

NESNEYE YÖNELİK PROGRAMLAMA

Yrd.Doç.Dr. Zeynep ORMAN
ormanz@istanbul.edu.tr

Kullanım Diyagramları (Use Case Diagram)

- Kullanım senaryoları sadece **düz metin** (*text*) olarak değil, istendiğinde metin yerine UML diyagramı olarak da ifade edilebilirler.
- Kullanım diyagamlarında, kullanım senaryolarının aktörler ile ve kendi aralarındaki ilişkileri grafik olarak gösterilir.
- Bir sistemin içinde bir çok senaryo grubu bulunabilmekte ve değişik aktörler değişik senaryo grupları ile ilişkili olabilmektedir.
- Ayrıca senaryoların kendi aralarında da **icerme** (*include*) ve **genişletme** (*extend*) ilişkileri bulunabilmektedir.

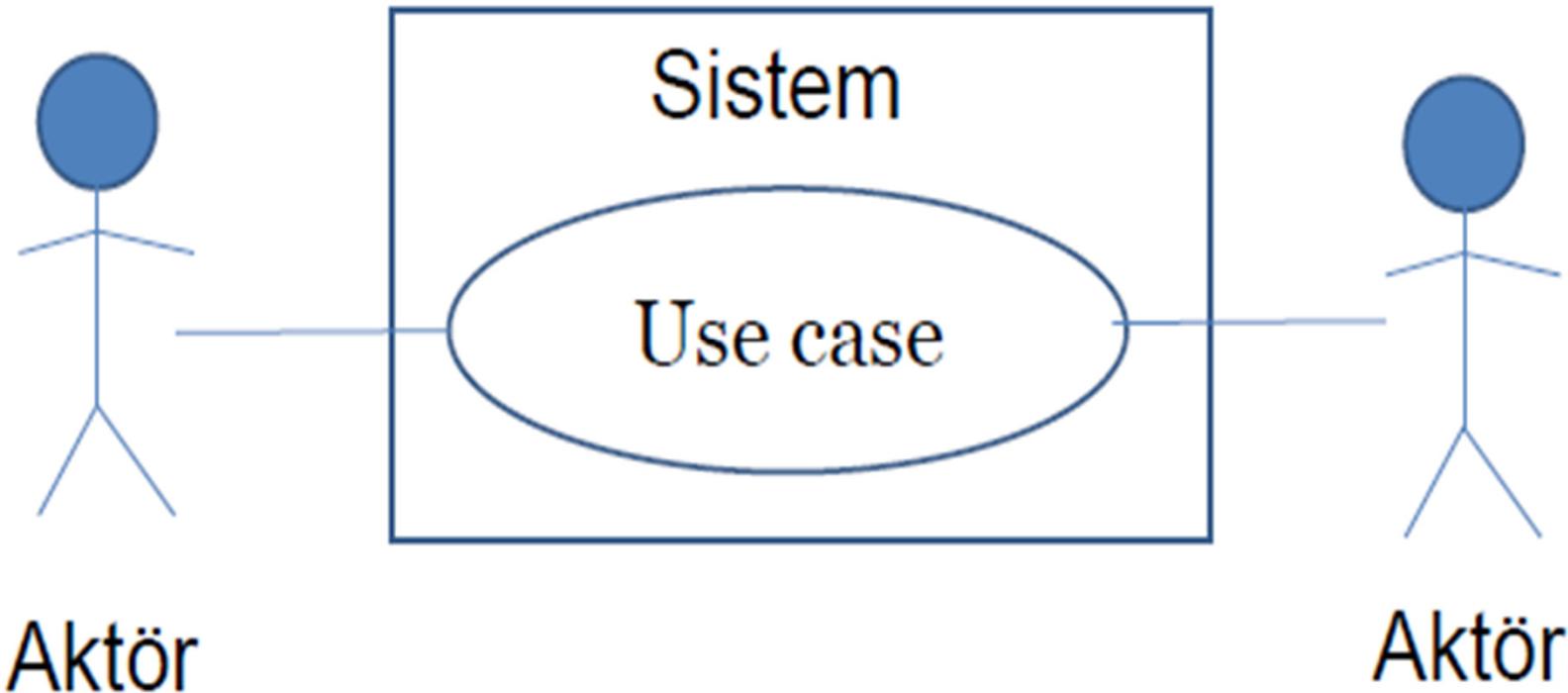
Kullanım Durum Diyagramları

- Kullanım Durum Diyagramları, 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.

Kullanım Durum Diyagramları

- 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)
- Sistemdeki kullanım durumlarını bulmak için o sisteme kısaca aktörlerin sistemi ne amaçla kullanmak istediklerini sormak yeterli olacaktır.

Basit bir Kullanım Durum Diyagramı yapısı



KULLANIM DURUM DİYAGRAMLARI BİLEŞENLERİ

Aktör

- Sistemin kullanıcılarıdır.
- Aktörler genelde belirli bir rol ifade ederler.
- Sistem sınırları dışında gösterilir.
- Aktörler arasında doğrudan (genelleme ilişkisi dışında) ilişki tanımlamak mümkün değildir.
- Arada mutlaka kullanım durumları olmalıdır



KULLANIM DURUM DİYAGRAMLARI BİLEŞENLERİ

- Kullanım Durumu
- Sistemde yer alan tüm alternatif senaryolardır.
- Sistem fonksiyonellüğünün büyük bir parçasını gösterir.
- Diğer bir use case ile genişletilebilir. (extend)
- Diğer bir use case içerebilir. (include)
- Sistem sınırları içinde gösterilir.



Use case

KULLANIM DURUM DİYAGRAMLARI BİLEŞENLERİ

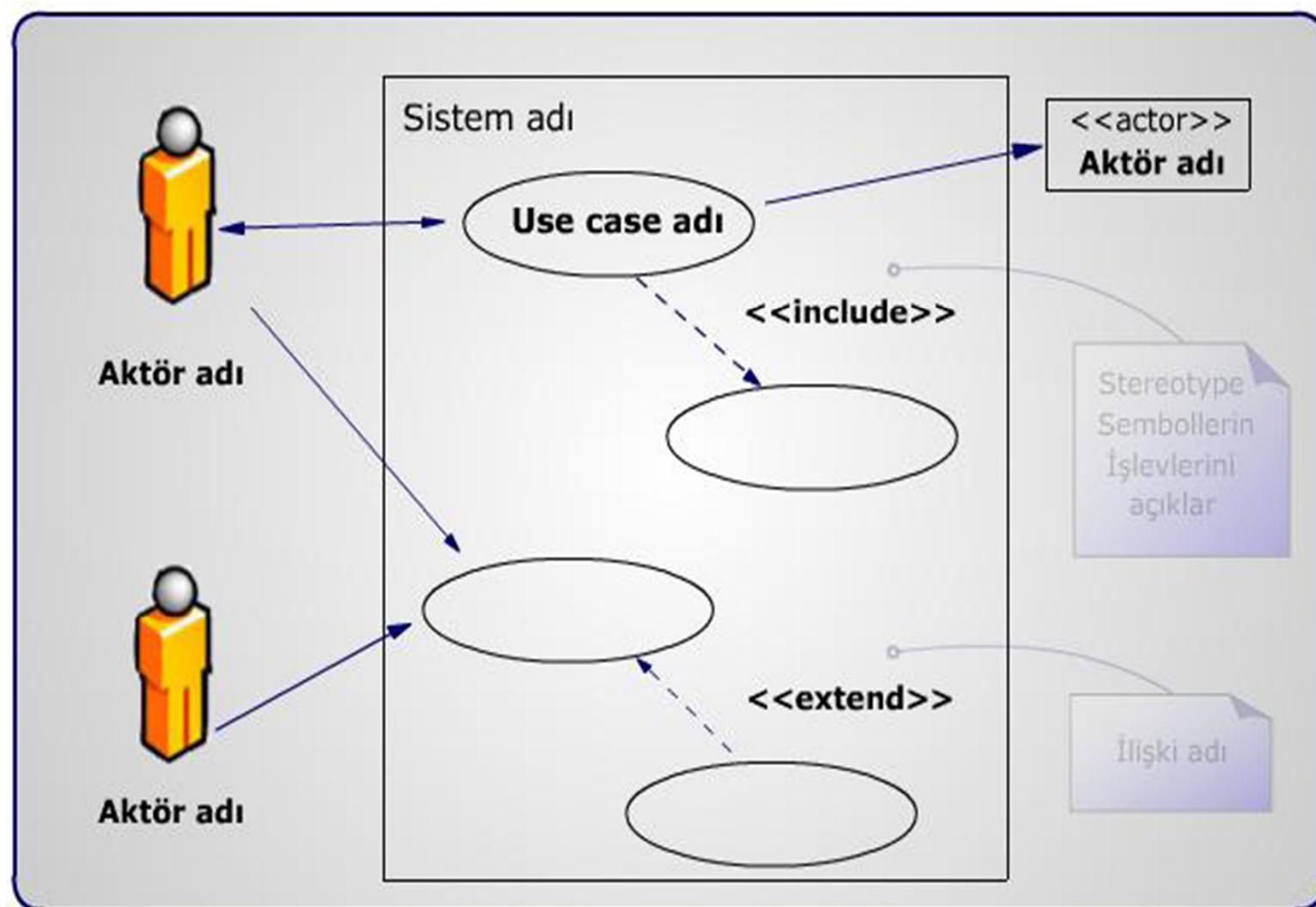
1. İçerme (*includes, uses*):

Birçok senaryo grubunda kullanılan başka bir senaryo grubudur. Örneğin otomasyon sistemini kullanmak için giriş yapılması gereklidir. Bir senaryonun içinden bir alt programa dalandırıp geri dönmek gibidir.

2. Genişletme (*extends*):

Senaryo grupları doğal akışa göre hazırlanır. Çeşitli koşullar altında bu doğal akıştan sapmalar olabilir. Genişletme ilişkisi ana senaryodan ayrılma noktasından sonra yapılanları belirtir.

KULLANIM DURUM DİYAGRAMLARI



İçerme (include) ilişkisi

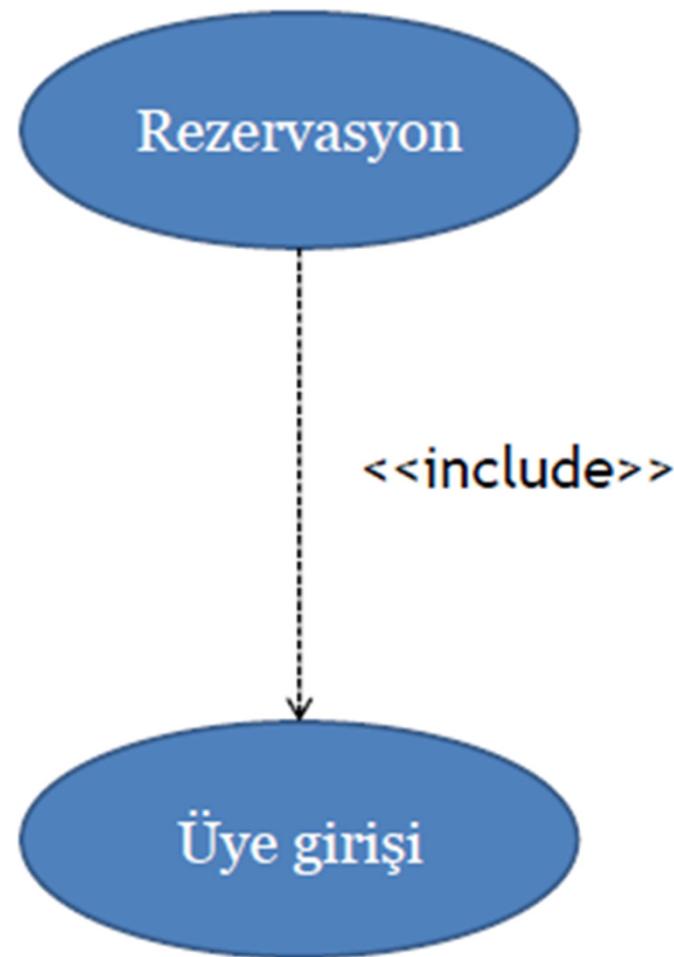
- İçerme ilişkisini kullanmak için <<include>> şeklindeki bir ifade kullanılır.
- Kullanmak istediğimiz kullanım durumları arasında çektiğimiz noktalı çizginin üzerine <<include>> yazısını yazarız.

<<include>>



İçerme (include) ilişkisi

- Örnek:



Genişletme (extend) İlişkisi

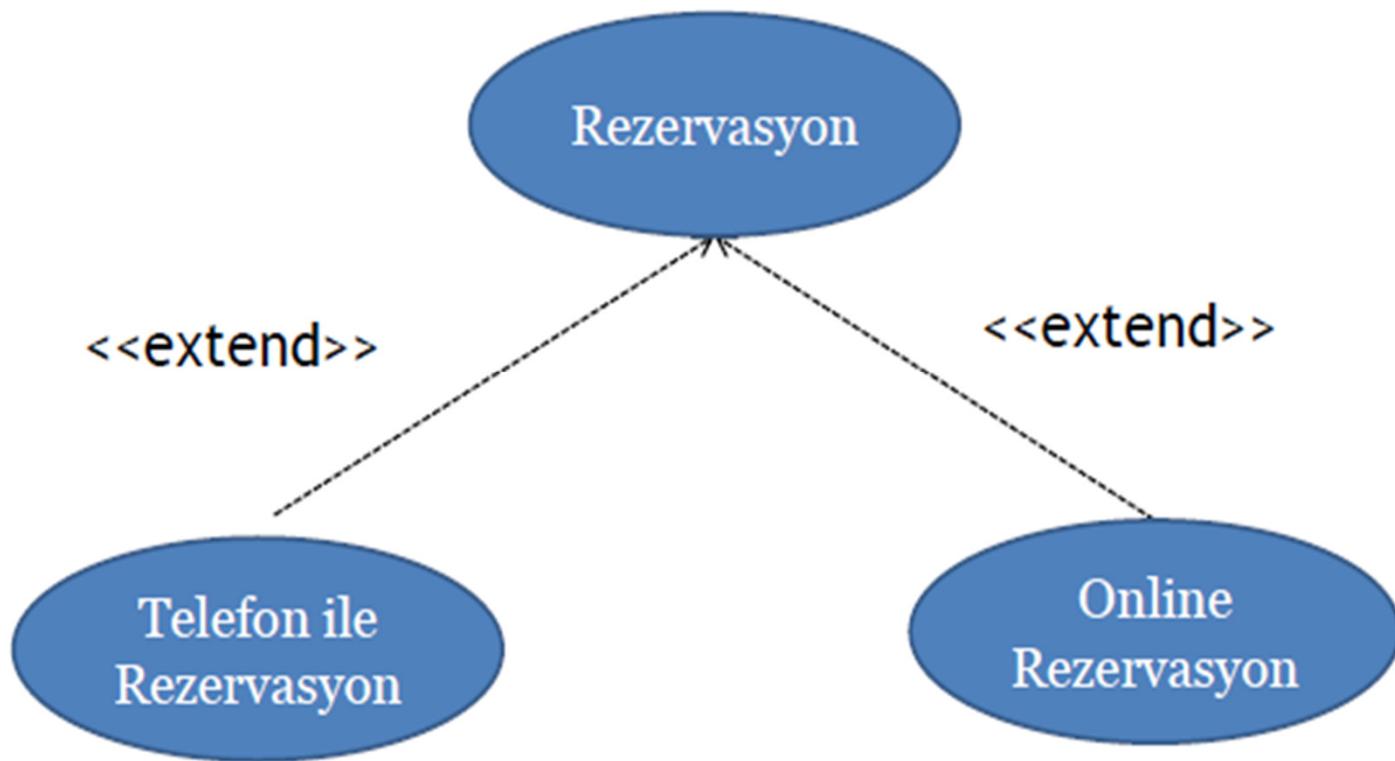
- Include'da olduğu gibi genişletme ilişkisini göstermek için yine kullanım durumları arasında noktalı çizgiler konur ve üzerine <<extend>> ibaresi yazılır.

<<extend>>

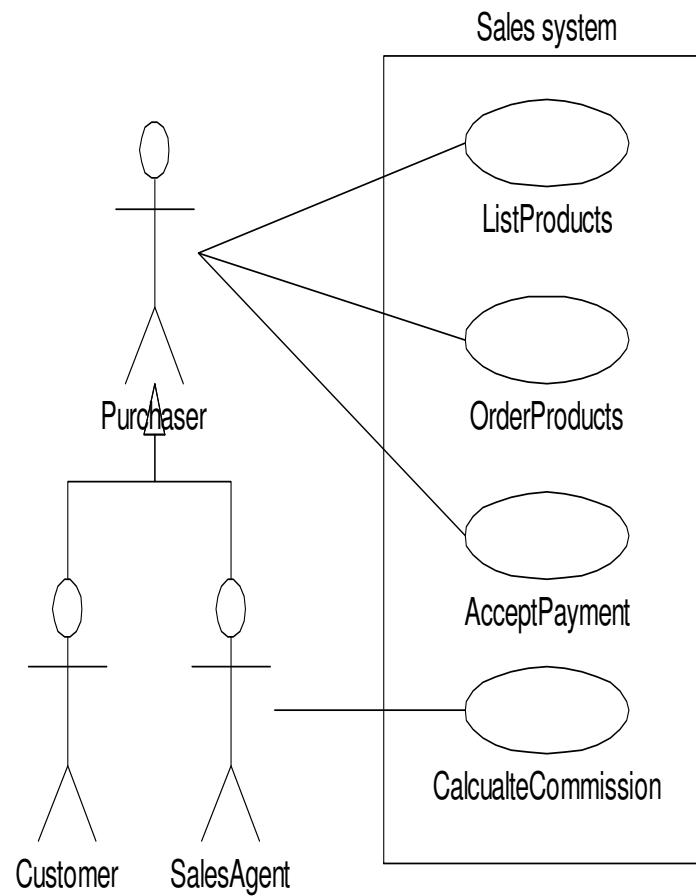
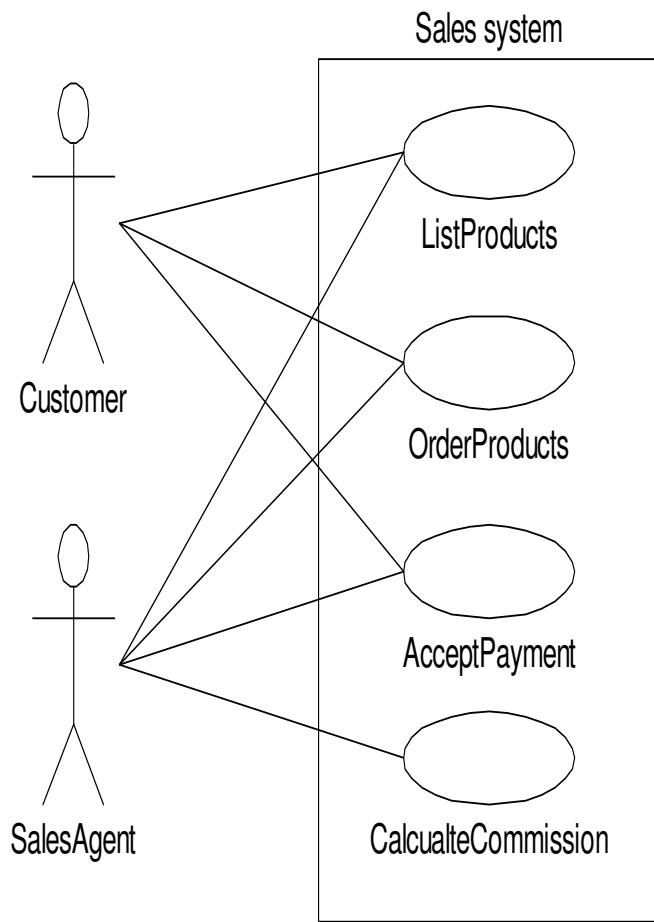


Genişletme (extend) İlişkisi

Örnek:

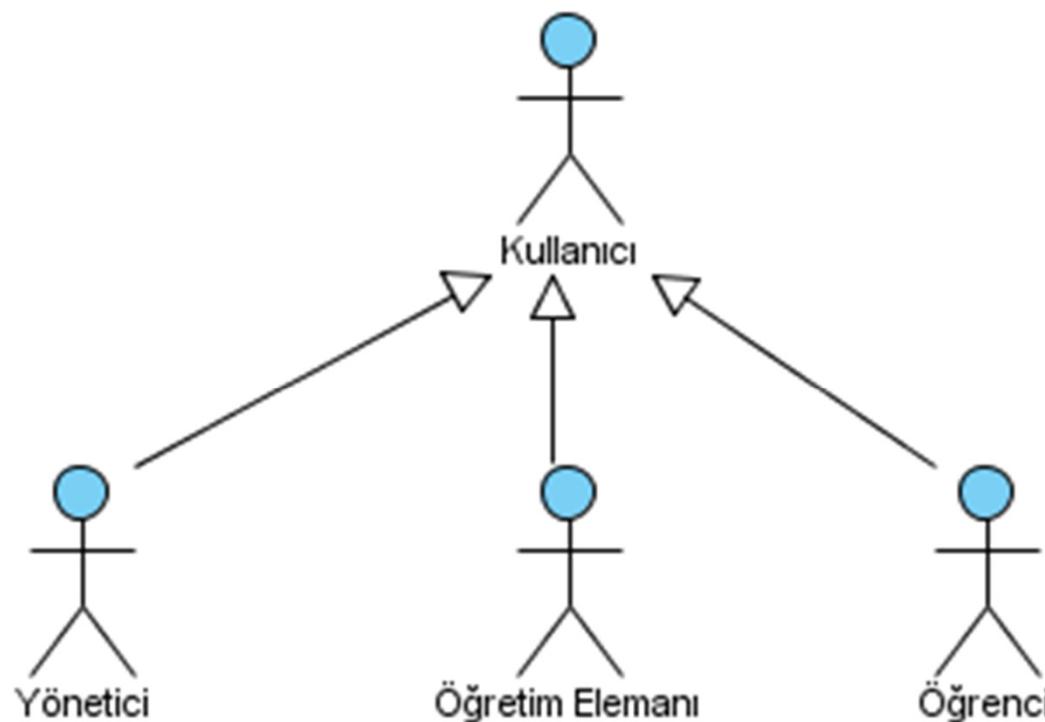


Genelleme İlişkisi

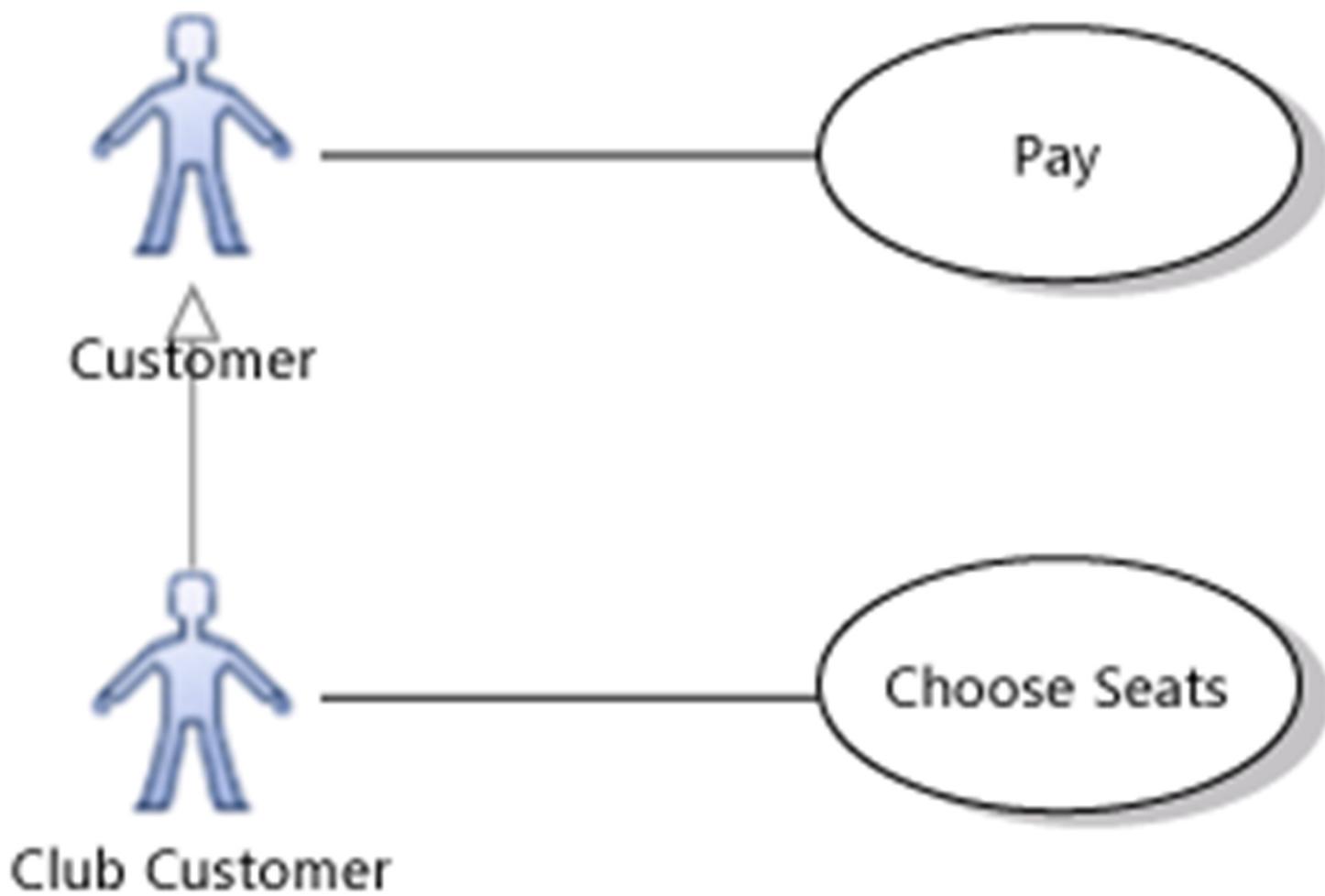


Genelleme İlişkisi

- İki kullanım durumu veya iki aktör arasındaki kalıtım ilişkisidir.



Genelleme İlişkisi



Bir örnek: ATM uygulaması

- Bir bankanın ATM cihazı için yazılım geliştirilecektir.
- ATM, banka kartı olan müşterilerin hesaplarından para çekmelerine, hesaplarına para yatırmalarına ve hesapları arasında para transferi yapmalarına olanak sağlayacaktır.
- ATM, banka müşterisi ve hesapları ile ilgili bilgileri, gerektiğinde merkezi banka sisteminden alacaktır.



Bir örnek: ATM uygulaması

- ATM uygulama yazılımının kullanıcıları:

- Banka müşterisi
 - Merkezi Banka Sistemi
- 
- Aktörler

Bir örnek: ATM uygulaması

- Belirlenen aktörler ATM'den ne istiyorlar?
- **Aktör:** Banka müşterisi
 - ✓ Para çekme
 - ✓ Para yatırma
 - ✓ Para transferi
- **Aktör:** Merkezi Banka Sistemi
 - ✓ Günlük özet alma

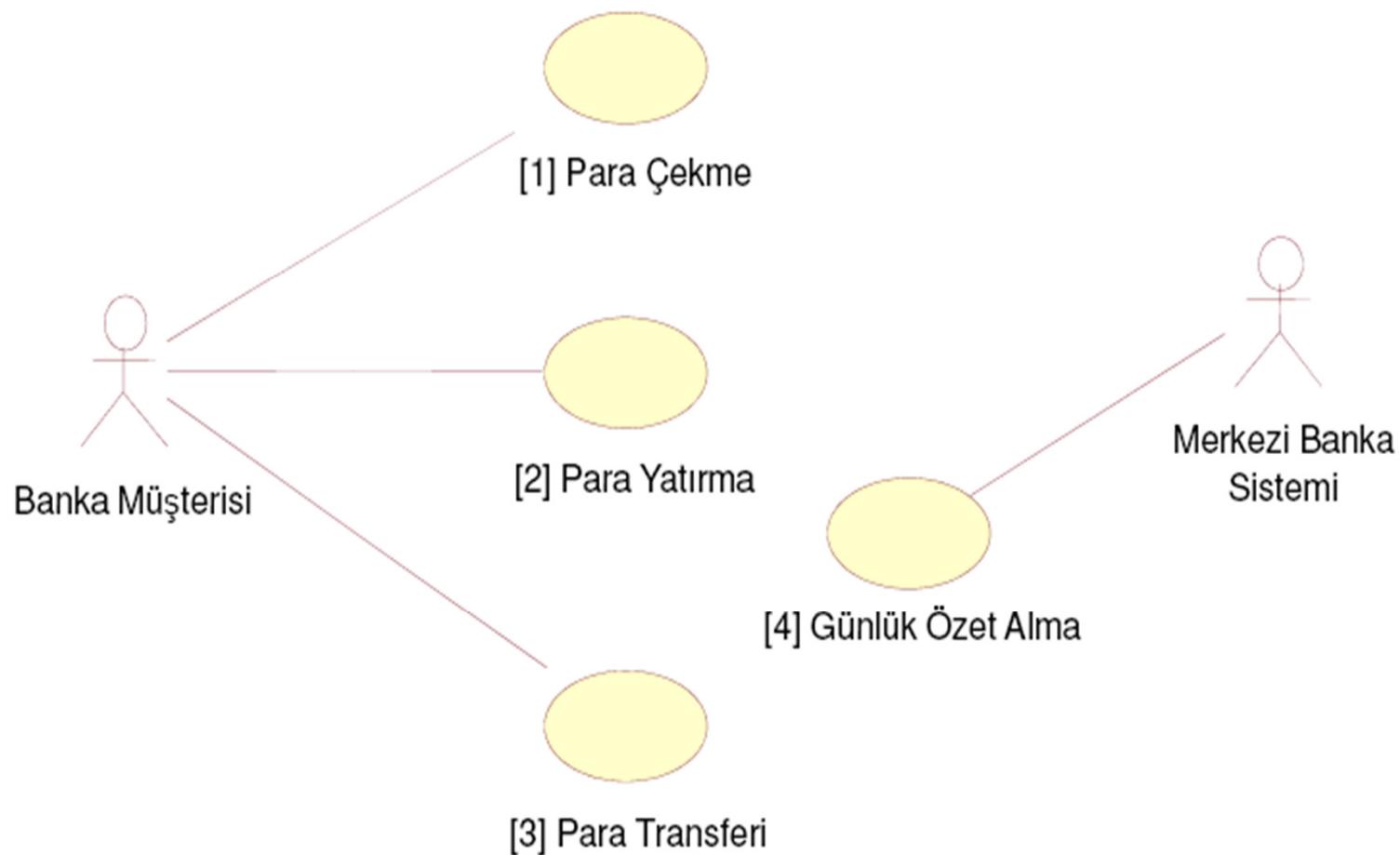
Bir örnek: ATM uygulaması

- Aktör: Banka müşterisi
- Bankada hesabı ve banka kartı olan, ATM'den işlem yapma hakkı olan kişidir.

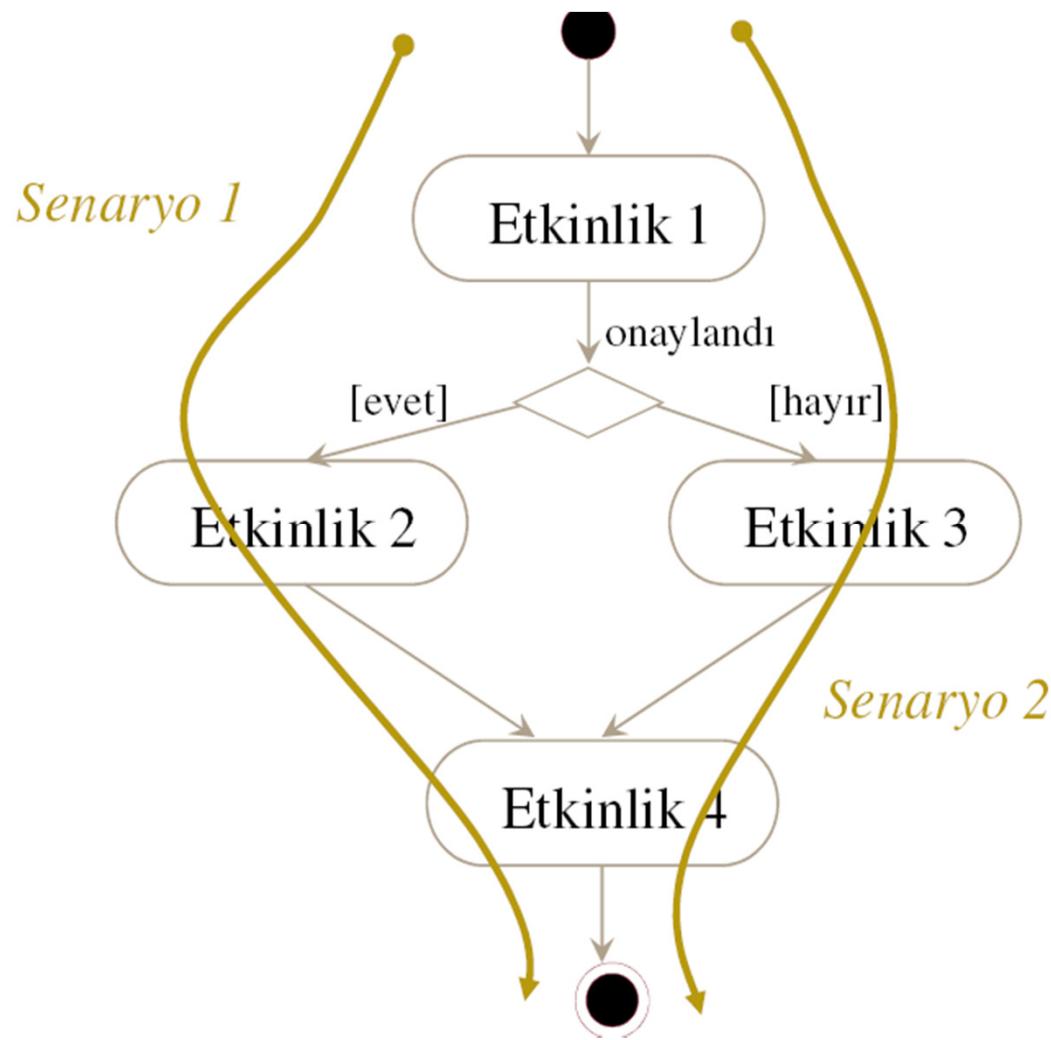
Kullanım Durumu: Para çekme

- Banka müşterisinin nasıl para çekeceğini tanımlar. Para çekme işlemi sırasında banka müşterisinin istediği tutarı belirtmesi ve hesabında bu tutarın mevcut olması gereklidir.

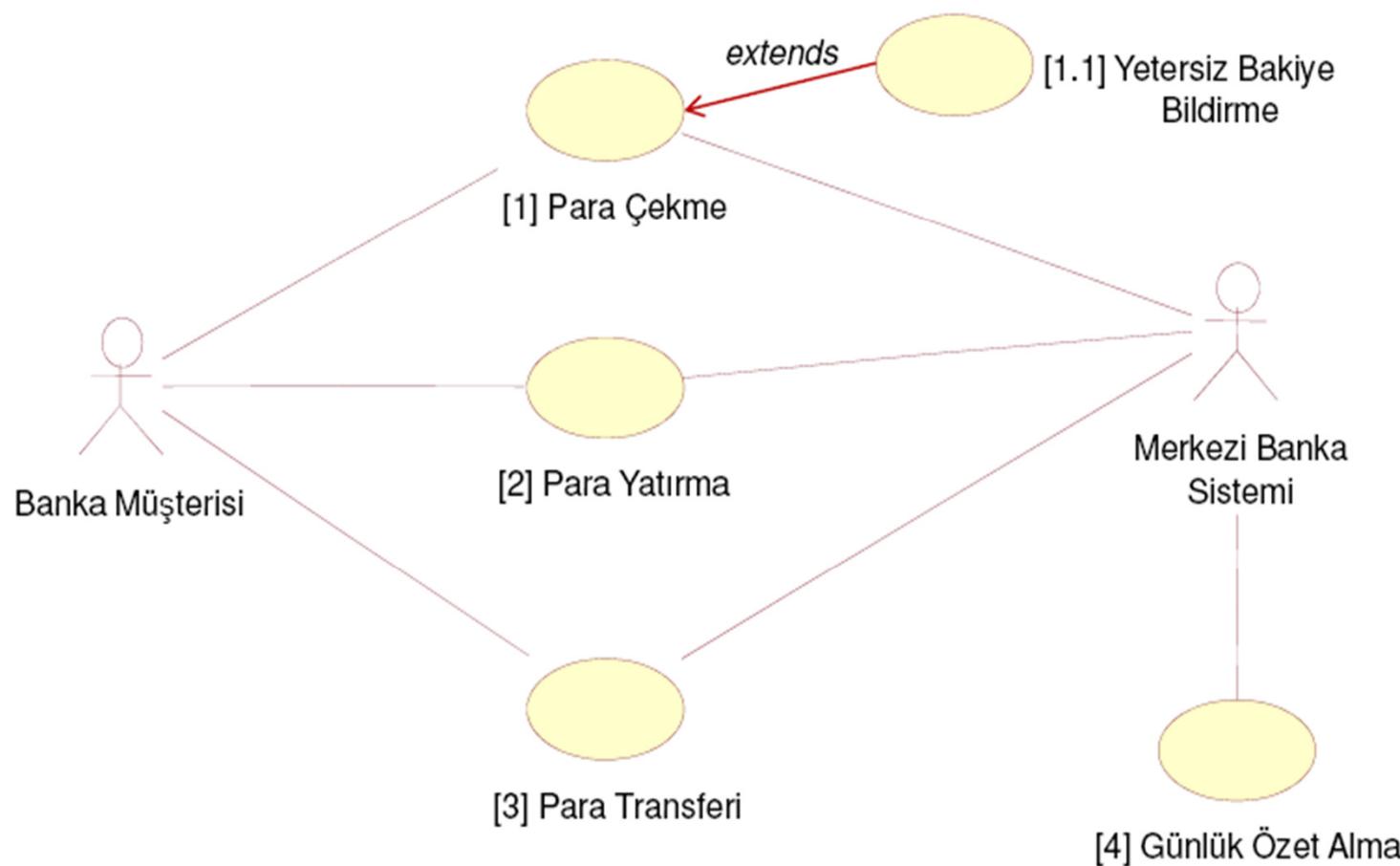
Bir örnek: ATM uygulaması



Bir örnek: ATM uygulaması



Bir örnek: ATM uygulaması

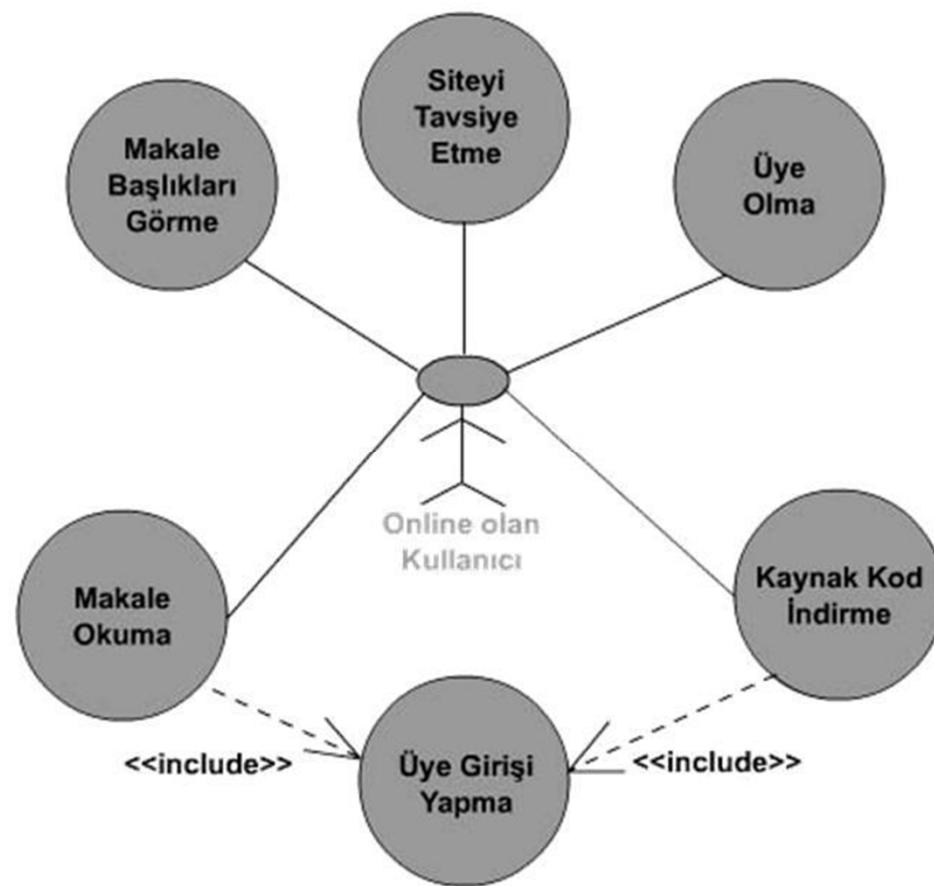


Bir use case diyagramı örneği

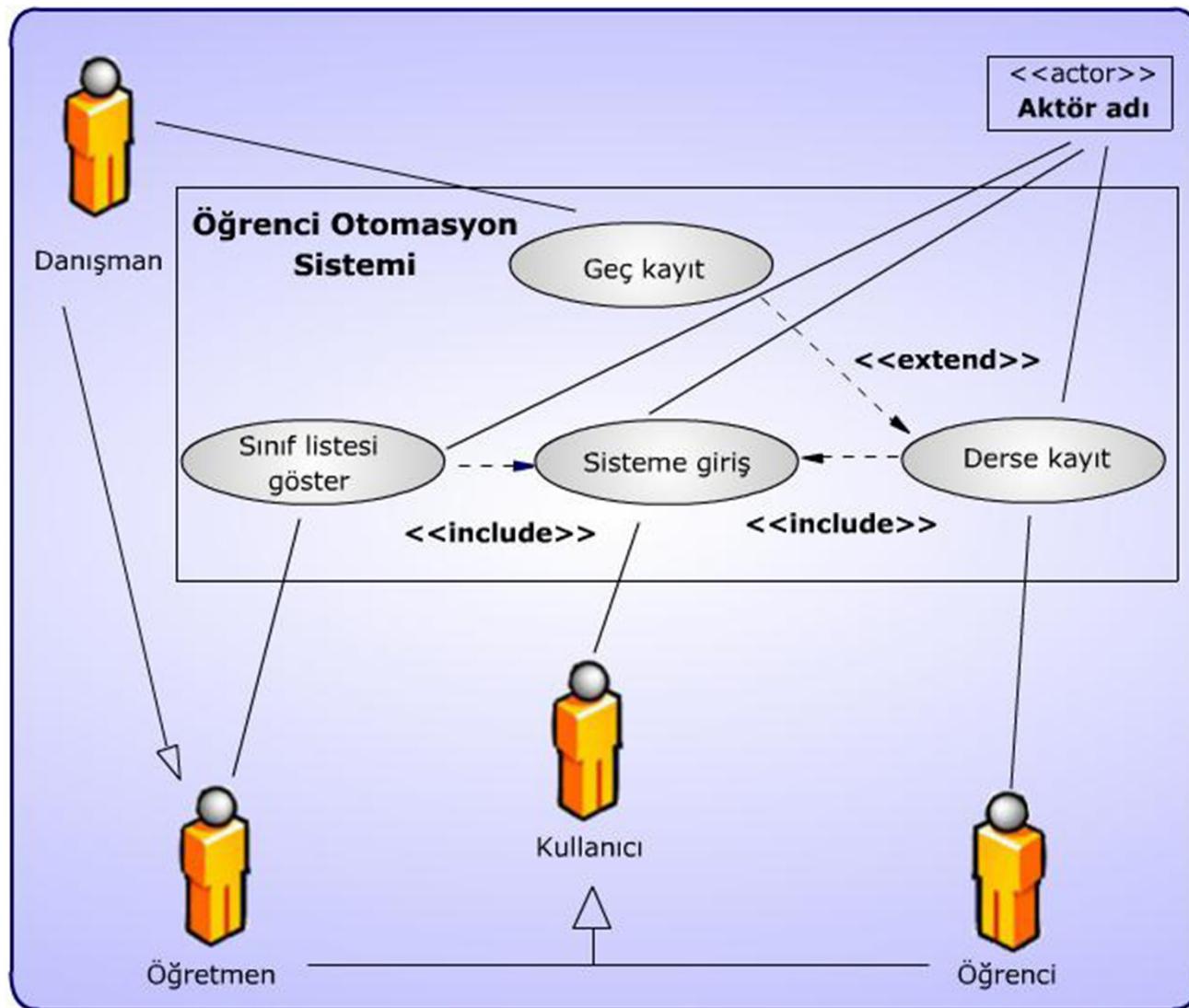
- Bir web sayfasına gelen bir kullanıcının neler yapabileceğini kullanım durum diyagramlarıyla göstermeye çalışalım.
- Siteye gelen bir kullanıcı kayıtsız şartsız makale başlıklarını görebilmektedir.
- Online olan kullanıcı siteyi tavsiye edebilir, siteye üye olabilir, kitapları inceleyebilir.
- Ancak makale okuması ve kaynak kod indirebilmesi için siteye üye girişi yapmalıdır.
- Makale okuması ve kaynak kod indirebilmesi için gereken şart siteye üye olmaktadır.

Kullanım Durum Diyagramı Örneği

- Siteye bağlanan bir kullanıcının site üzerindeki hareketlerini belirtir diyagram bu şekilde oluşturulabilir.



Kullanım Durum Diyagramı Örneği



UML Sınıf Diyagramları (Class Diagrams)

- Bir **sınıf**, ortak yapısı, ortak davranışları, ortak ilişkileri ve ortak anlamı bulunan nesneler koleksiyonudur.
 - Örneğin, Volkswagen, Toyota ve Ford ortak özellikleri olan ve ortak davranışlar gösteren birer arabadır.
 - Öyleyse bunların hepsini “Araba” sınıfıyla ifade edebilirim.
- **Nesne**, belirli bir sınıfa ait olgudur(instance).
 - Örneğin Araba sınıfının bir nesnesi Toyota olabilir.

UML Sınıf Diyagramları (Class Diagrams)

- **Sınıf diyagramı:** Bir sistemin yapısını; sistemdeki sınıfları, sınıfların niteliklerini ve sınıflar arasındaki ilişkileri göstererek ifade eden diyagramdır.
 - Sistemi oluşturan sınıflar ve bunlar arasındaki ilişkileri gösterir.
 - Sistemin statik yapısını ifade eder.
 - Yol Haritası gibi
 - Nesneler şehirleri, ilişkiler şehirler arasındaki yolları gösterir
 - Hedefe ulaşmak için hangi yolu takip edilmesi gerektiğini söylemez

UML Sınıf Diyagramları

- UML'deki en temel diyagram tiplerinden biridir.
- Sınıf diyagramları, nesneye-yönelik modellemenin yapıtaşıdır.
- Sınıf diyagramı, sistem için tanımlanan tüm sınıfları içermeyebilir.
- Bir sistemi modellemek için birden fazla sınıf diyagramı kullanılabilir.
- Bir sistemle ilgili çizilen bir sınıf diyagramı sistemin belirli bir görünümünü ifade ederken, çizilen bütün sınıf diyagramları birlikte bütün sistemi gösterir.

UML Sınıf Diyagramları

- 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 standardizasyon olması amacıyla bütün isimler büyük harf ile başlarlar.

UML Sınıf Diyagramları

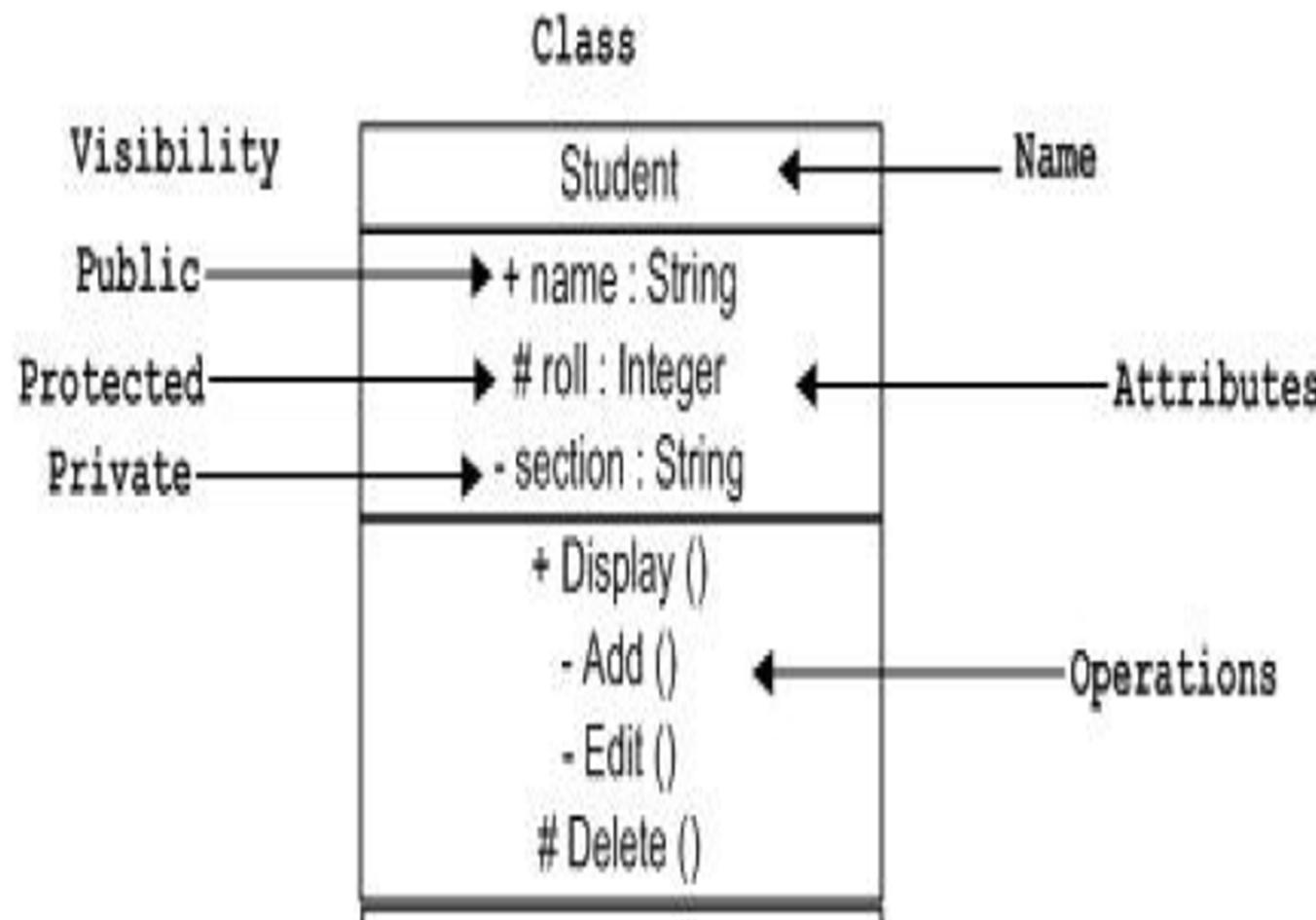
Sınıf Adı

Özellik 1:Tür1
Özellik2:yaş=19
.....

İşlev1()
İşlev2(parametreler)
İşlev 3():geri dönen değer tipi

- **Sınıf Adı**
- **Özellikler(properties)**
- **Metodlar(functions)**

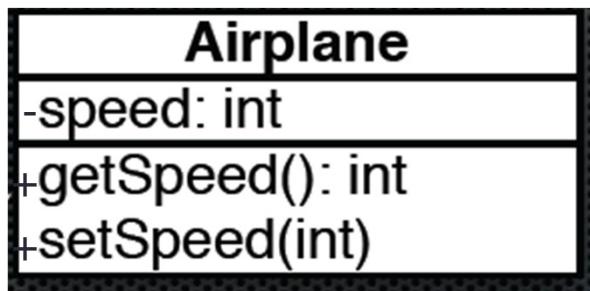
Görünürlük (Visibility)



Görünürlük (Visibility)

- **Public:**diğer sınıflar erişebilir. UML'de **+** simgesi ile gösterilir.
- **Private:**yalnızca içinde bulunduğu sınıf tarafından erişilebilir (diğer sınıflar erişemezler). UML'de **-** simgesi ile gösterilir.
- **Protected:**aynı paketteki (*package*) diğer sınıflar ve bütün alt sınıflar (*subclasses*) tarafından erişilebilir. UML'de **#** simgesi ile gösterilir.
- **Package:**aynı paketteki (*package*) diğer sınıflar tarafından erişilebilir. UML'de **~** simgesi ile gösterilir.

UML Sınıf Diyagramı Kod Arası İlişki



```
1 public class Airplane {  
2  
3     private int speed;  
4  
5     public Airplane(int speed) {  
6         this.speed = speed;  
7     }  
8  
9     public int getSpeed() {  
10        return speed;  
11    }  
12  
13    public void setSpeed(int speed) {  
14        this.speed = speed;  
15    }  
16  
17 }
```

UML Sınıf Tanımlamaları

- **Alanlar:**

Kod → private long maas

UML → - maas:long

- **Metotlar:**

Kod → public double maasHesapla(int,double,int)

UML → + maasHesapla(int,double,int):double

UML'de Nesne Gösterimi

nesneAdı : SınıfAdı

alan₁= değer₁

alan₂= değer₂

.

.

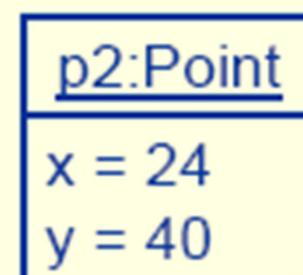
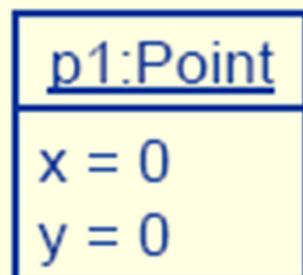
alan_n= değer_n

← Nesne ve Sınıf Adı;
altı çizili

← Alanlar ve aldıkları değerler

UML'de Nesne Gösterimi

UML



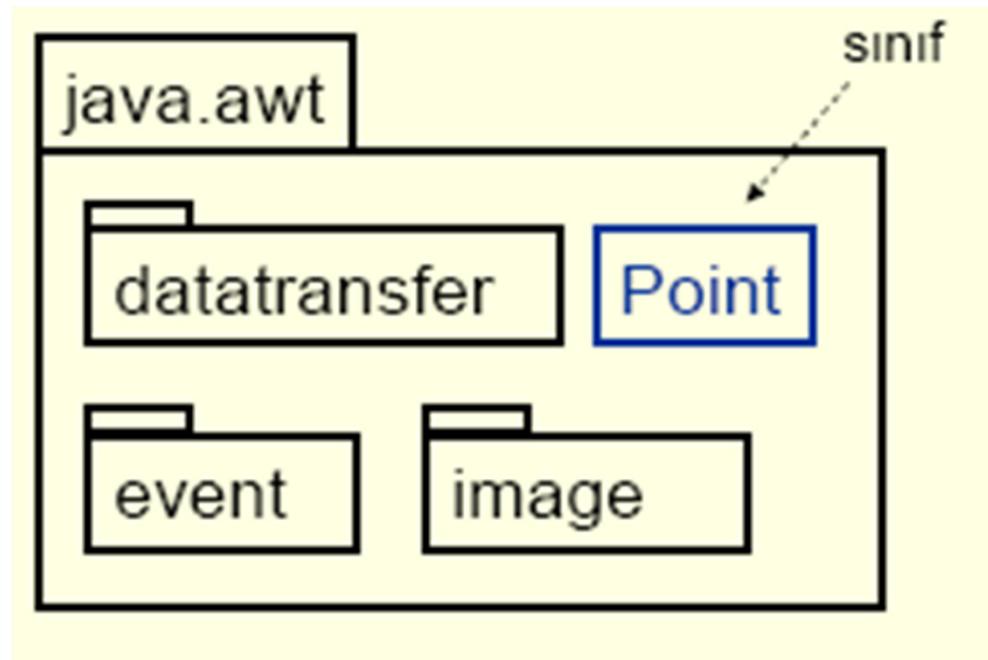
Java

```
Point p1 = new Point();
p1.x = 0;
p1.y = 0;
```

```
Point p1 = new Point();
p1.x = 24;
p1.y = 40;
```

UML'de Paket Gösterimi

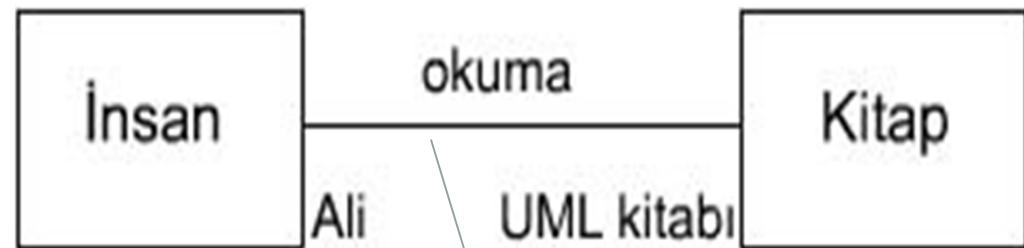
- Birbirleriyle ilişkili sınıflar bir paket (package) içine yerleştirilirler.
- Paket isimler küçük harflerle yazılır



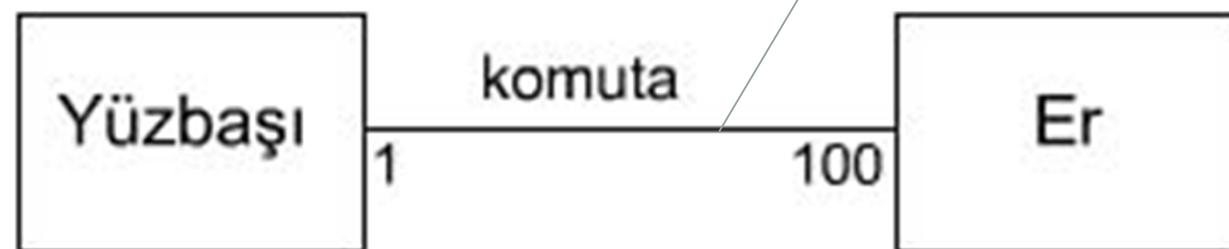
Sınıflar Arası İlişkiler

(multiplicity)

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



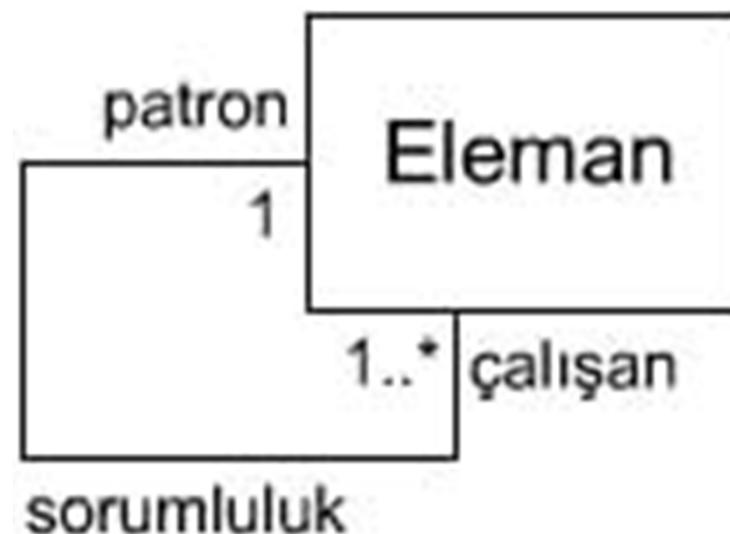
**İlişki
(association):**
'has a' ilişkisidir.



Sınıflar Arası İlişkiler

- **Reflexive(Kendine dönen)ilişki:**

Bir sınıfın sistemde birden fazla rolü vardır.



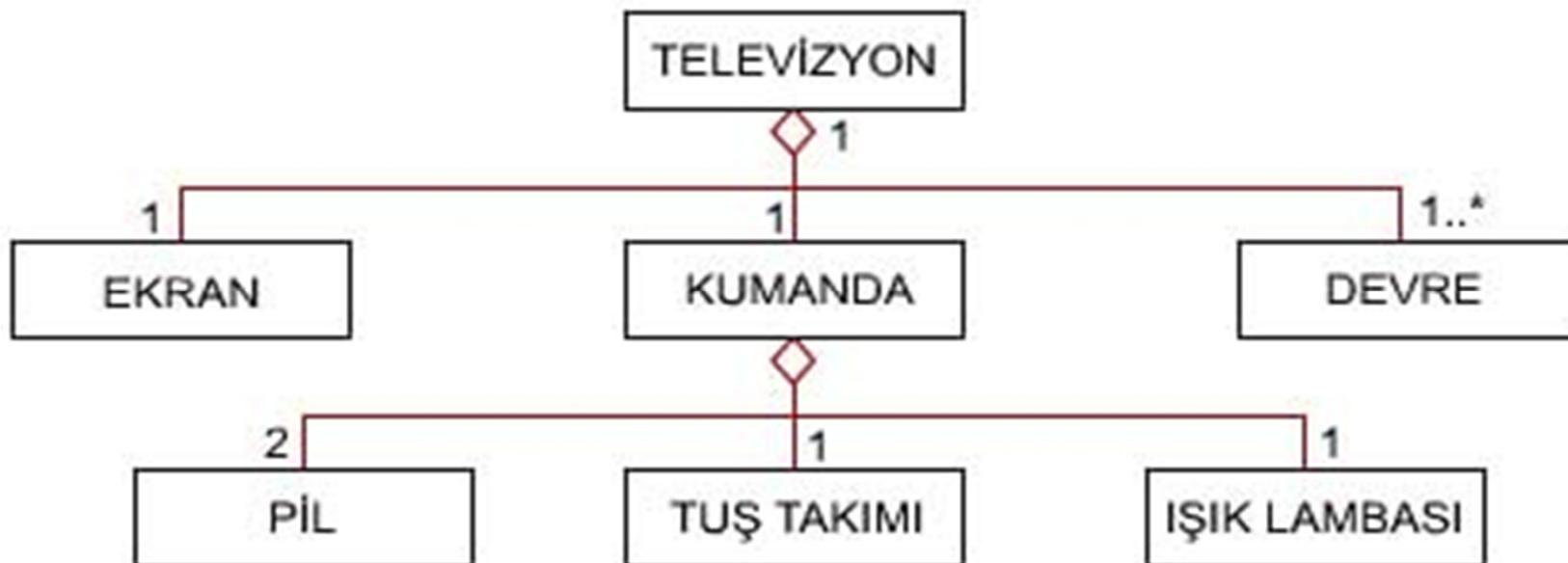
Sınıflar Arası İlişkiler

- 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:
 - **İçerme** (Aggregation)
 - **Kompozisyon -Oluşum** (Composition)

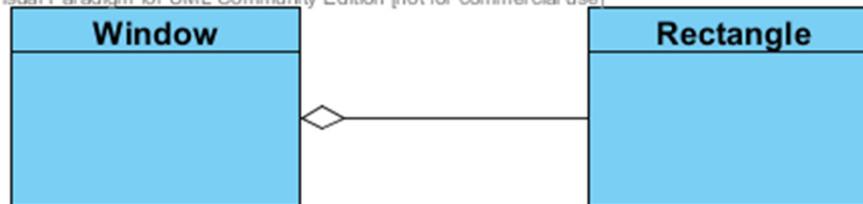
İçerme (Aggregation) Kavramı

- Bütün parça yukarıda olacak şekilde ve bütün parçanın içi boş elmas yerleştirilecek şekilde gösteririz.
- İçi boş elmas ile gösterilen ilişkilerde her bir parça ayrı bir sınıfır ve tek başlarına anlam ifade eder.
- “**owns a**” ilişkisi vardır.
- Parça bütün arasında sıkı bir ilişki yoktur.
 - İki sınıfın yaşam evreleri arasında bir ilişki vardır. Ancak birbirlerinden bağımsız olarak yok edilebilirler.

İçerme (Aggregation) Kavramı



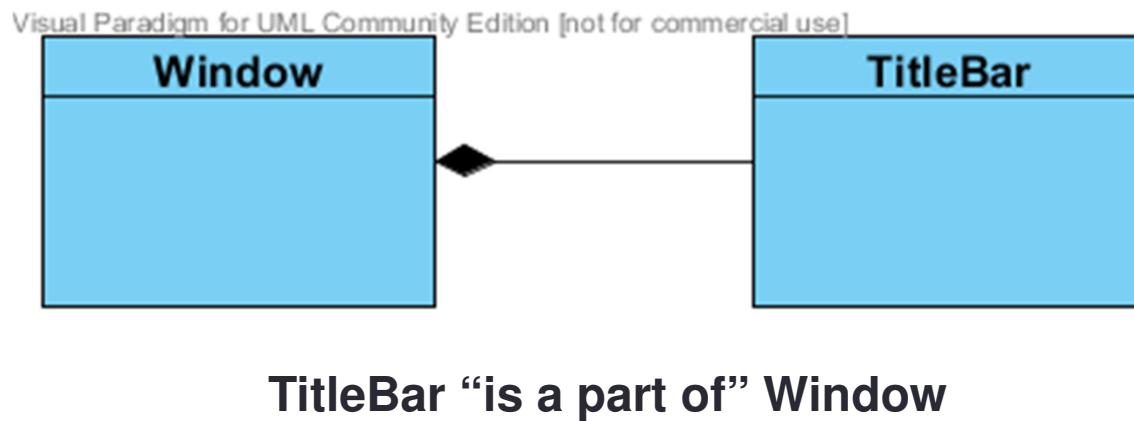
Visual Paradigm for UML Community Edition [not for commercial use]



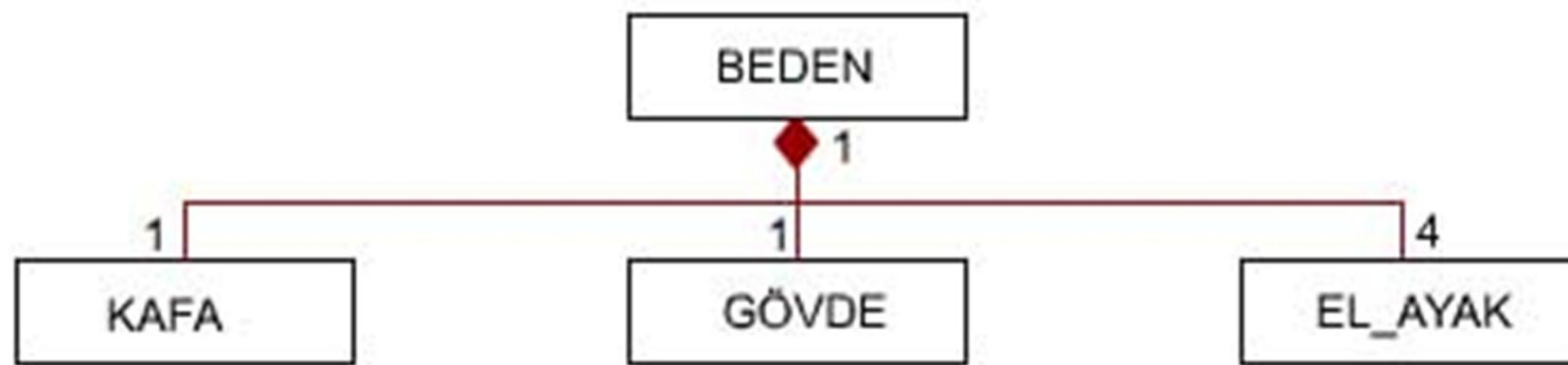
Window “owns a” Rectangle

Kompozisyon(Composite) İlişki

- Sınıflar arasında çok güçlü bir ilişki kurar.
- Parça-bütün ilişkisi kurar.
- Bütün rolündeki nesne yok edildiğinde parça da yok olur.
- “**is part of**” ilişkisi vardır.



Kompozisyon(Composite) İlişki



Bağımlilik (Dependency) İlişkisi

- Sınıflar arasındaki en zayıf ilişkidir.
- İki sınıf arasında dependency ilişkisinin olması demek, bir sınıf diğer sınıfı kullanır ya da onun bilgisine sahiptir demektir.
- “**uses**” ilişkisi vardır.
- Sürekli bir ilişki yok. (transient relationship)
- Bağımlı sınıfın nesnesi diğer sınıfın nesnesini gerekiğinde kullanır.

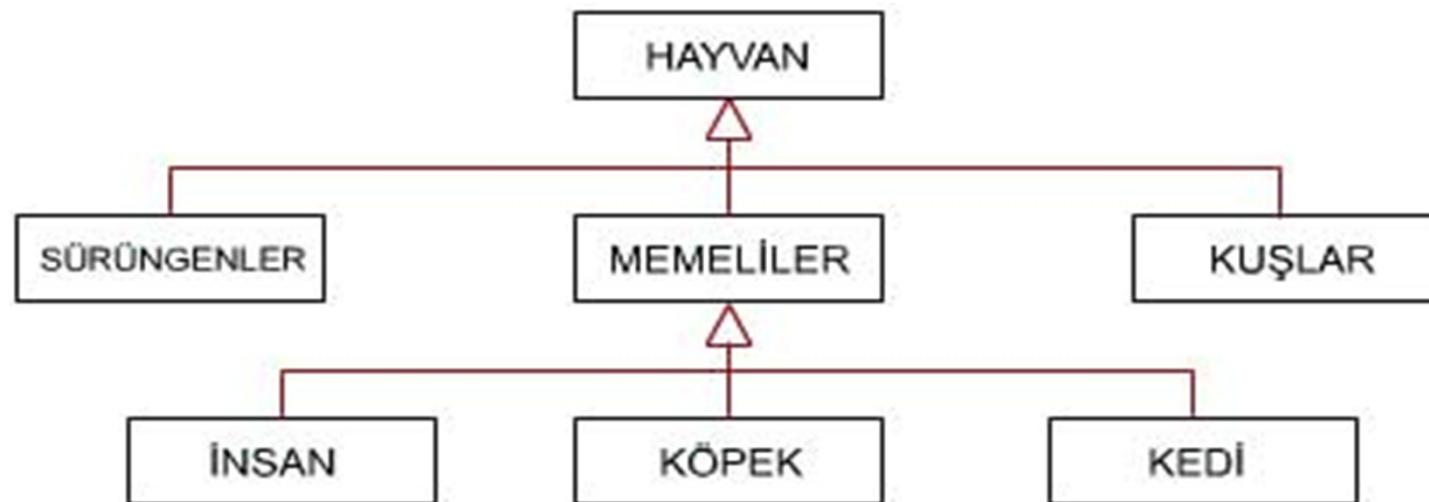
Bağımlilik (Dependency) İlişkisi

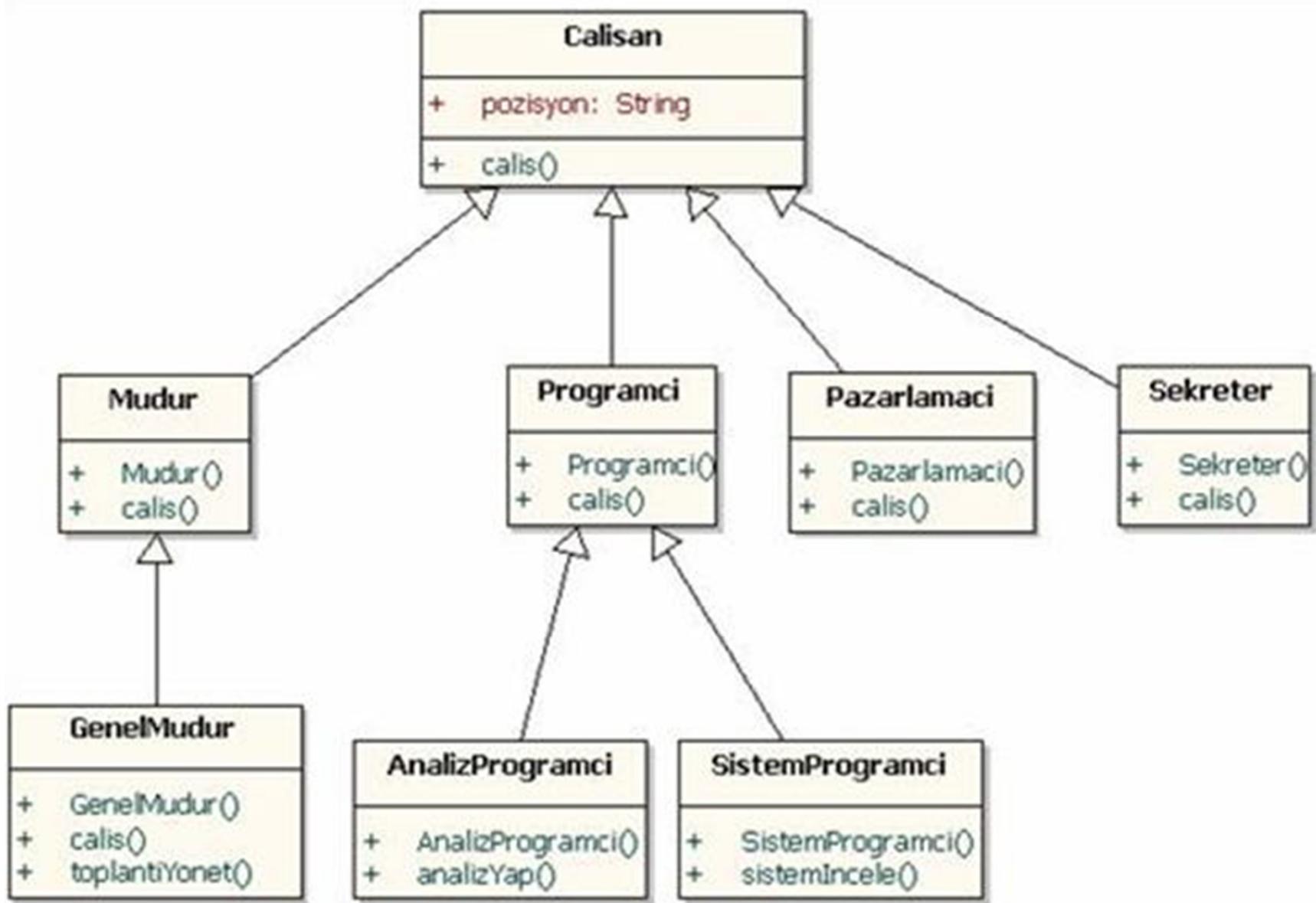


Window “uses” WindowClosingEvent

Genelleştirme (generalization) İlişkisi

- Sınıf genelleştirmesi yapmak amaçlı kullanılır.
- “**is a**” ilişkisi vardır.
- İlişki isimlendirilmez, multiplicity tanımlanmaz.
- UML çoklu kalıtımı destekler.





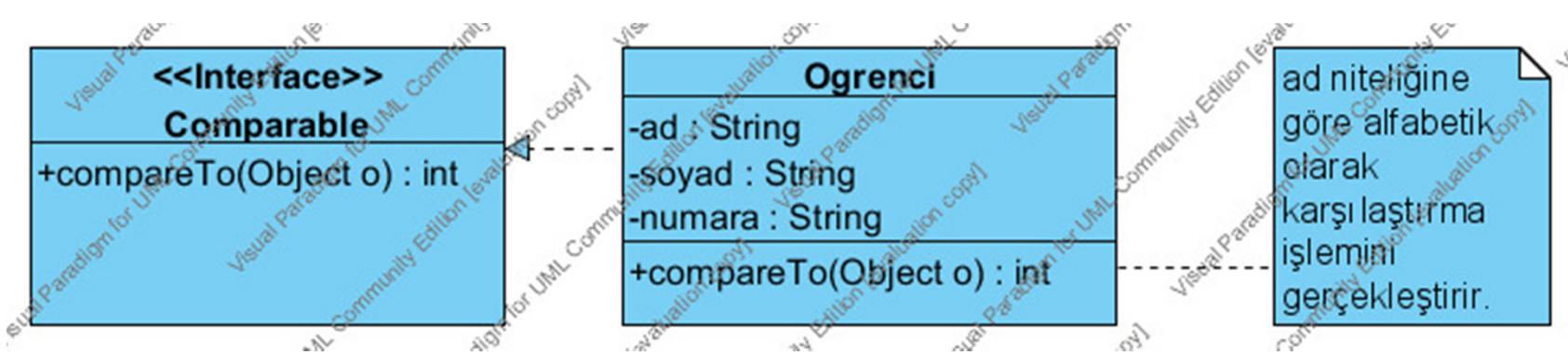
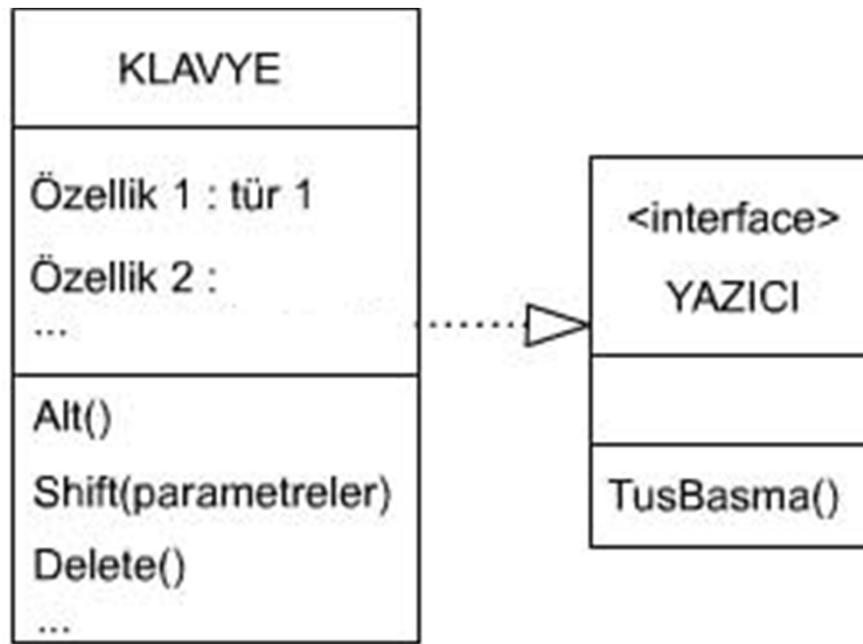
Arayüz (Interface) Kavramı

- Herhangi bir sınıfla ilişkisi olmayan ve standart bazı işlemleri yerine getiren sınıfa benzer yapılara **arayüz** denir.
- Arayüzlerin özellikleri yoktur.
- Sadece bazı işlemleri yerine getirmek için başka sınıflar tarafından kullanılırlar.
- Kesik çizgilerle ve çizginin ucunda boş bir üçgen olacak şekilde gösterilir.
- **Gerçekleme(Realization)**Bir sınıfın bir arayüze erişerek , arayüzün fonksiyonlarını gerçekleştirmesine denir.

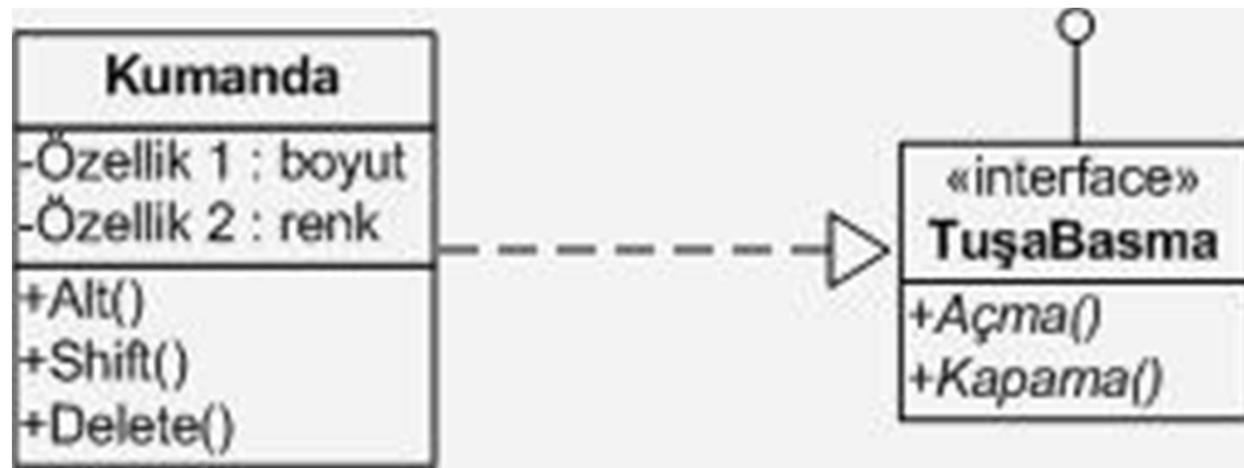
Arayüz (Interface) Kavramı

- **Arayüz** kavramı, nesnelerin davranışlarını belirleyen kurallar bütünü olarak düşünülebilir.
- Ara yüzler kuralları belirlerler ancak, bu kuralların nasıl uygulanacağına karışmazlar.
- Bir sınıf, ilgili ara yüzün yordamlarını gerçekleyerek, ara yüzün belirlediği kurallara uymuş olur.

Arayüz (Interface) Kavramı

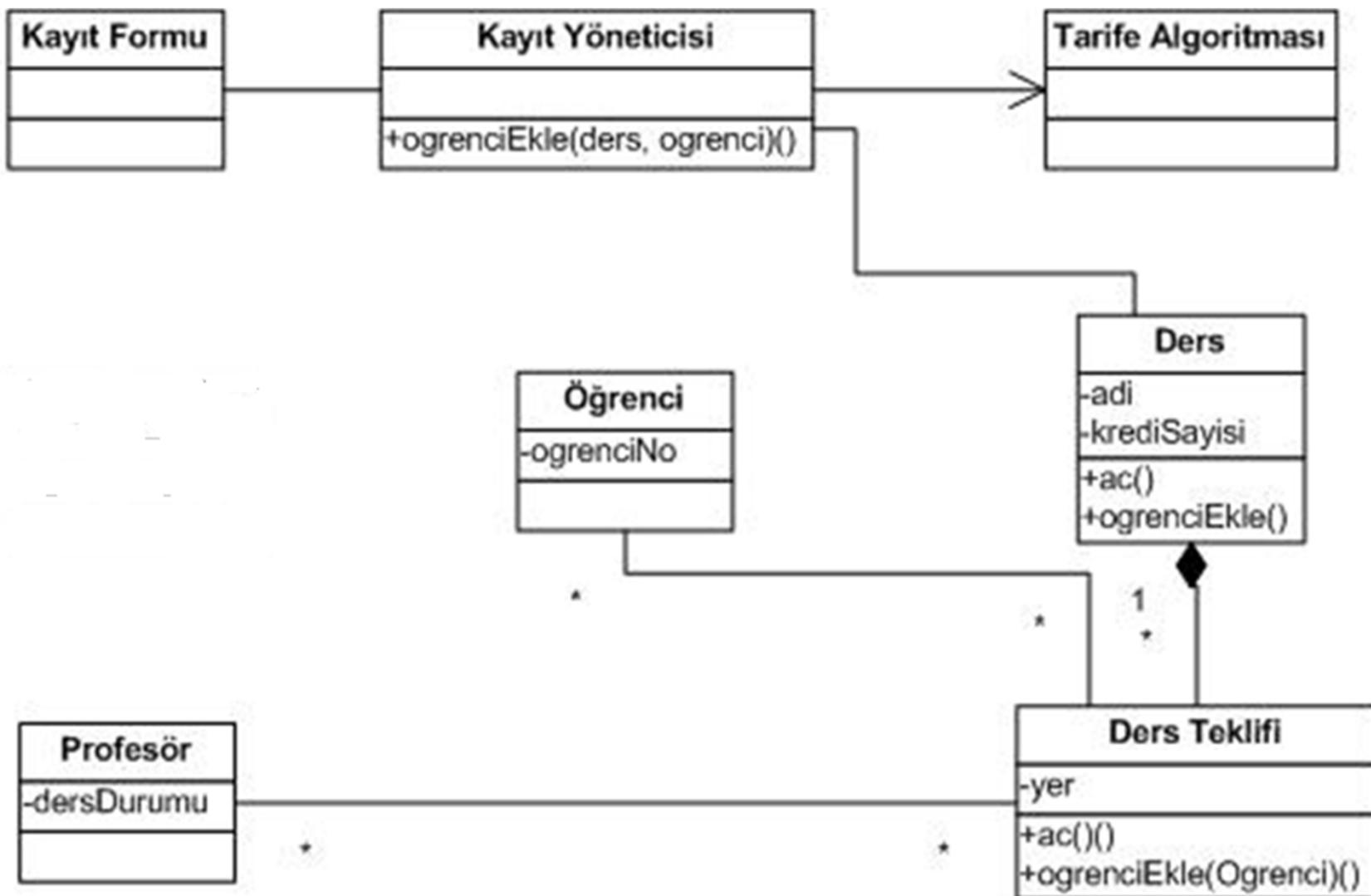


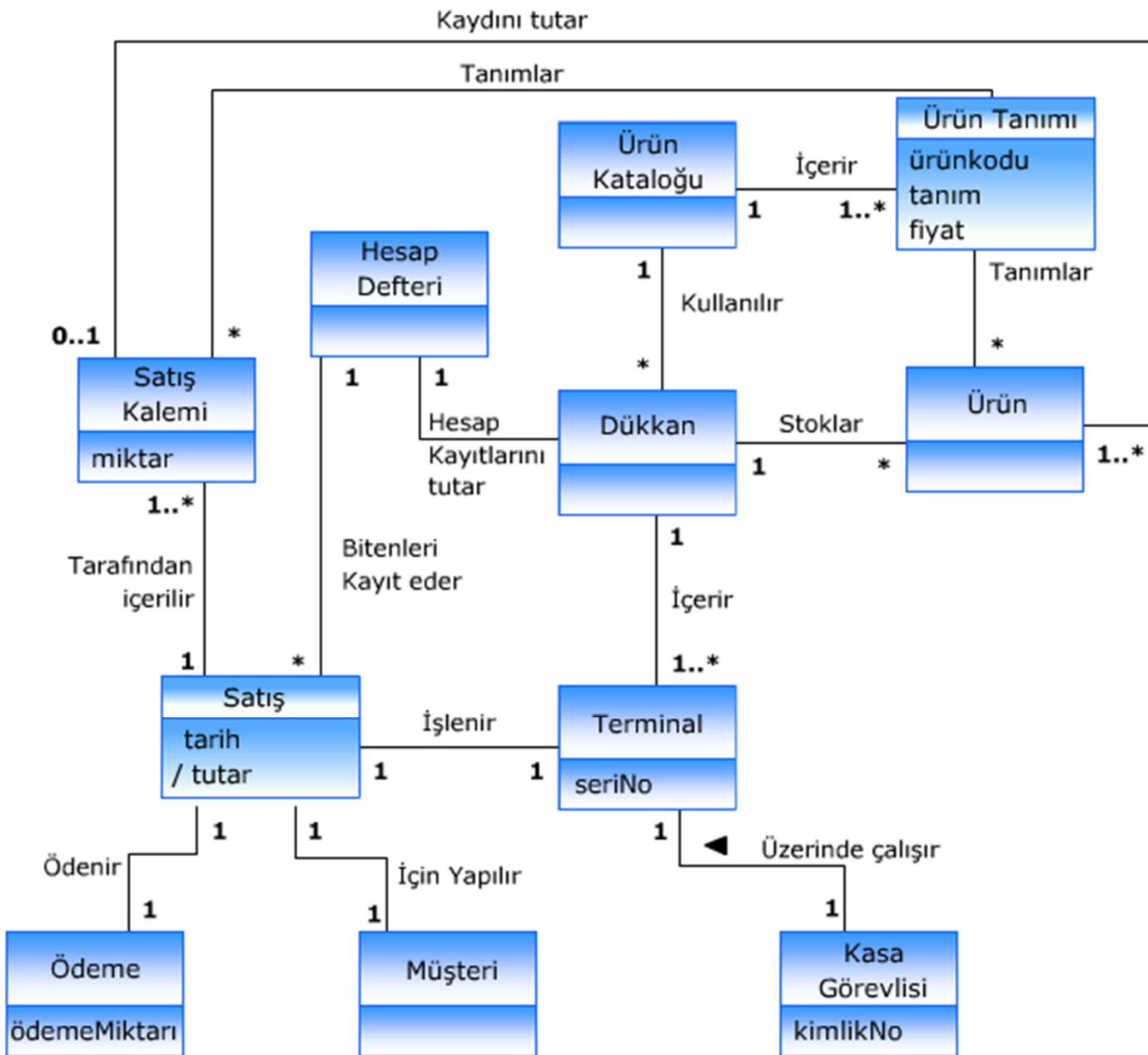
Arayüz (Interface) Kavramı

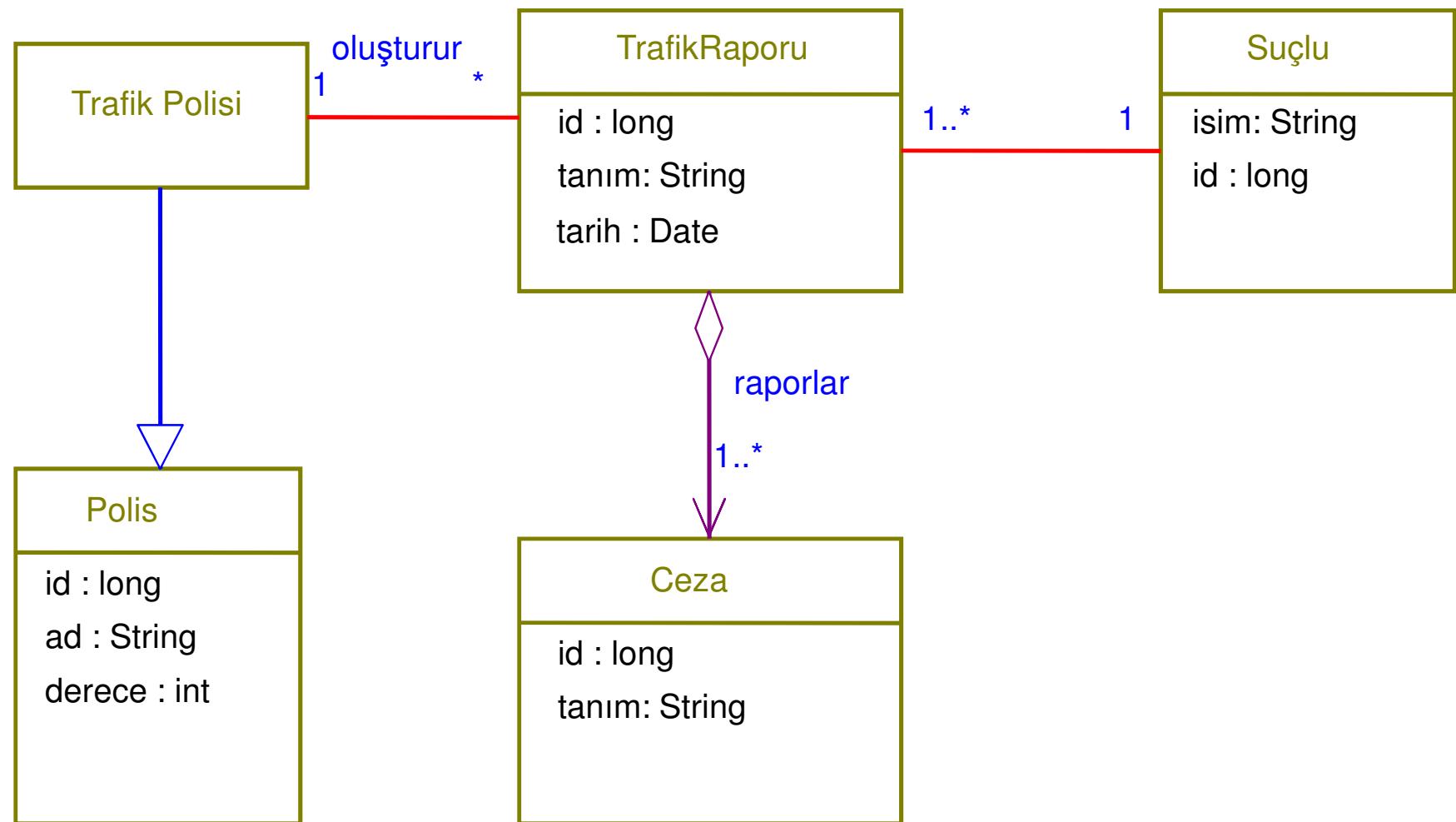


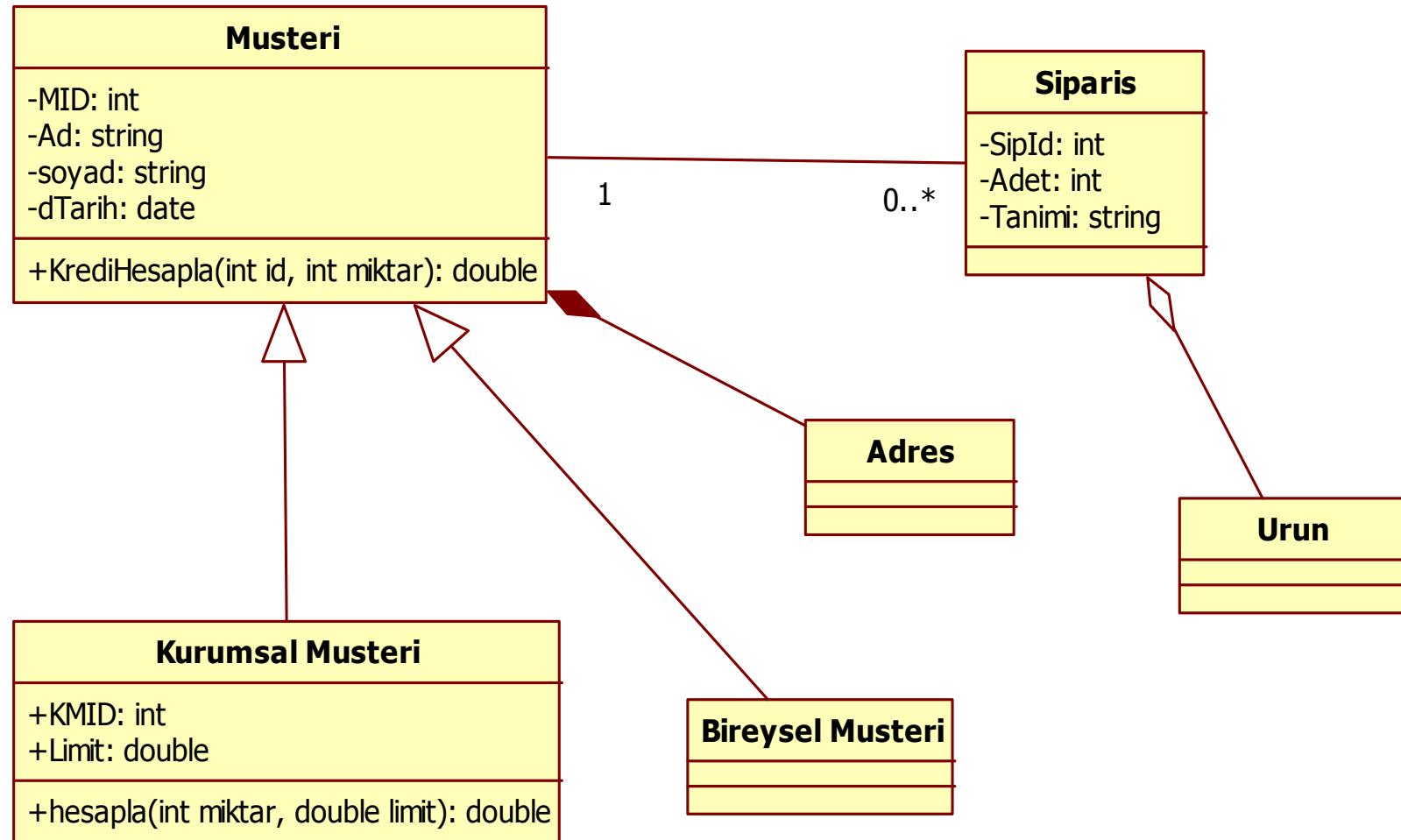
Bir TuşaBasma arayüzü yapılarak istenirse
Kumanda sınıfında, istenirse de Klavye sınıfında
kullanılabilir.

Sınıf Diyagramları - ÖRNEKLER

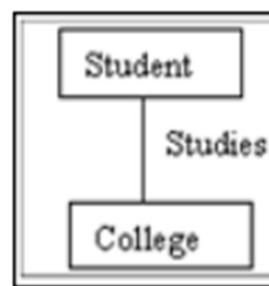
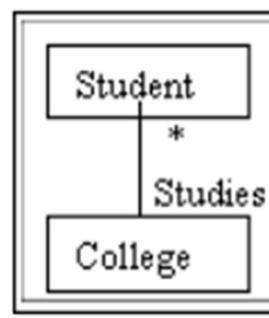
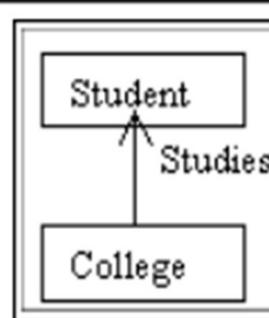


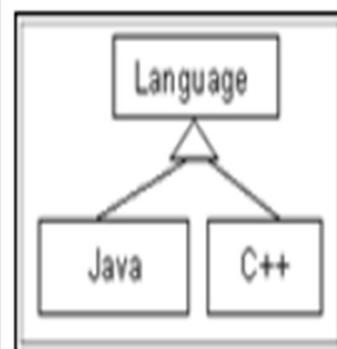


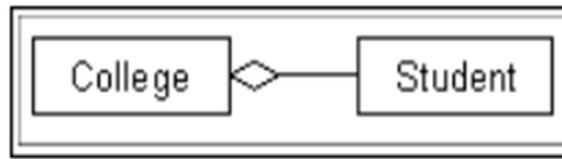
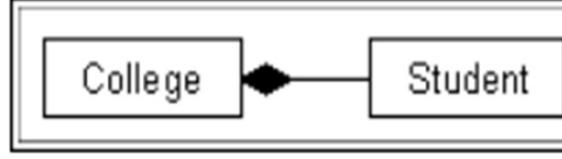
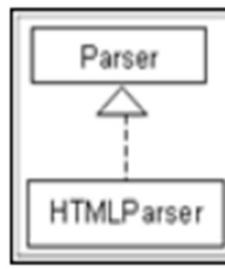




Sınıf İlişkileri Özeti Tablosu

No	İlişki	Örnek Gösterim	Açıklama
1	Bağıntı(Association)		İki sınıfın herhangi bir şekilde birbirleriyle iletişim kurmasıdır. Örneğin : Öğrenci ve Okul arasındaki ilişki
1.a	Çokluk (Multiplicity)		İki sınıf arasındaki 1:n ilişkidir. Birden fazla öğrencinin okul ile olan ilişkisi olarak açıklanabilir. Şekildeki * işaretini 1:n ilişkisi göstermektedir.
1.b	Yönlü Bağıntı (Directed Association)		Sınıflar arasındaki tek yönlü ilişkiyi gösterir. Ok işaretini ile gösterilir.

1.c	Döndüşlü Bağıntı(Reflexive Association)	Belirgin bir çizimi yoktur.	Bir sınıfın kendisiyle kurduğu ilişkidir.Bu tür ilişkiler genellikle bir sınıfın sistemde birden fazla rolü varsa ortaya çıkar
2	Kalıtım/Genelleme (Inheritance/Generalization)	 <pre> classDiagram class Language class Java class C++ Language < -- Java Language < -- C++ </pre>	Bu ilişki, bir nesnenin bir diğerinin özel bir türü olduğu gerçekini modellemek için kullanılır.

3	İçerme(Aggregation)		İki sınıf arasındaki "sahiptir" veya içerir türünden bağıntıları modellemekte kullanılır. Bütün parça içi boş elmas ile gösterilir.
3.a	Oluşum(Composition)		İçerme ilişkisinin farklı bir çeşididir. Tüm bağıntılar içinde en güçlü olanıdır. Parça-bütün ilişkilerini modellemekte kullanılır.
4	Gerçekleştirme(Realization)		Kullanıcı arayüzlerinin modellemesinde kullanılır. Arayüz yalnızca method adlarını ve bunların parametrelerini içermektedir.