UML

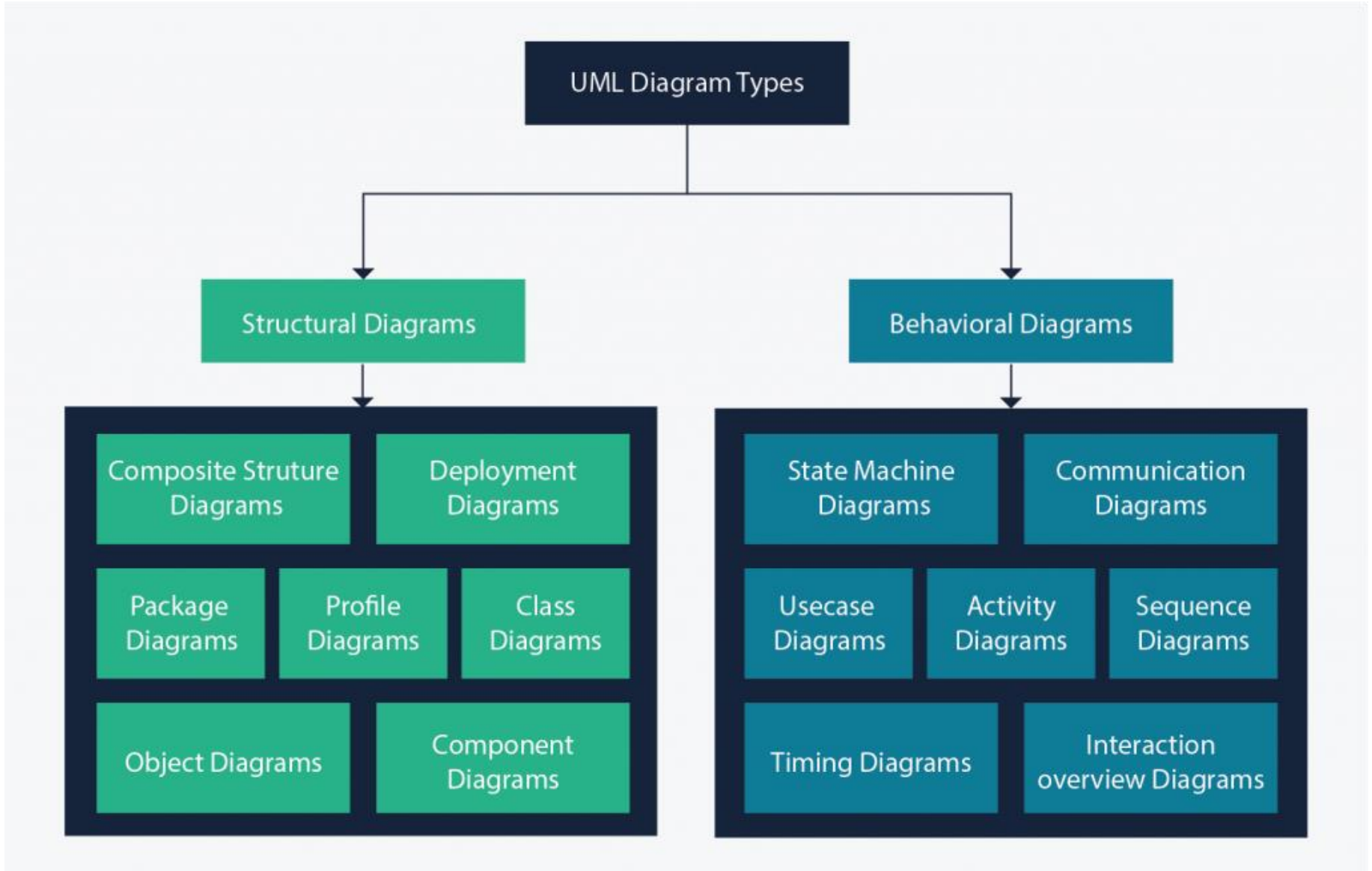
Unified Modeling Language

UML Nedir

- Unified Modeling Language
- Object Management Group (OMG)
- Yazılım geliştirmede yazılım sistemlerinin tasarlanmasında kullanılır.
- Farklı amaçlar için tasarlanmış çizimler
- Sistemler ve süreçler farklı seviyelerde görsel olarak tasarlanabilir.

UML Faydaları

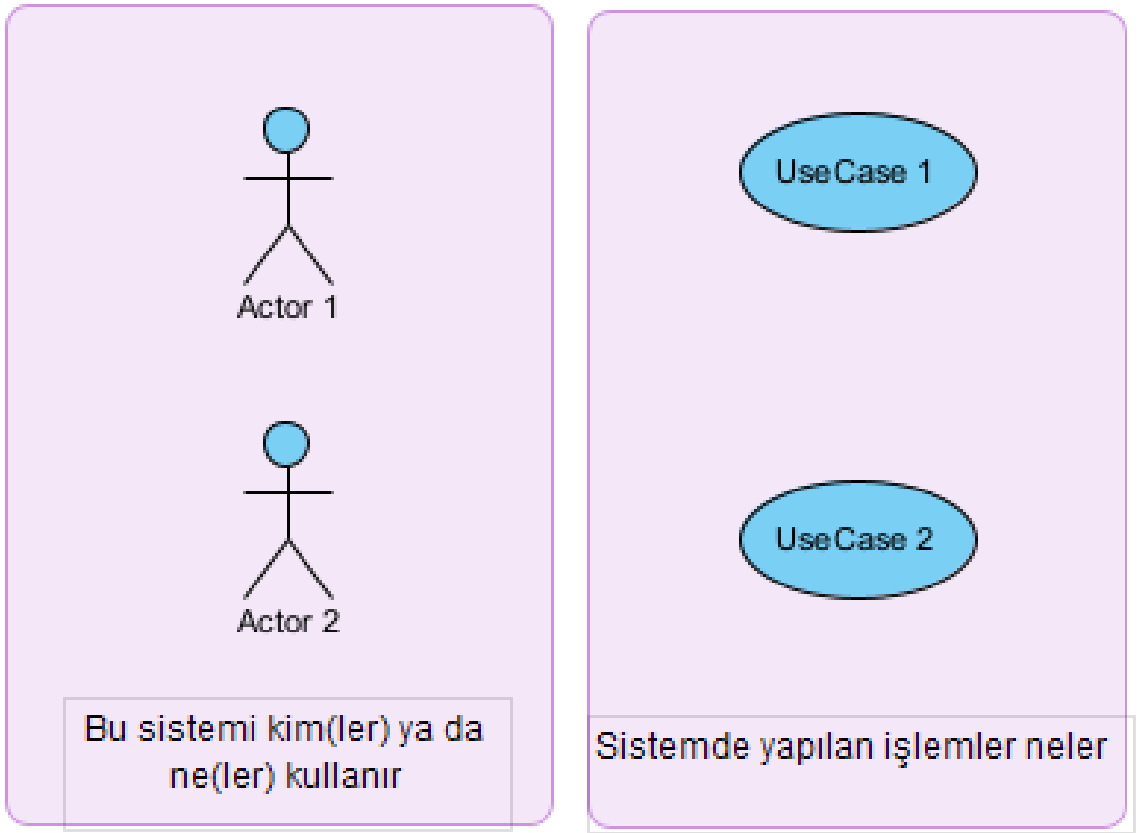
- UML, standartlaşmış bir yapı olduğundan dolayı, dili bilenler tarafından okunur ve aynı şekilde yorumlanır.
- Takım çalışmasına birebirdir.
- Yazılımlardaki hataları (bug) azaltmaya yarar.
- UML ile sistem veya yazılım başta belirlendiği için tekrar tekrar dizayn ve kod yazmanın önüne geçilmiş olunur.



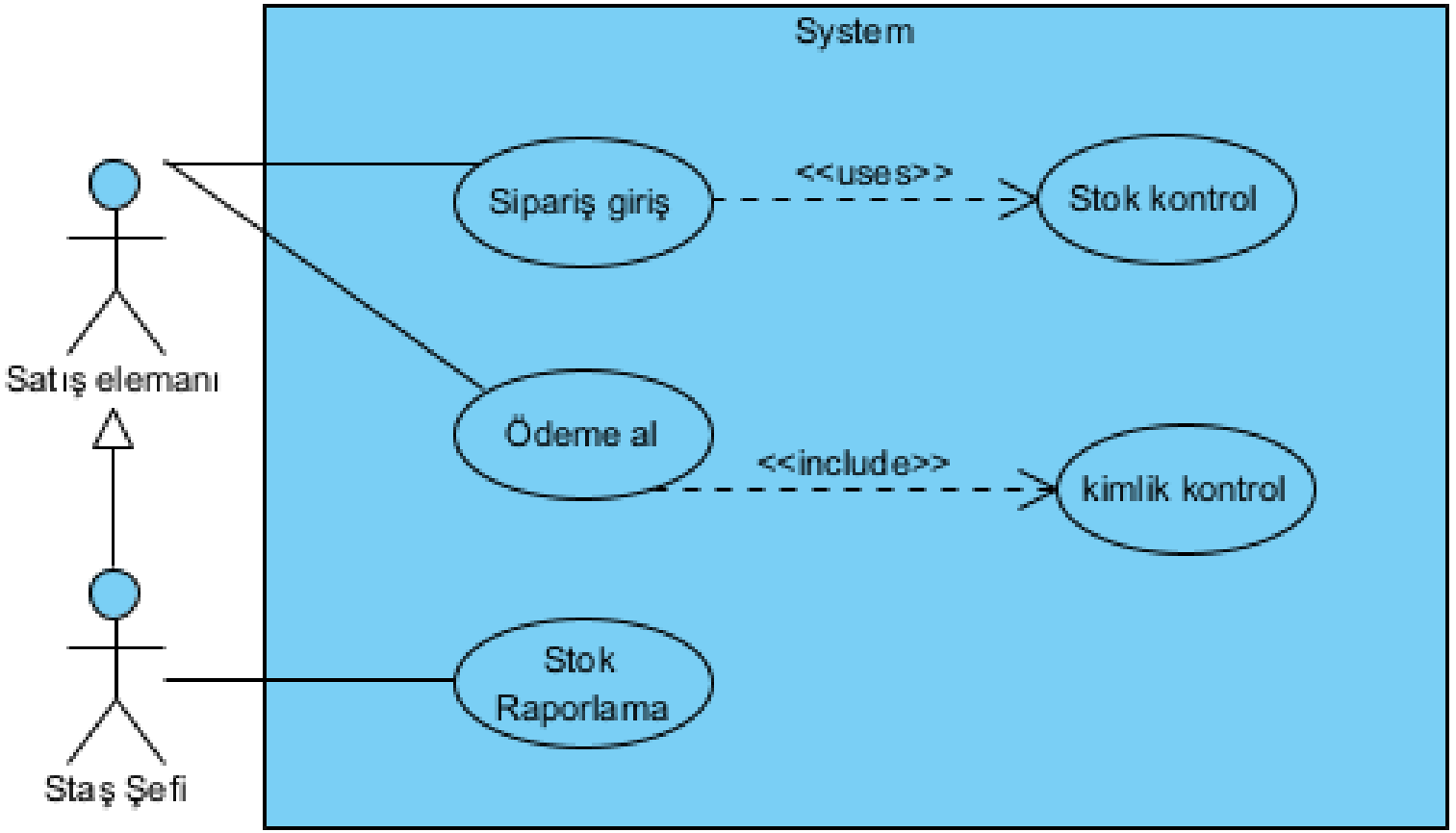
Sistemi farklı açılarda analiz etmek için farklı diyagramlar kullanılmakta

Kullanım Senaryosu (use case) Diyagramları

- Analiz aşamasında problemin tanımlanmasında kullanılır.



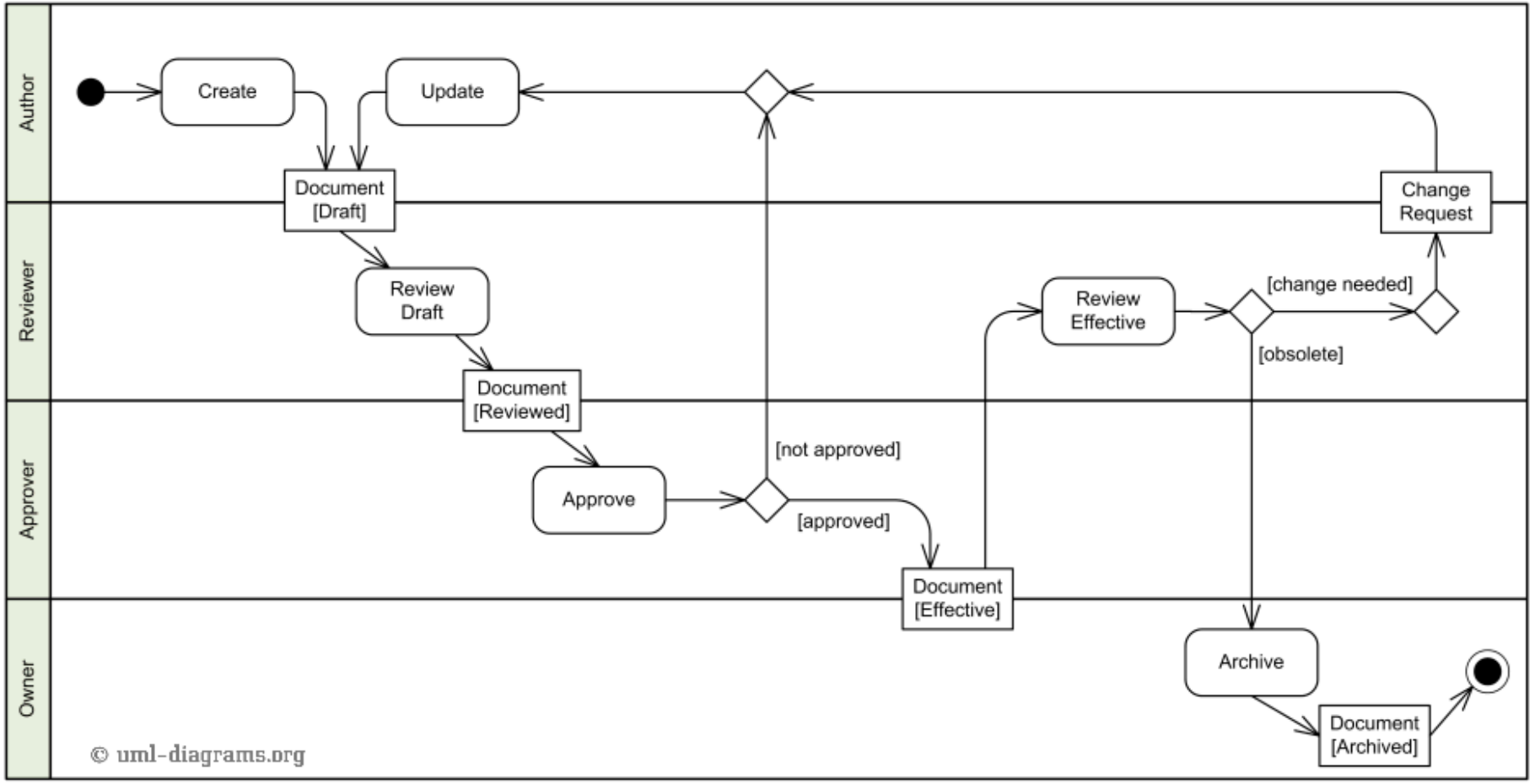
Kullanım Senaryosu (use case) Diyagramları



Faaliyet (Activity) Diyagramı

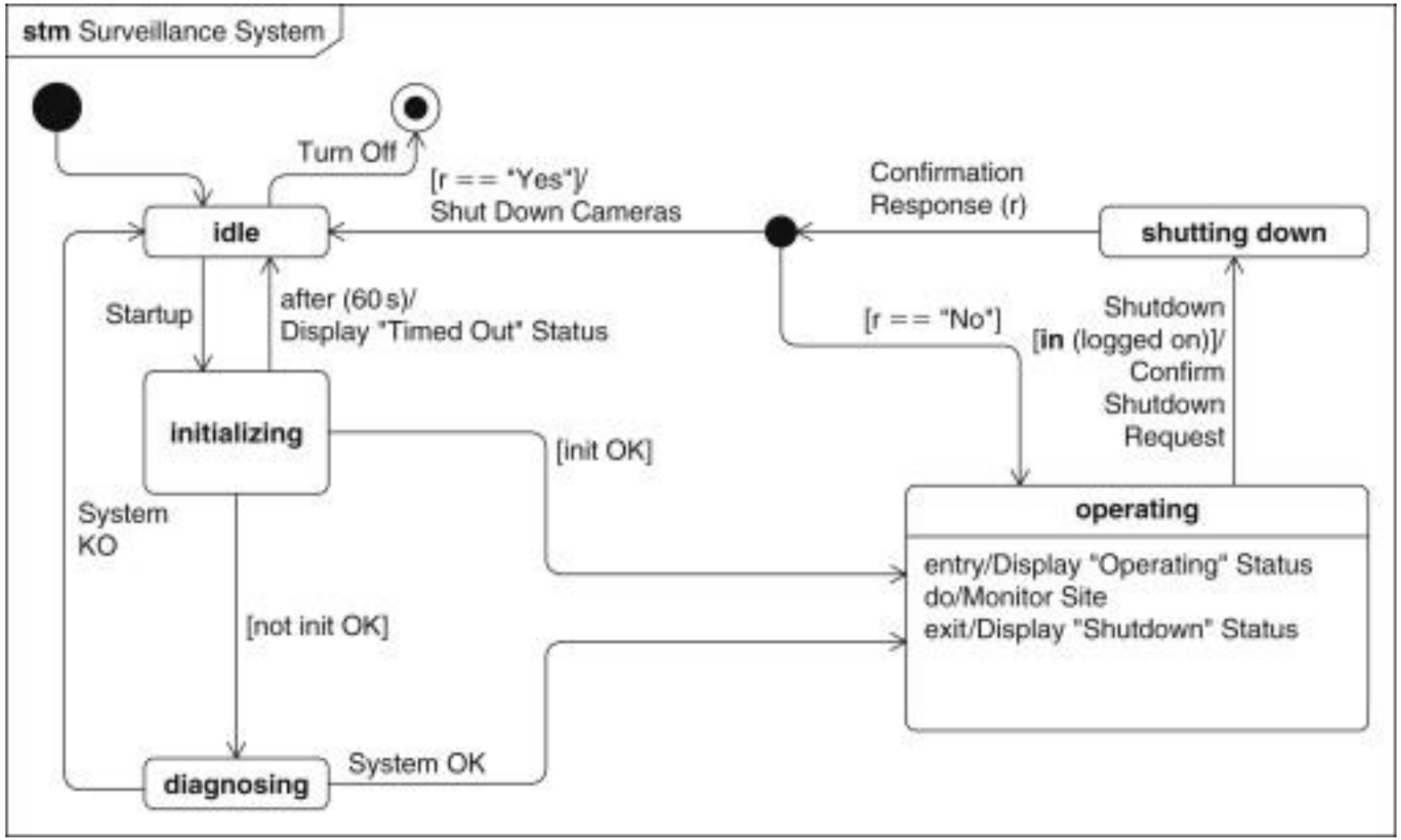
- İş akışlarının görselleştirilmesi

  Aktörler -> Faaliyetler -> Çıktılar -> Durumlar -> Akışlar



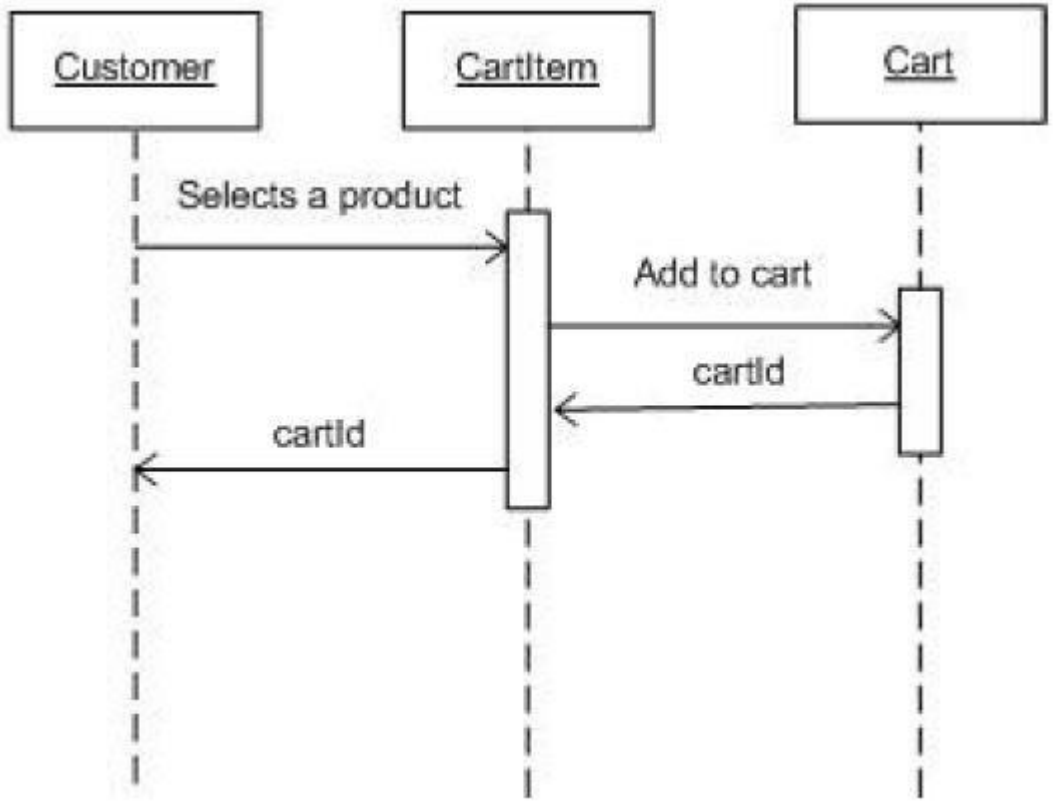
Durum (State) diyagramı

- Sistem durumlarını görselleştirir.
  - Durumlar
  - Eylemler
  - Kararlar



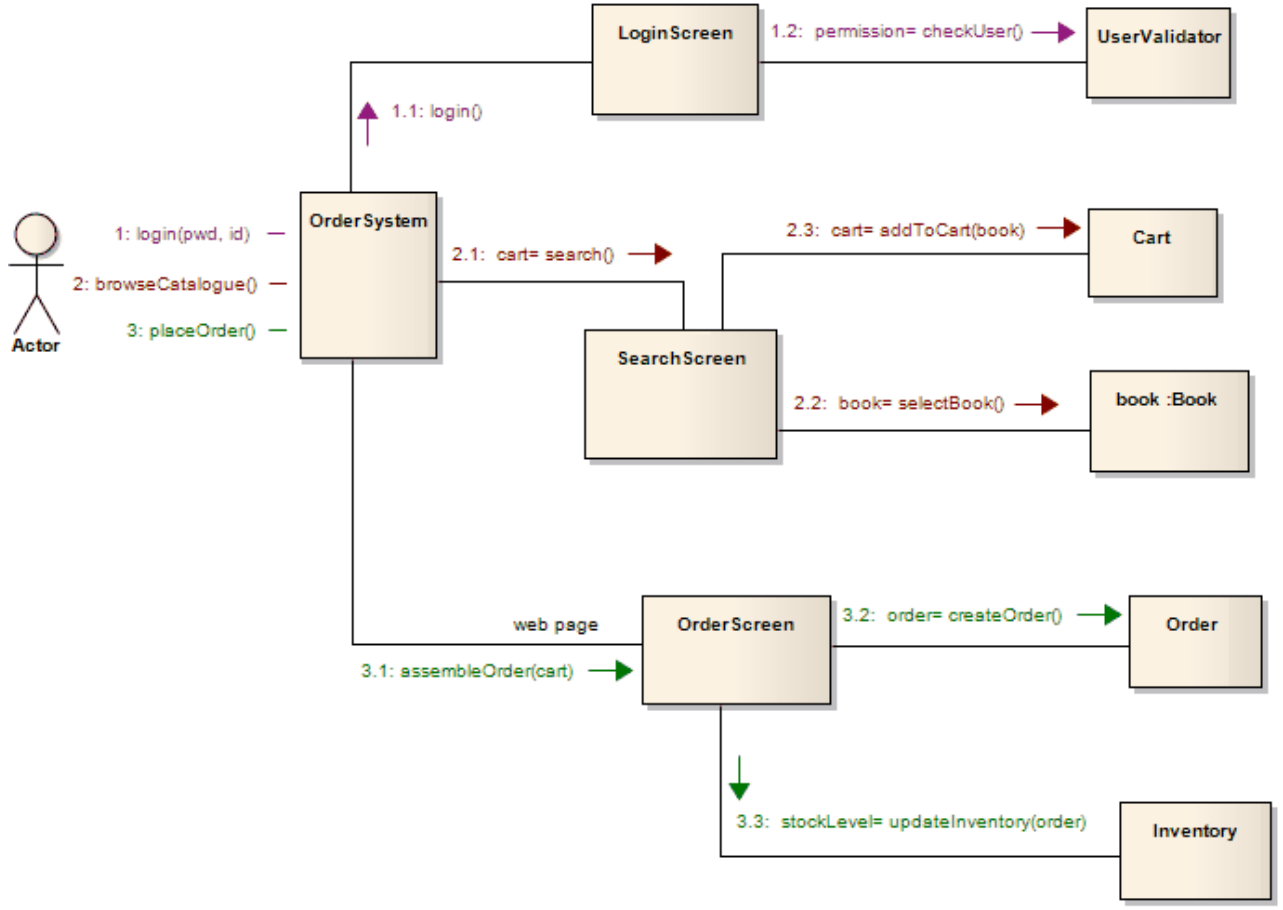
Etkileşim (sequence) diyagramı

- Sınıf ve nesnelerin zamana bağlı ilişkilerini görselleştirir.



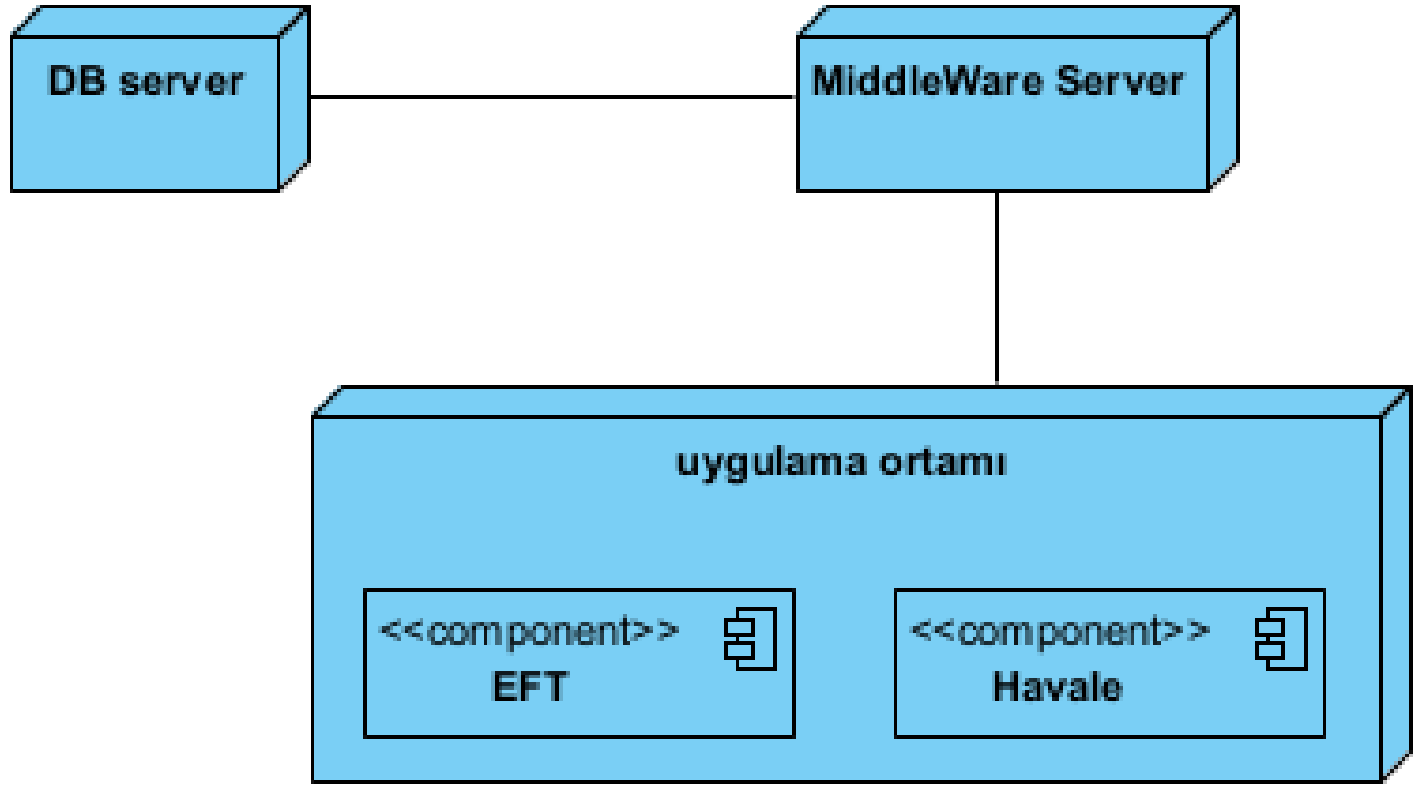
İletişim (Communication) Diyagramı

- Nesneler arasındaki ilişkiyi zaman akışı olmaksızın görselleştirir.
- Etkileşim (sequence) diyagramının zaman kavramı olmayan versiyonu



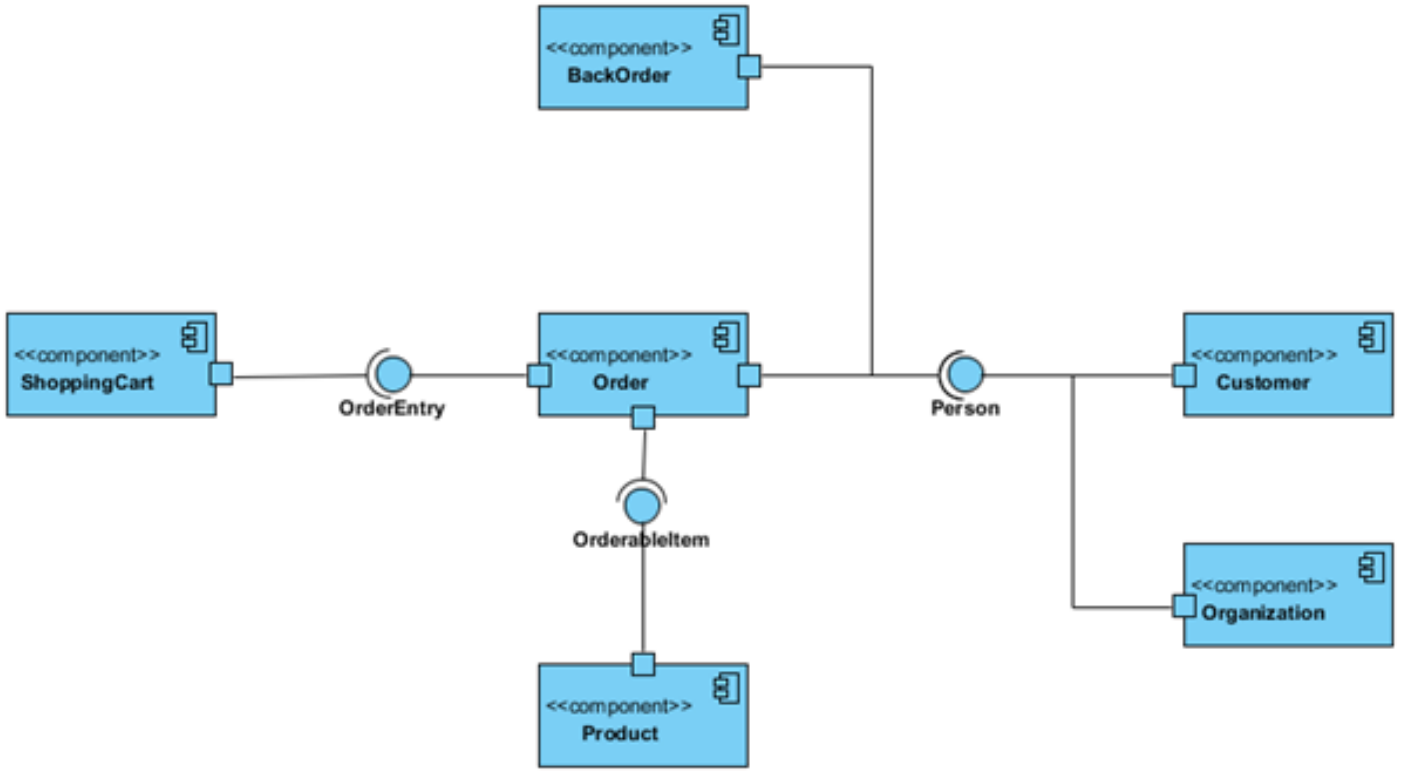
Deployment (Dağıtım) Diyagramları

- Yazılım sistemi ile ilişkili fiziksel mimari ve topolojiyi en temel unsurları ile ifade eder
  - Donanım
  - İşletim sistemi
  - Sunucu
  - Vs.
- Node (düğüm)



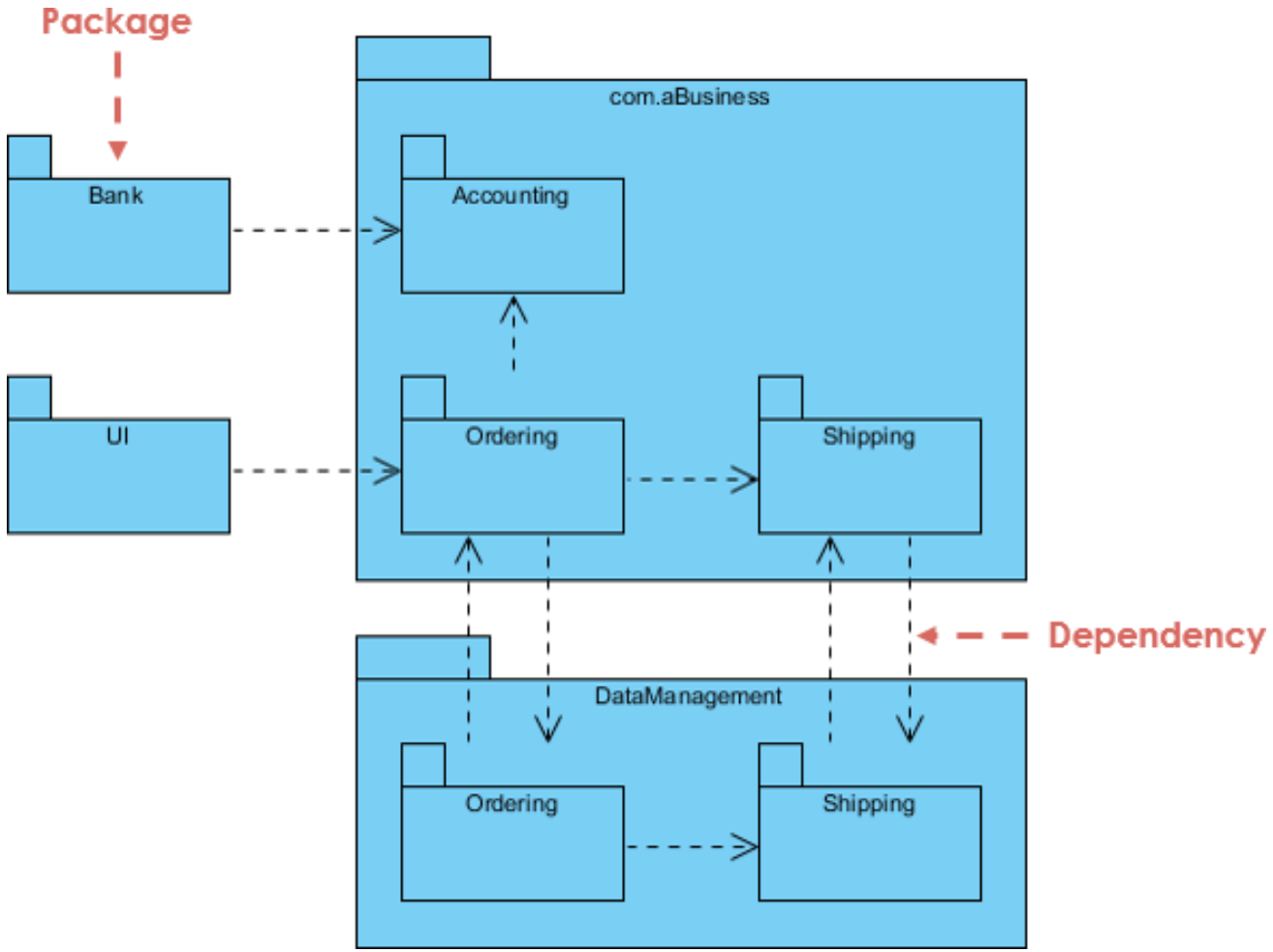
Component (Bileşen) Diyagramları

- İş akışları ve gereksinimler belirlendikten sonra sınıf veya nesne düzeyinde tasarım yapmak lazım
- Büyük sistemlerde sınıf düzeyinde tasarım çok karmaşık olacaktır.
- Bunun yerine sistemin majör bileşenleri tasarlanır.
- Bileşenler
  - Dll
  - Jar
  - Ocx
  - Xml
  - .java
  - .cpp
  - İşletim sistemi
  - Veritabanı



Paket Diyagramı

- Birbiri ile alakalı sınıfları gruplayan yapılar ve bunların ilişkileri



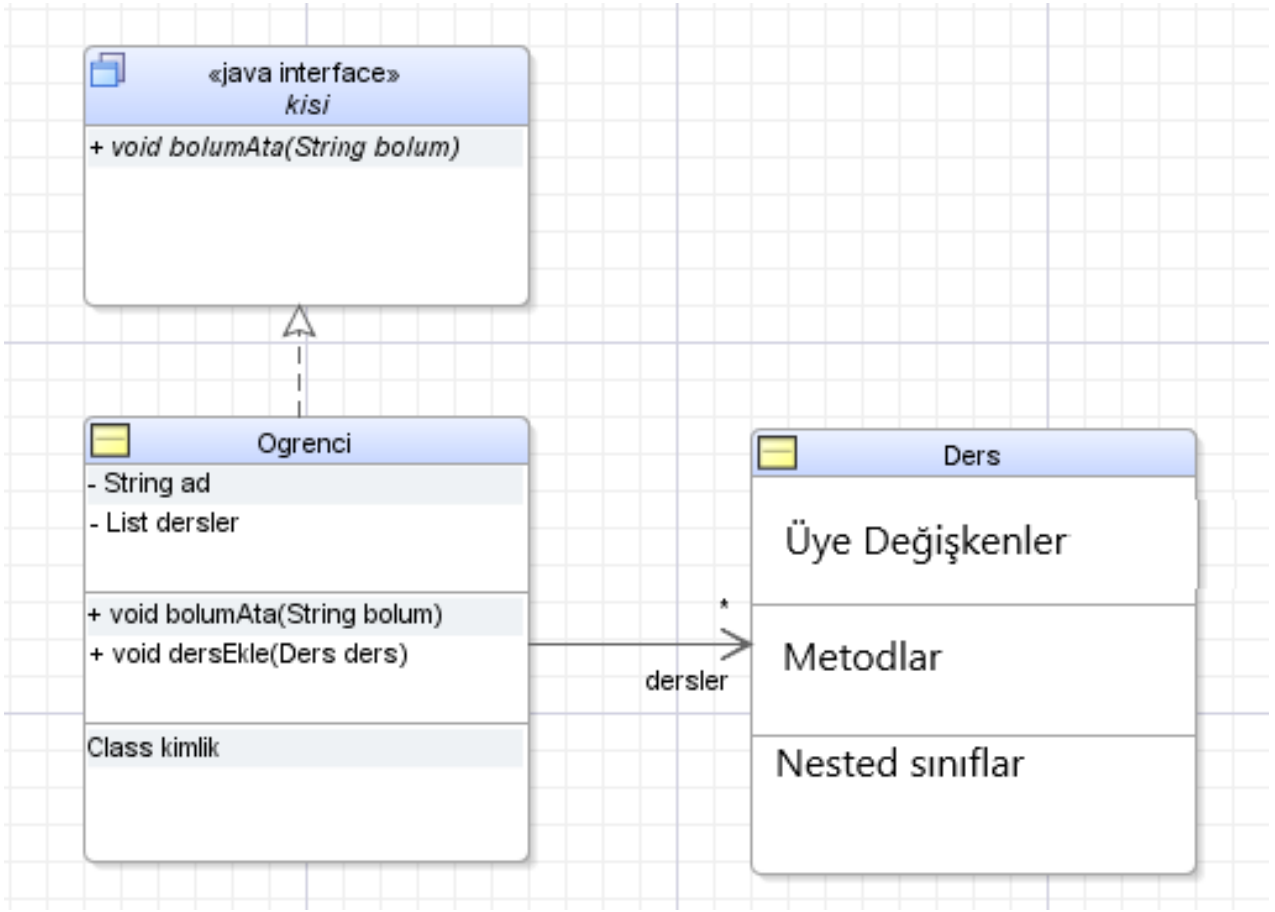
Sınıf Diyagramları

- Uygulamanın tasarım ve analizi
- Yazılımın en temel yapılarının tanımlanması
- Sorumluluklarının tanımlanması
- Sistemdeki sınıflar
  - yapıları
  - ilişkileri
- Tersine mühendislik

Sınıf Diyagramları

- Sınıflar
- Arayüzler
- Sınıf ilişkileri
- Streotype (<< >>)
- Erişebilirlik düzeyleri

Public	+
Private	-
Protected	#
Package	~

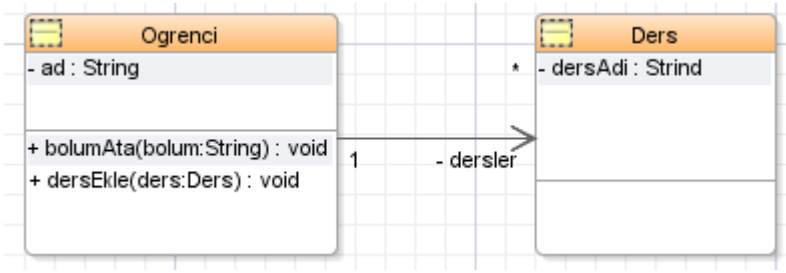


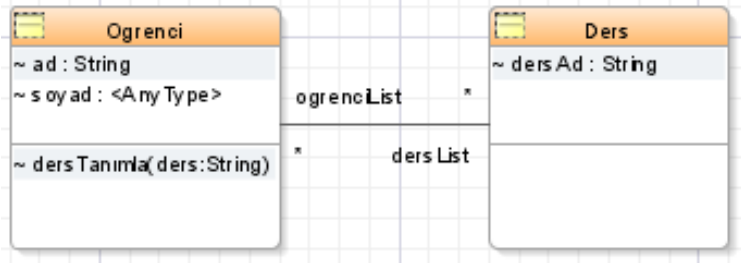
Sınıflar Arası İlişki Türleri

- Association (Referans, Birliktelik)
- Aggregation (barındırma, içerme)
- Compozition (bağımlılık)
- Generilazition (genelleme)
- Dependency (bağımlılık)
- Usage (kullanma)
- Realization (gerçekleme)

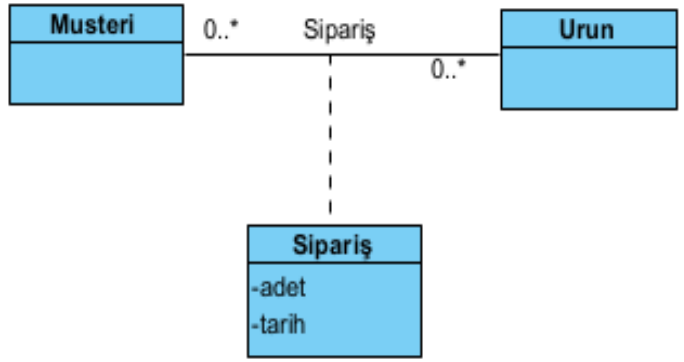
Association (Referans, Birliktelik)

- En temel ilişki türü
- Has-a ilişkisi olarak da isimlendirilir
- Bir sınıfın, diğer sınıf türünden nesne referansı içermesi
- Unidirectional - bidirectional
- Multiplicity (adet)



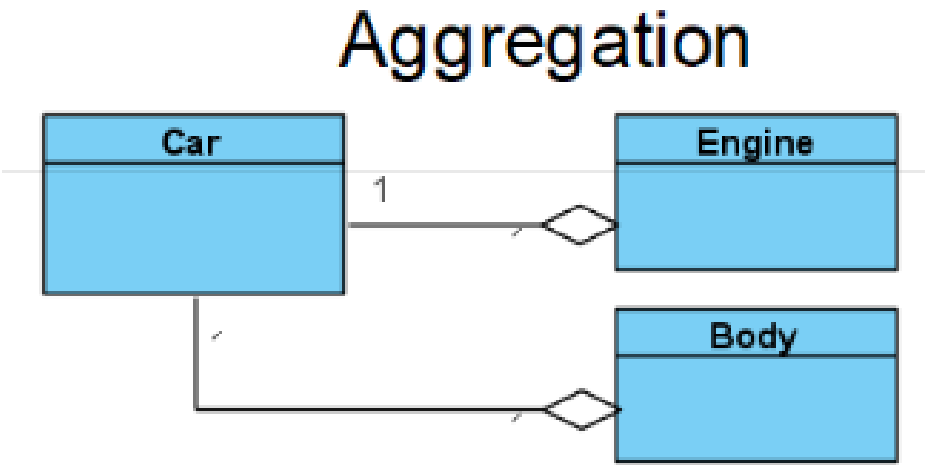


1	1 adet
*	Belirsiz
0..1	Var ya da yok
1..*	Bir ya da sonsuz
m..n	2 değer arasında
0..*	Hiç ya da sonsuz



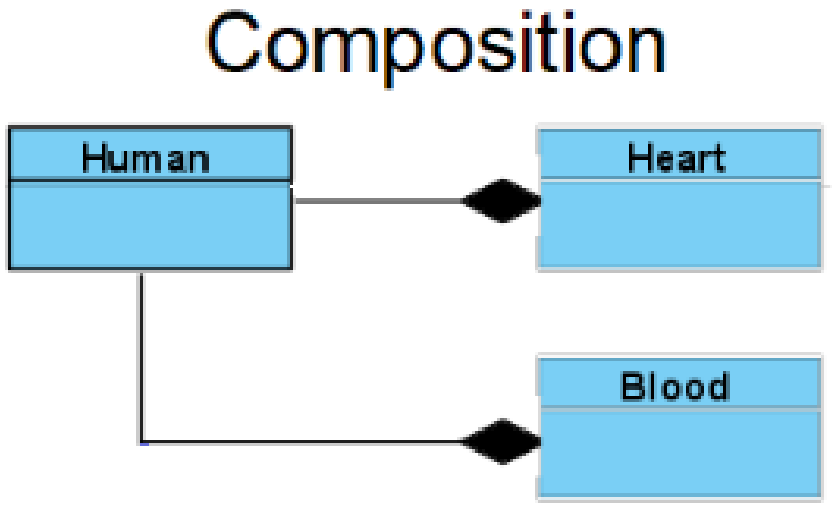
Aggregation (barındırma, içerme)

- Association bağlantının özel bir türüdür
- Kodlamada genelde fark yoktur
- Kavramsal fark vardır
- Parça bütün ilişkisi olduğunu ifade eder



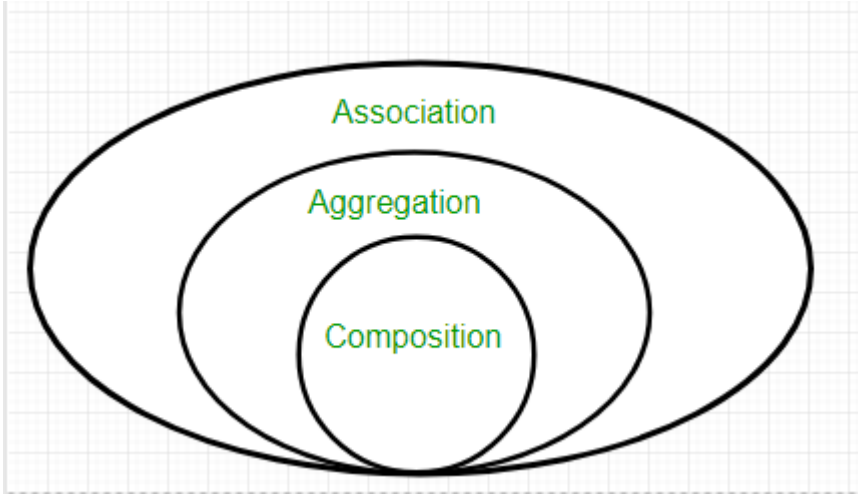
Compozition (bağımlılık)

- Parça bütün ilişkisi
- Parça ve bütün birbirinden bağımsız bulunamaz



Association, Aggregation, Compozition

- Kodlamada farklılık yok
- Aralarında mantıksal farklılılık var



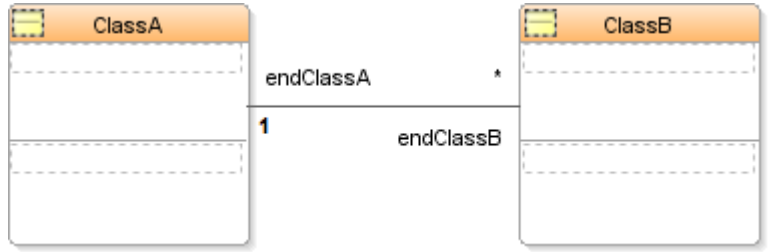
Nasıl kodlarız

```
public class ClassA
{
    List<ClassB> endClassB;
    …
}
```

```
public class ClassB
{
    List<ClassA> endClassA;
    …
}
```

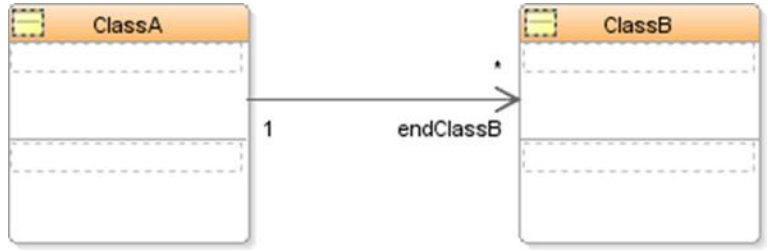

```
public class ClassA
{
    List<ClassB> endClassB;
    …
}
```

```
public class ClassB
{
    ClassA endClassA;
    …
}
```




```
public class ClassA
{
    List<ClassB> endClassB;
    …
}
```

```
public class ClassB
{
    …
}
```

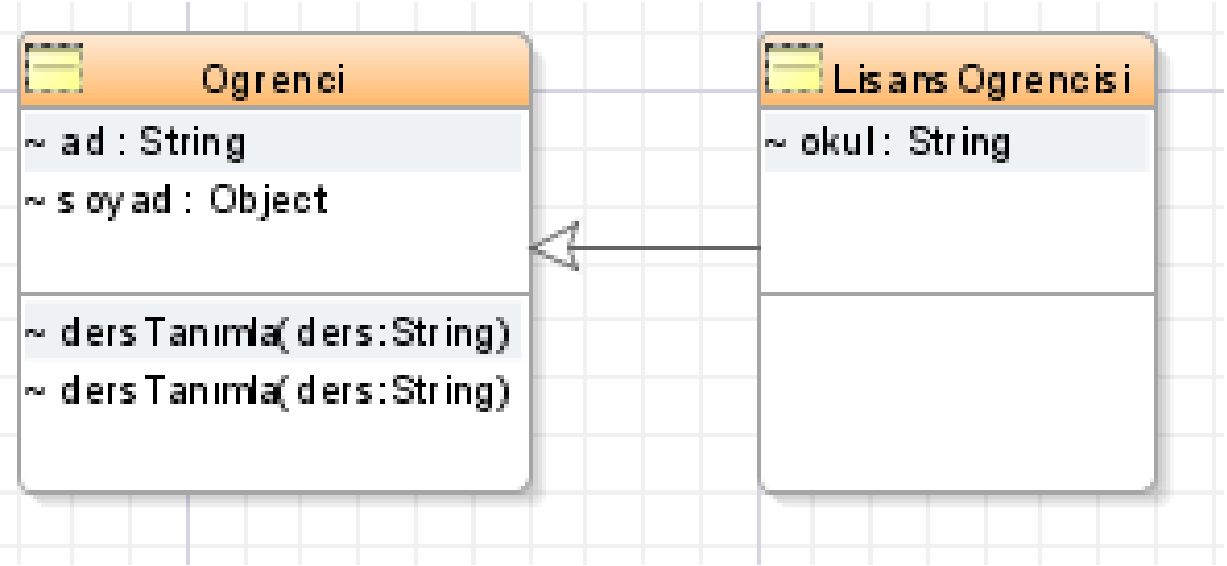


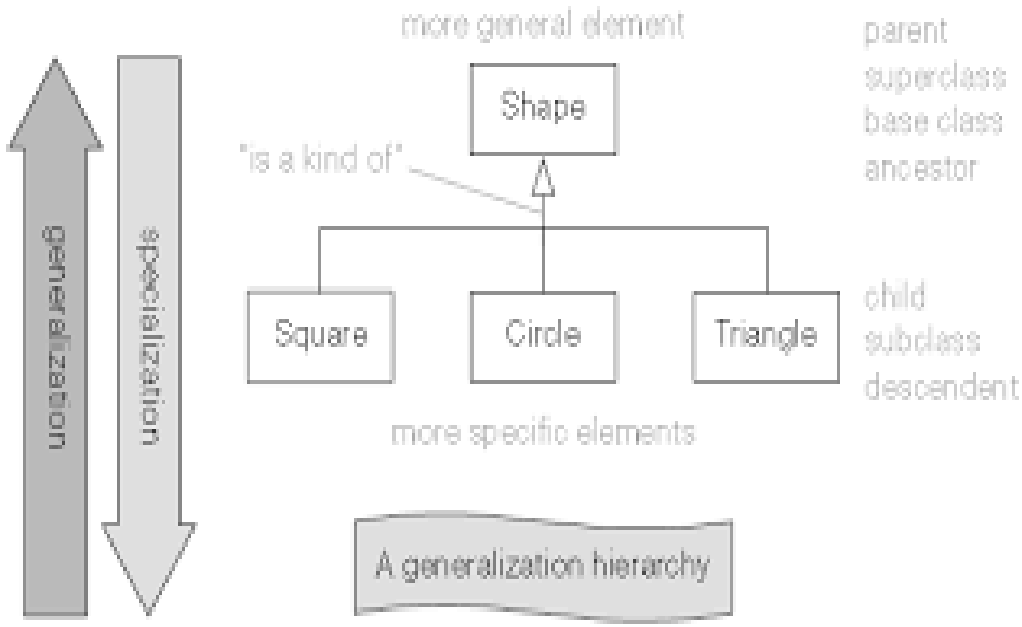

```
public class ClassA
{
    ClassB endClassB;
    …
}
```

```
public class ClassB
{
    …
}
```

Generilazition (genelleme)

- Kalıtım sürecini modeller
- is-a ilişkisi olarak da isimlendirilir.

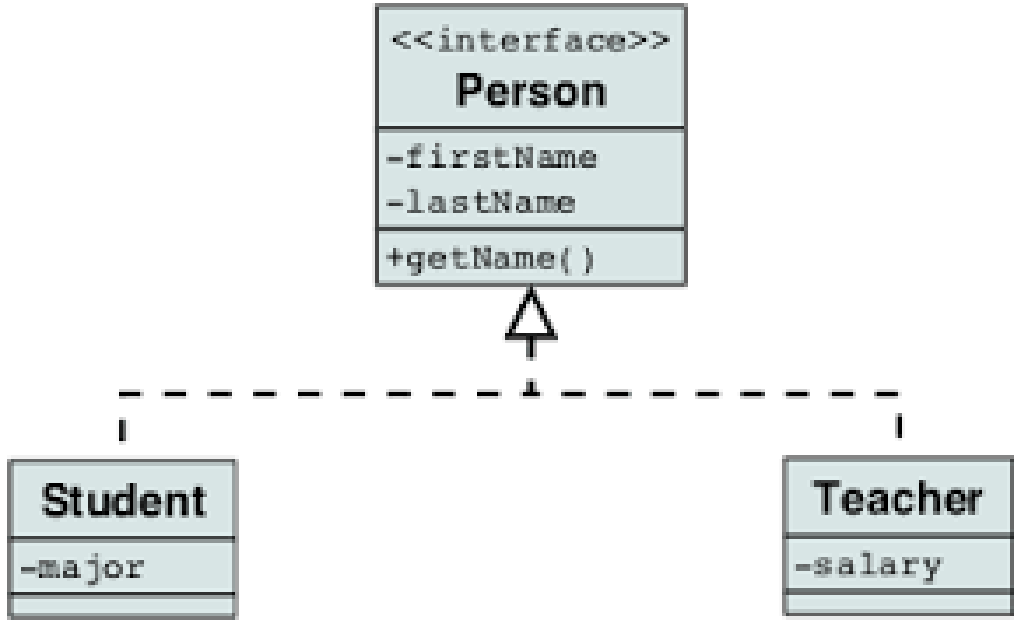




Intherince : Miras :

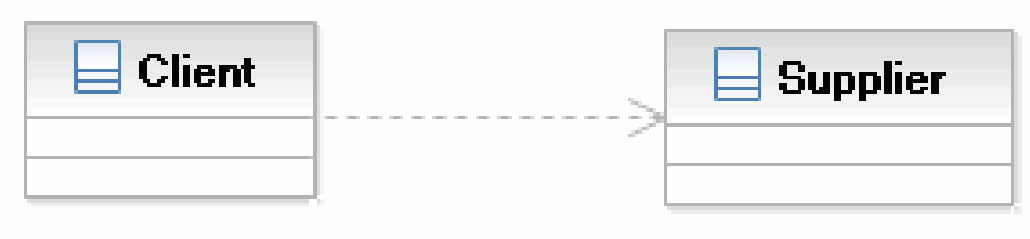
Realization (gerçekleme)

- Interface’in implemantasyonunu ifade eder.

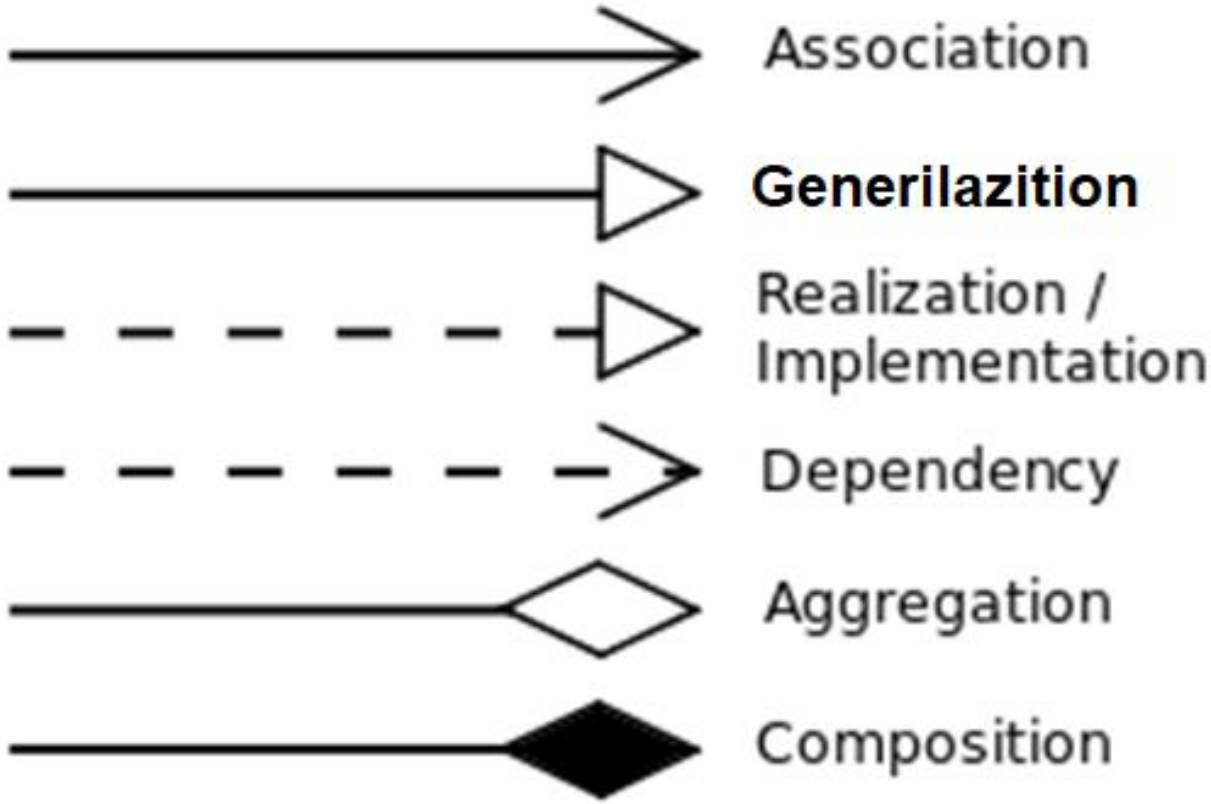


Dependency (bağımlılık)

- İki sınıf arasında bir bağımlılık olduğunu ifade eder
- Has-a ya da is-a ilişkisi yok
- Bağımlılık ilişkisi, client sınıfının aşağıdaki işlevlerden birini gerçekleştirdiğini gösterir:
  - Geçici olarak global kapsamı olan bir Supplier sınıfını kullanır
  - Geçici olarak bir Supplier sınıfını metodlarından biri için parametre olarak kullanır
  - Geçici olarak bir Supplier sınıfını kendi metodlarından biri için yerel değişken olarak kullanır
  - Supplier sınıfına bir mesaj gönderir

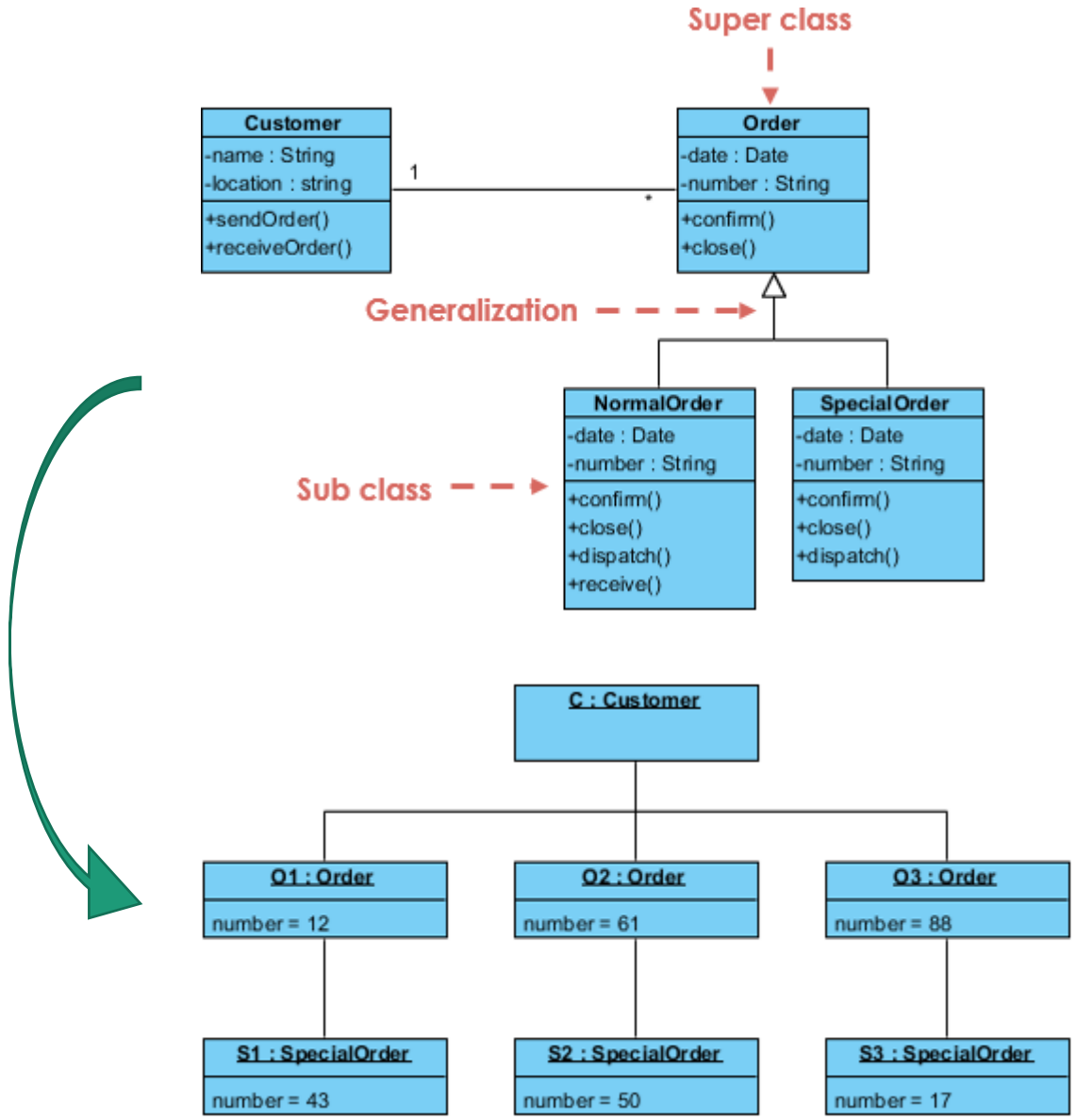


Sınıf ilişkileri için UML de kullanılan şekiller



Nesne (object) Diyagram

- Belirli bir anda oluşturulmuş nesneler ve ilişkilerini gösterir.
- Yazılım sisteminin bir andaki görüntüsünü vermektedir.



Araçlar

- Visual Paradigm
- Rational Rhapsody
- Enterprise Architect
- Eclipse Papyrus
- Lucidchart
- Visio
- Paint
- …