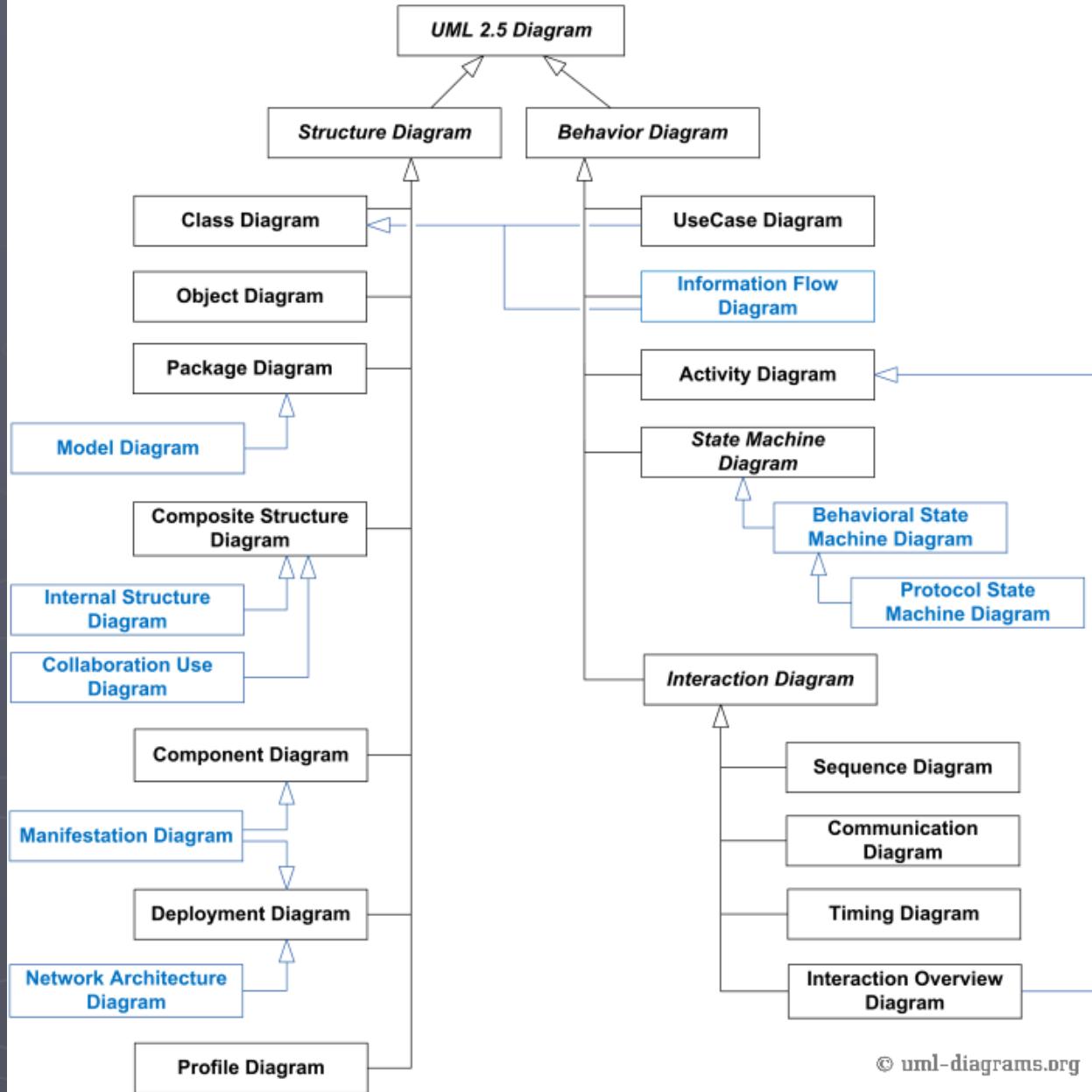
Sembol	Açıklama
use case	Belli bir hedefe ulaşmak için kullanılabilen tüm alternatif senaryolar
aktör	Sistemle etkileşen kişilerin canlandırdıkları roller ve dış sistemler
extend	Bir use case'in kullanıcısının seçimine bağlı olarak çalıştığı diğer bir use case'i vurgular
include	Bir use case'nin her zaman çalıştığı diğer bir use case'i vurgular



UML Use Case Diyagramları



Ek 1: 2.5 Standartına göre UML Diyagramları



Ek 2: UML Çizim Programları

- ▶ LucidChart : <https://www.lucidchart.com/> (browser bazlı)
- ▶ Microsoft Visio, Visual Studio 2015 Enterprise
- ▶ Visual Pradigm
- ▶ starUML: <http://staruml.io/download>