CLASS DİAGRAMS
(SINIF DİYAGRAMLARI)

Arş. Gör. Simge YILDIRIM

Sınıf(Class) Diyagramı

Sınıf aynı işlevlere ,aynı ilişkilere ve aynı anlama sahip nesneler topluluğunun ortak tanımıdır.

Gerçek hayattan alınan örneklere göre sınıflar, ona ait özellikler ve ifade edebileceği davranışlar belirlenir ve bunlar sınıf diyagramı olarak çizilir.

Sınıf diyagramları, doğrudan nesne yönelimli dillerle eşlenebilen tek UML diyagramı olduklarından, nesne yönelimli sistemlerin modellenmesinde yaygın olarak kullanılmaktadır.

Sınıf Diyagramlarının Amacı

- Bir sistemde ki sınıfların statik yapısını gösterir.
- Sınıfların öğelerini ve birbirleriyle olan ilişkilerini gösterir.
- Diyagram, UML tarafından öngörülen diğer yapı diyagramları için temel bir notasyon sağlar.
- Geliştiriciler ve diğer ekip üyeleri için de şablon görevi görür.
- İş Analistleri, sistemleri iş perspektifinden modellemek için sınıf diyagramlarını kullanır.

Class Diagram Gösterimi

Bir sınıf'ın (class) diyagram üzerinde gösterimi üç bölümden oluşur.

1. Class Name

Sınıfın adı ilk bölümde gösterilir.

2. Class Özellikler

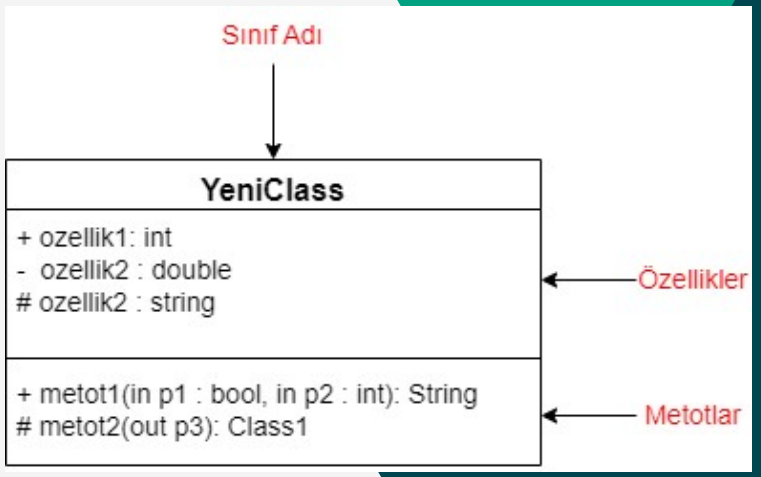
Özellikler ikinci bölümde gösterilir.

Erişim Belirliyici, Özellik adı, iki nokta (:), türü şeklinde belirtilir.

3 .Class Metotlar

Metotlar üçüncü bölümde gösterilir.

Erişim Belirliyici, Metot Adı, Parametre, İki nokta (:), dönüş tipi şeklinde belirtilir.



Class Diyagram Erişim Belirleyici Gösterimi

private : – «Sadece sınıfın içinde erişilebilir. Dışarıdan erişilemez. »

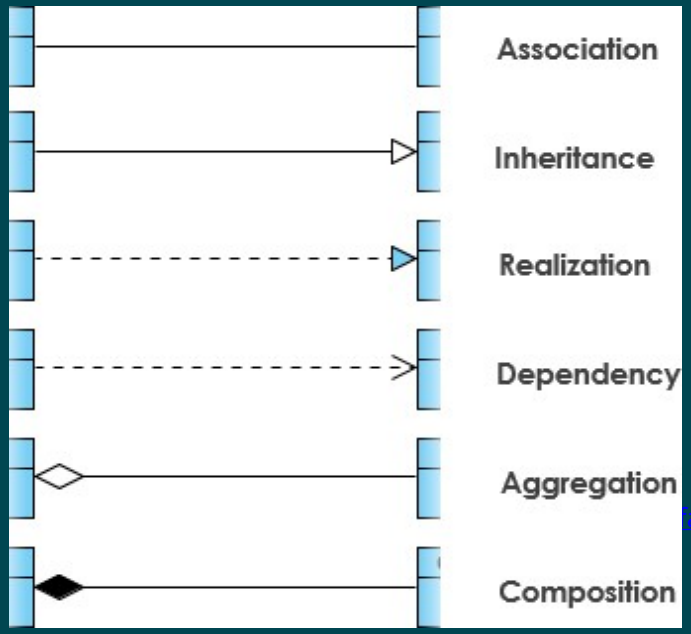
public : + «Her yerden erişilebilir. Sınıf dışından da kullanılabilir.»

protected : # «Yalnızca sınıfın kendisi ve ondan türeyen (kalıtım alan) alt sınıflar erişebilir.»

package : ~ «Aynı pakette (modülde) bulunan diğer sınıflar erişebilir; dışarıdan erişilemez.»

Class Diyagram İlişki Gösterimi

Bir sınıf, diğer sınıflarla bir veya daha fazla ilişki içinde olabilir. Bu ilişkiler aşağıda ki sembollerle gösterilir.

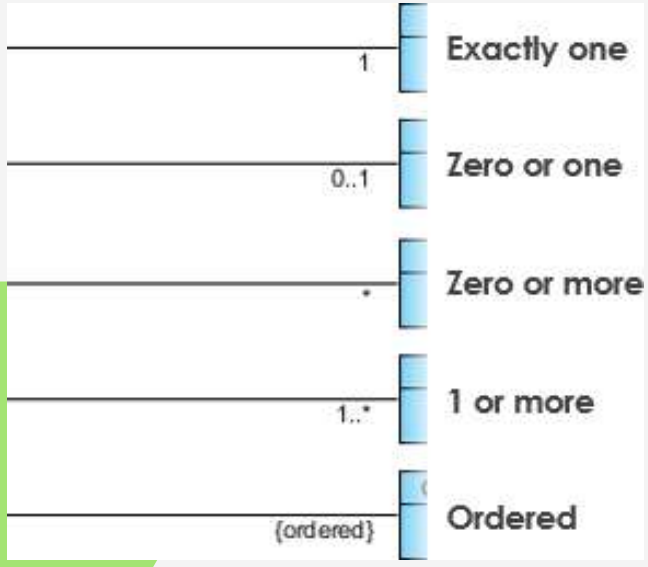


başına Geri Dön

Association (İlişki)

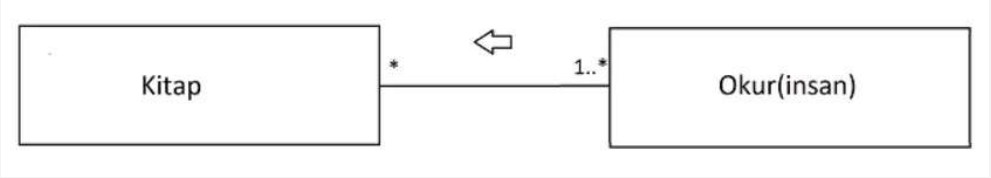
Bir UML Sınıf Diyagramındaki sınıflar arasındaki ilişkilerdir. Sınıflar arasında düz bir çizgi ile temsil edilirler.
En çok kullanılan ilişki türüdür. 2 nesne arasındaki ilişkiyi belirtir. Yön işareti koyarak bağıntının ne tarafa olduğunu gösterilebilir

.



- **one to one (bire bir)**
- **one to many (bire çok)**
- **many to many (çoka çok)**

Association (İlişki)



Buradaki (*) ve 1..* ilişki sayısını belirtmektedir. (*)bir kitabın herhangi bir okuyucu sayısı olabildiğini belirtmektedir 0 ya da 100 tane kişi o kitabı okuyabilir,(1..*) ise bir okuyucunun o kitabı en az bir kere okuduğunu göstermektedir, bunlar ile sınırlı mıdır tabii ki değil.

0 (hiç bir ilişki yok)
0..1 (hiç ilişki yok veya 1 tane ilişki var)
* (sıfır veya daha fazla ilişki)
1 (bir ilişki var)
1..* (birden fazla ilişki var)
1..1 (tam olarak bir tane var)
0..*(sıfır veya daha fazla ilişki)
.

Association (İlişki)

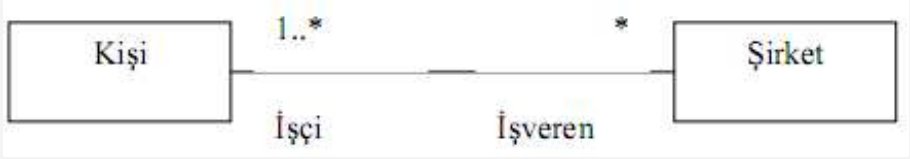
Bağıntı ilişkileri için tanımlanmış bilgiler aşağıdaki gibidir;

- Bağıntının Adı
- Sınıfın bağıntıdaki rolü
- Bağıntının çokluğu

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

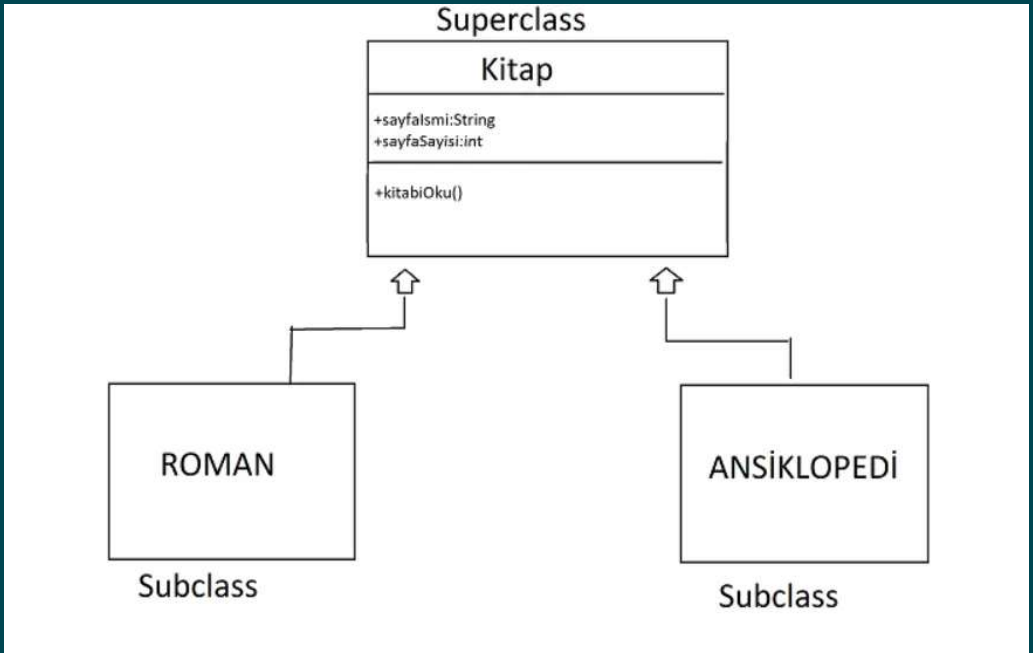
Bağıntı ilişkisinde bağıntı adı ve yönü

İşçi-işveren ilişkisinde bir şirketin en az bir işçisi olduğu (1..*), bir işçinin ise 0 ya da herhangi bir sayıda(*) şirkette işçi olarak çalışmış olabileceği ifade edilmektedir.



Inheritance

- Superclass(üst sınıf):Kalıtım alınan sınıf, alt sınıfı kapsar.
- Subclass(alt sınıf):Kalıtımı alan sınıf.
- Nesne tabanlı programlama dillerinin en temel yapılarından olan Inheritance (Kalıtım) yapılarının gösteriminde kullanılır. Base Class – Child Class ilişkisi. Abstract sınıflardan alınan kalıtım ilişkisi bu aynı gösterim şekline sahiptir.
- Bu tip kalıtım ilişkilerinde IS-A ilişkisi vardır. -> Eagle is a bird. – Kartal bir kuştur.
- Bu şekilde türetmenin yapıldığı sınıfları modellerken ok şeklinde çizgiler kullanılır. Okun ucunun olduğu taraftaki sınıf türetilen sınıftır, diğer taraftaki sınıf ise türeyen sınıftır.

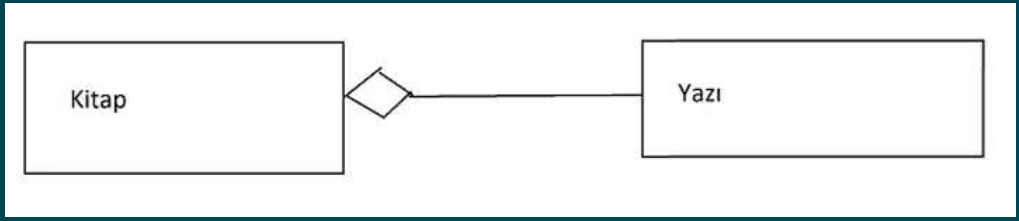


Dependency(Bağımlılık)

Bazen sınıfların işlevselliklerini yerine getirebilmeleri için dış kaynaklara ihtiyaçları olur. (Bir sınıfın diğer bir sınıfın bir parçası olduğu durumlar.) Bu gibi durumlarda modellemeyi ok çizerek yaparız ve okun ucu kullanılan sınıfı gösterir. Dependicy bağımlılık belirtmektedir. Bir sınıfın parçaları ile arasında olan bağımlılığı ifade eder. Dependicy 2'ye ayrılır.

Aggregation(İçerme)

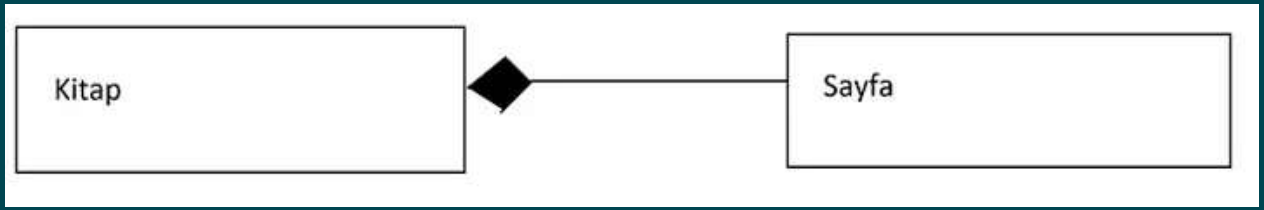
İçermelerde (Aggregation) parçalar arası hayati bir ilişki yoktur. Parçalar olmadan da sınıf varlığını devam ettirebilir. Yazı olmadan kitap olmaz ama kitap olmadığı durumda yazı varlığını sürdürmeye devam ettirecektir.



Dependency(Bağımlılık)

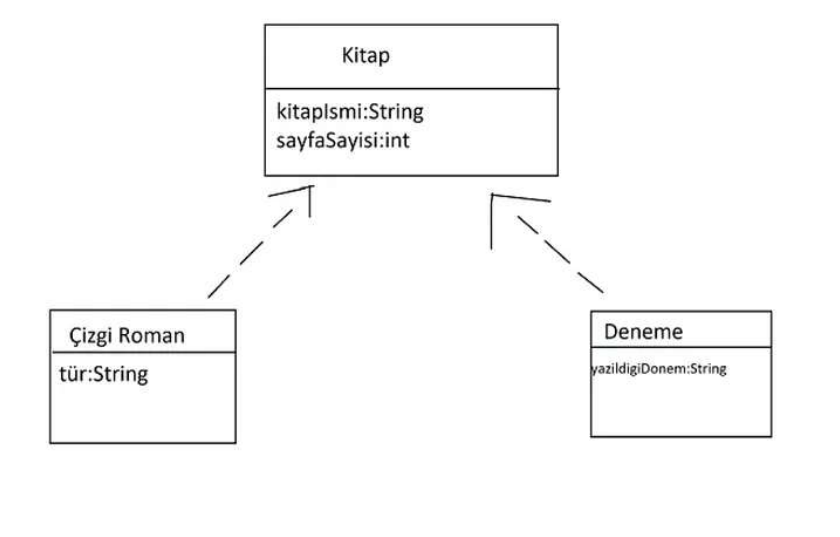
Composition(Oluşum)

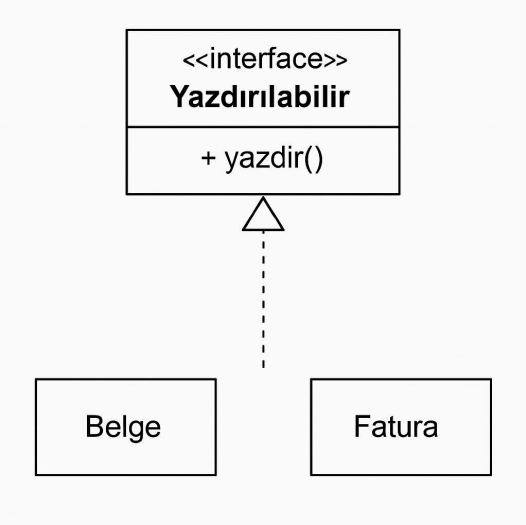
- Oluşumlarda (Composition) parçalar arası ilişki daha hayatidir, birinin varlığının sonlanması halinde diğerini de yanında götürecektir. Bir kitap sayfasız olamaz, sayfalar da kitapsız olamaz. Daha iyi bir örnek olması adına kalp ve insan arasındaki ilişkiyi de düşünebilirsiniz. Gösteriş şekilleri bakımından 2'si de baklava deseninde, aggregationun baklava desenin içi boşken compositionlarda içi doludur.



Realization(Gerçekleştirim)

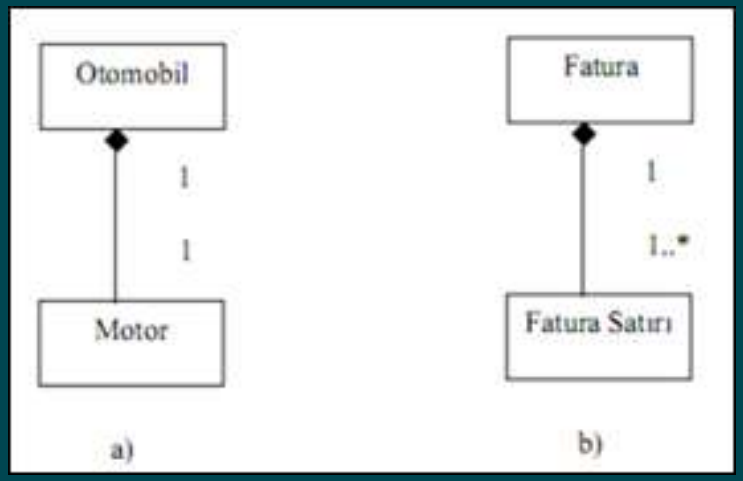
- Interface (arayüz)'ler ile sınıflar arasında ilişki modelini göstermek için kullanılır. Inheritance'da kullanılan çizginin kesik kesik olan haliyle temsil edilir. Sınıf, interface'teki metodları uygular.



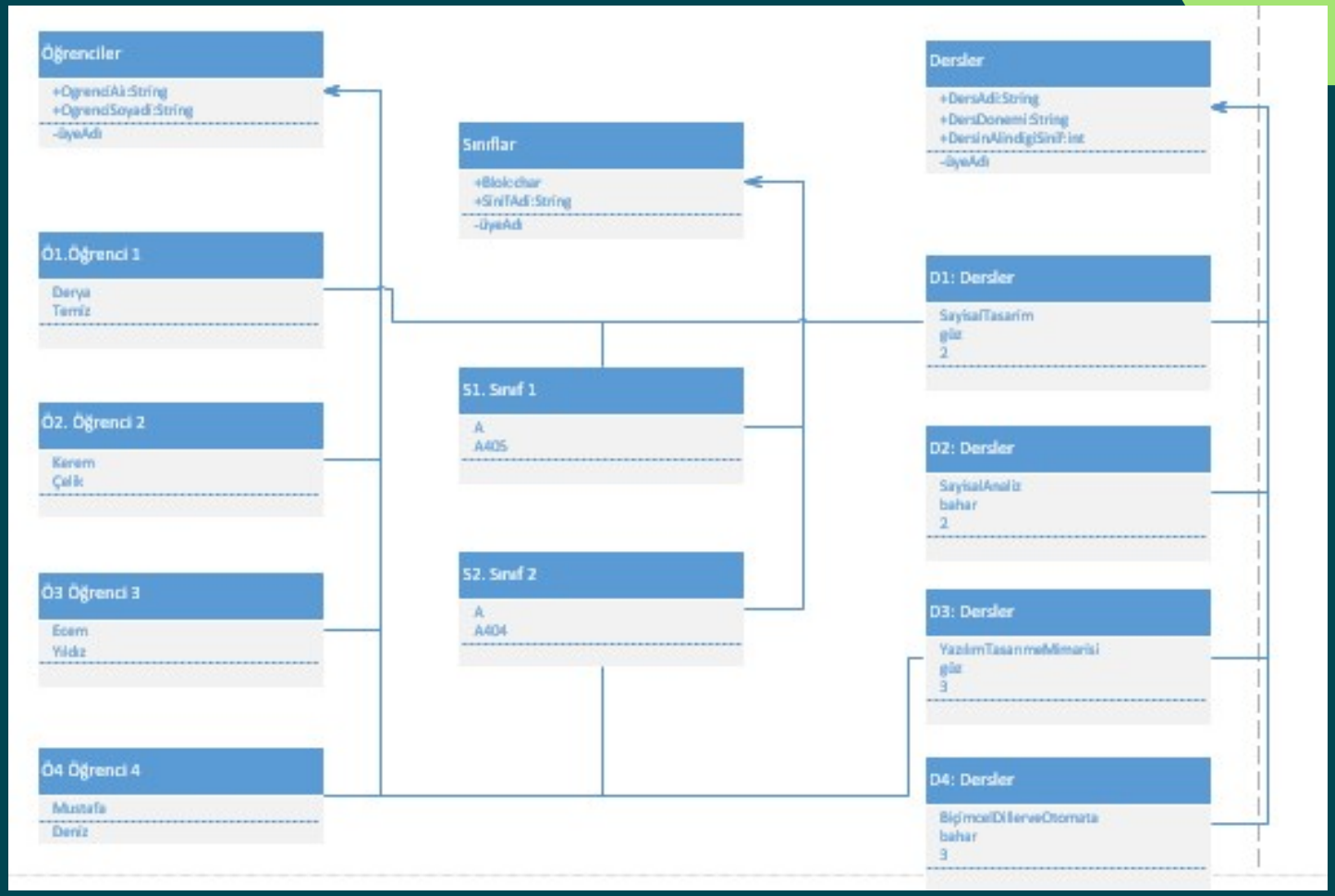


Aggregation (İçerme)– Composition(Oluşum)

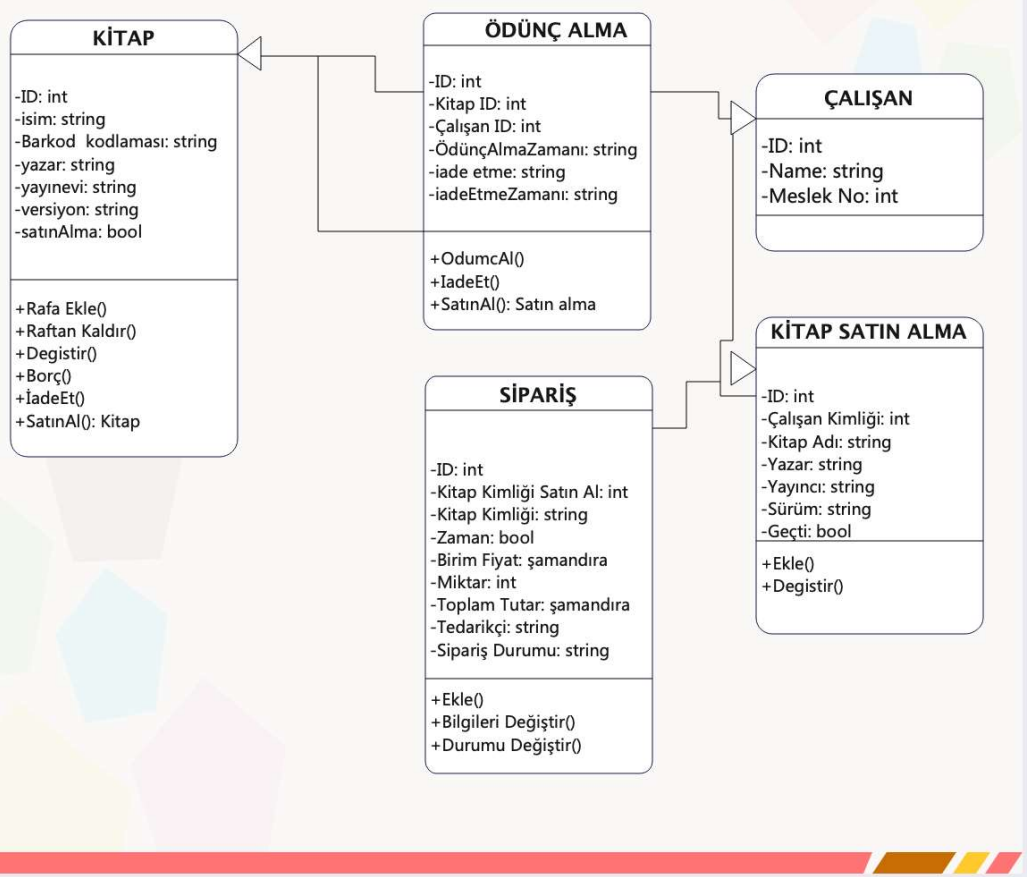
Aggregation ve Composition aslında bir dependency'dir. Bir sınıfın bir parçası olma durumu olarak düşünebiliriz. Oluşum bağıntısında her parça aynı anda sadece tek bir bütüne ait olabilir. Bu nedenle oluşum bağıntısında bütün tarafı çokluk değeri daima 1 olmak zorundadır. Oluşum ilişkisinde parçanın yaşam süresi doğrudan bütünün yaşam süresine bağlıdır. Bütün nesnesi yaratılmadan parça nesnesi yaratılamaz ve bütün tarafındaki nesne silindiğinde parça nesnelerde onunla birlikte silinmelidir.



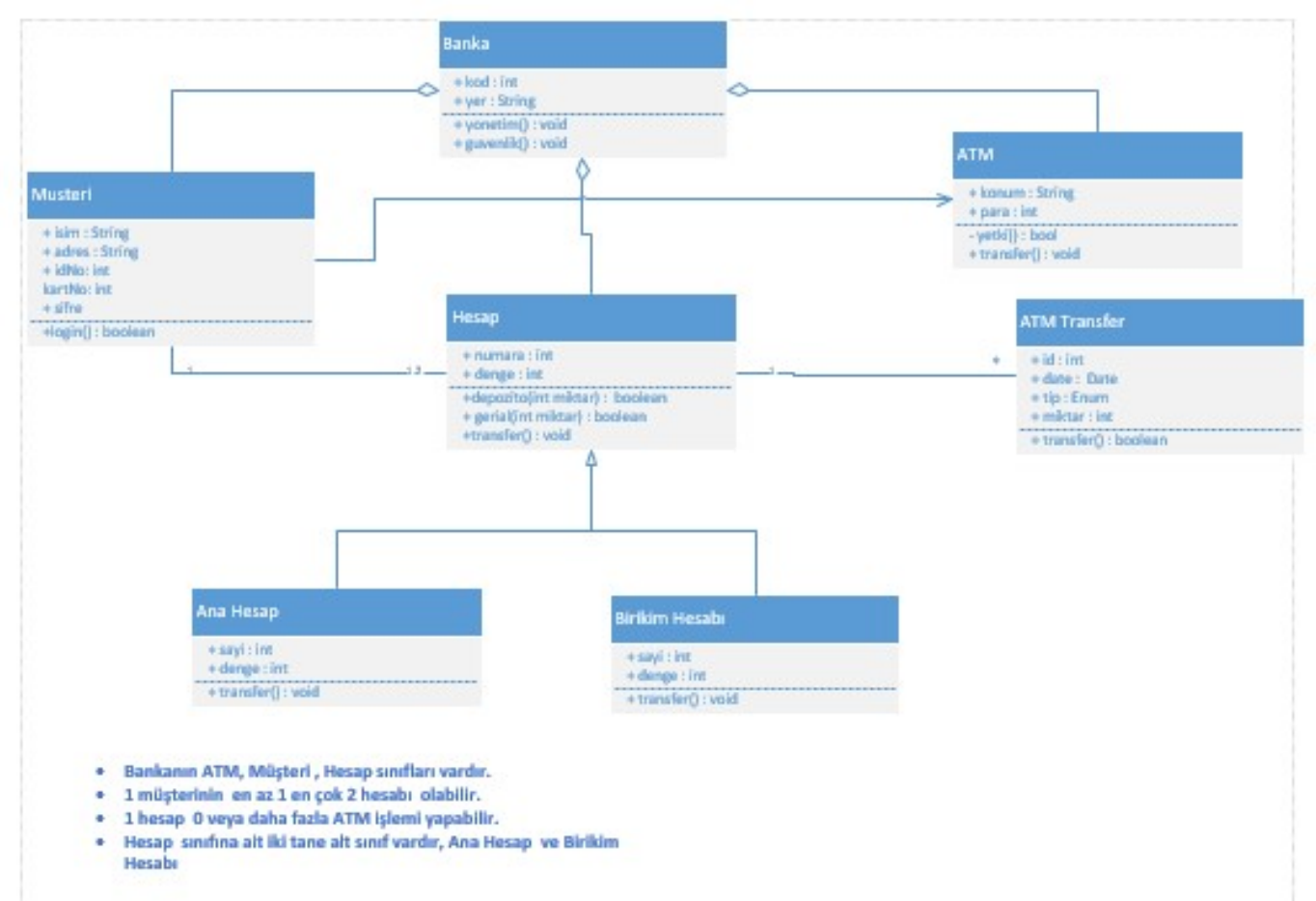
Class Diyagram Örnek



Ödünç Alma Sistemi Sınıf Diyagramı

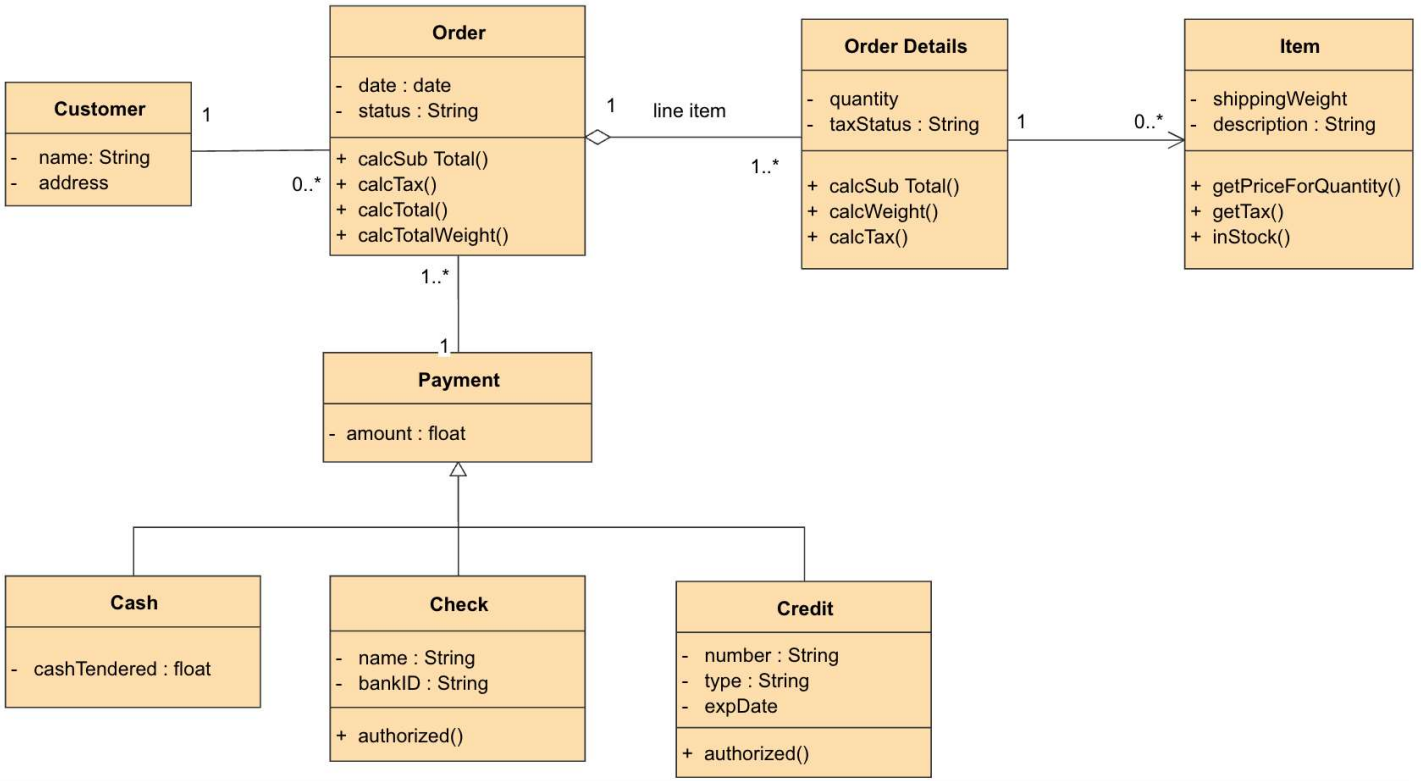


Banka Yönetim Sistemi Class Diyagram Örneği

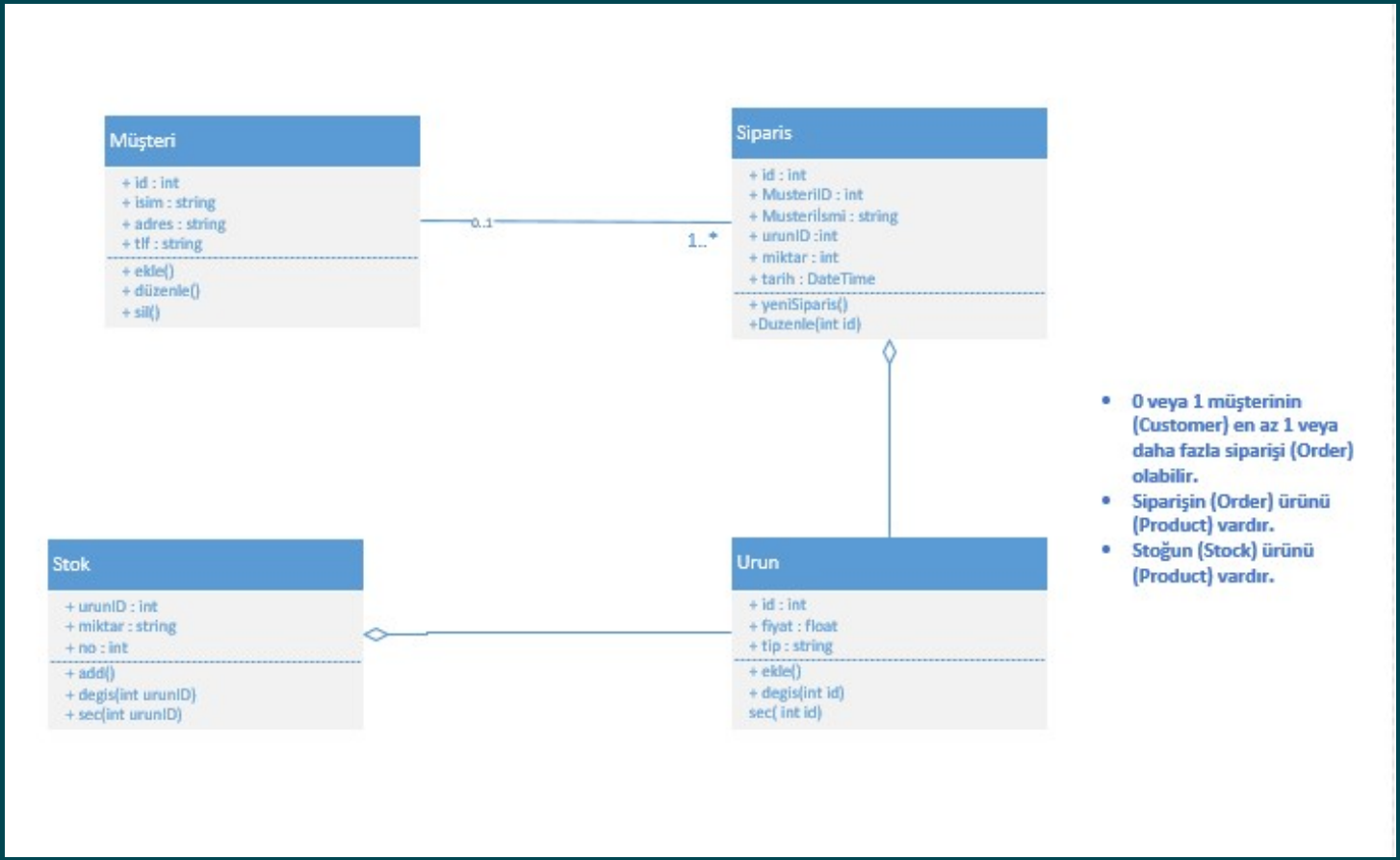


- Bankanın ATM, Müşteri , Hesap sınıfları vardır.
- 1 müşterinin en az 1 en çok 2 hesabı olabilir.
- 1 hesap 0 veya daha fazla ATM işlemi yapabilir.
- Hesap sınıfına ait iki tane alt sınıf vardır, Ana Hesap ve Birikim Hesabı

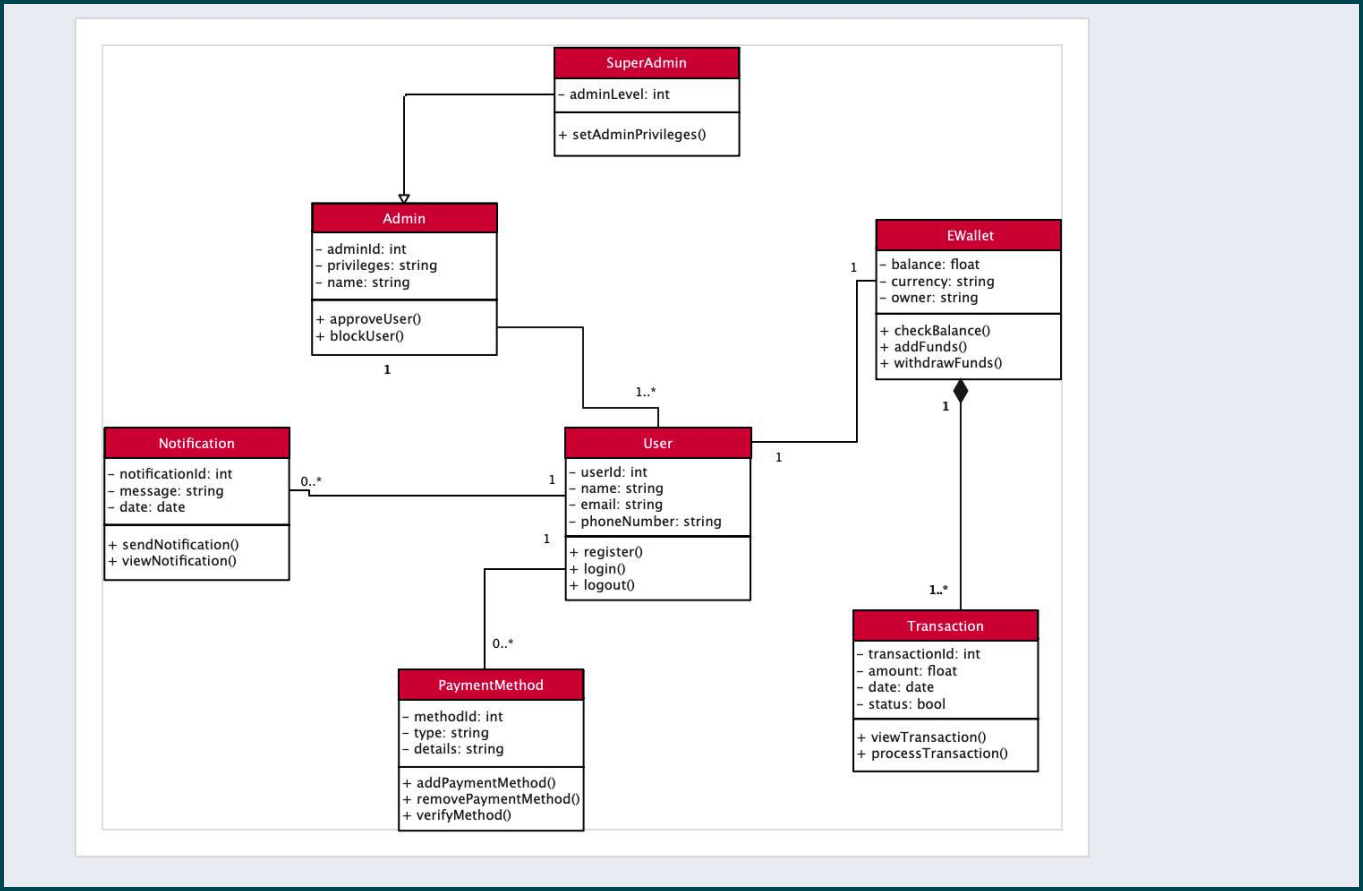
UML Class Diagram – Order System



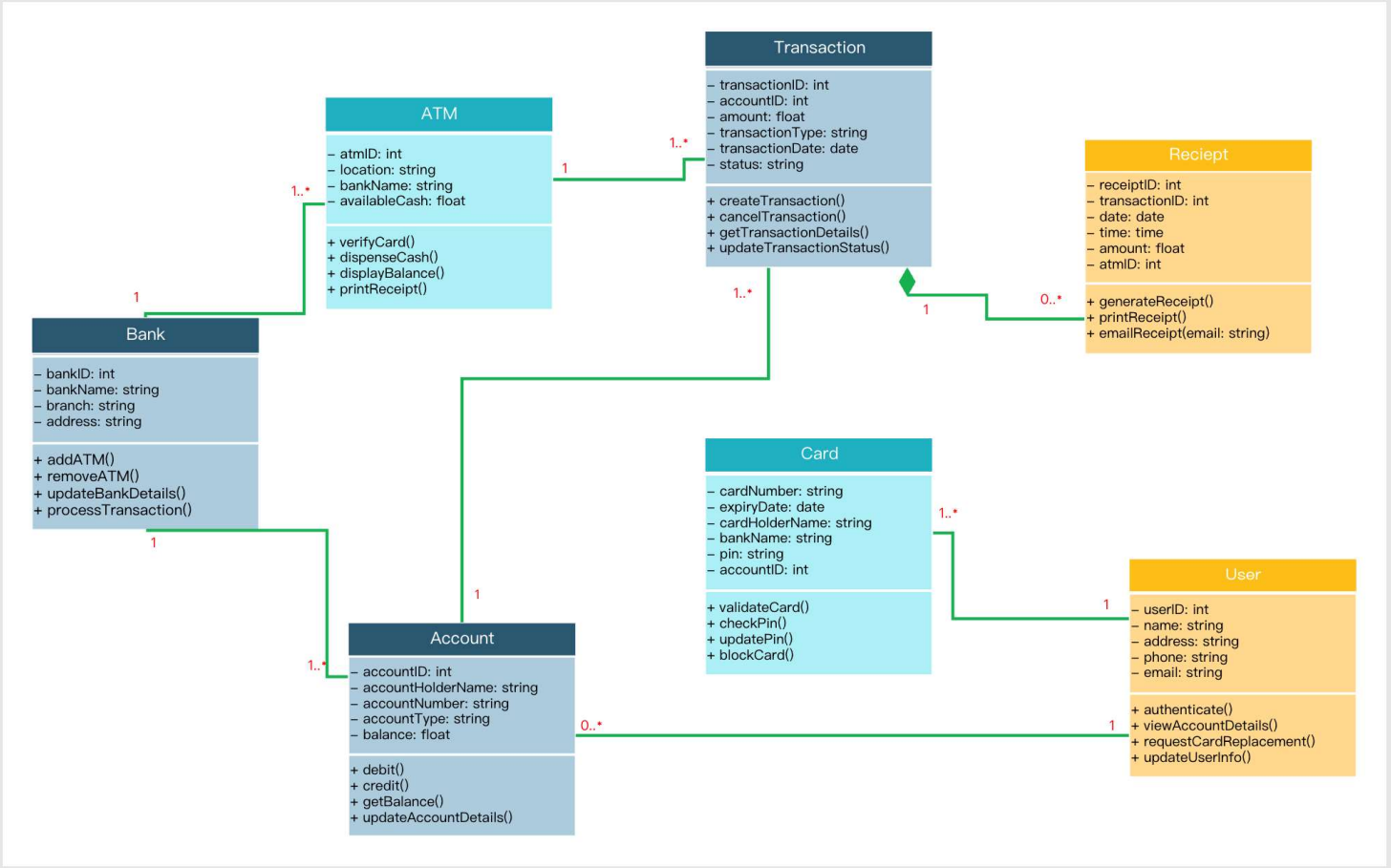
Sipariş İşlemleri Örnek



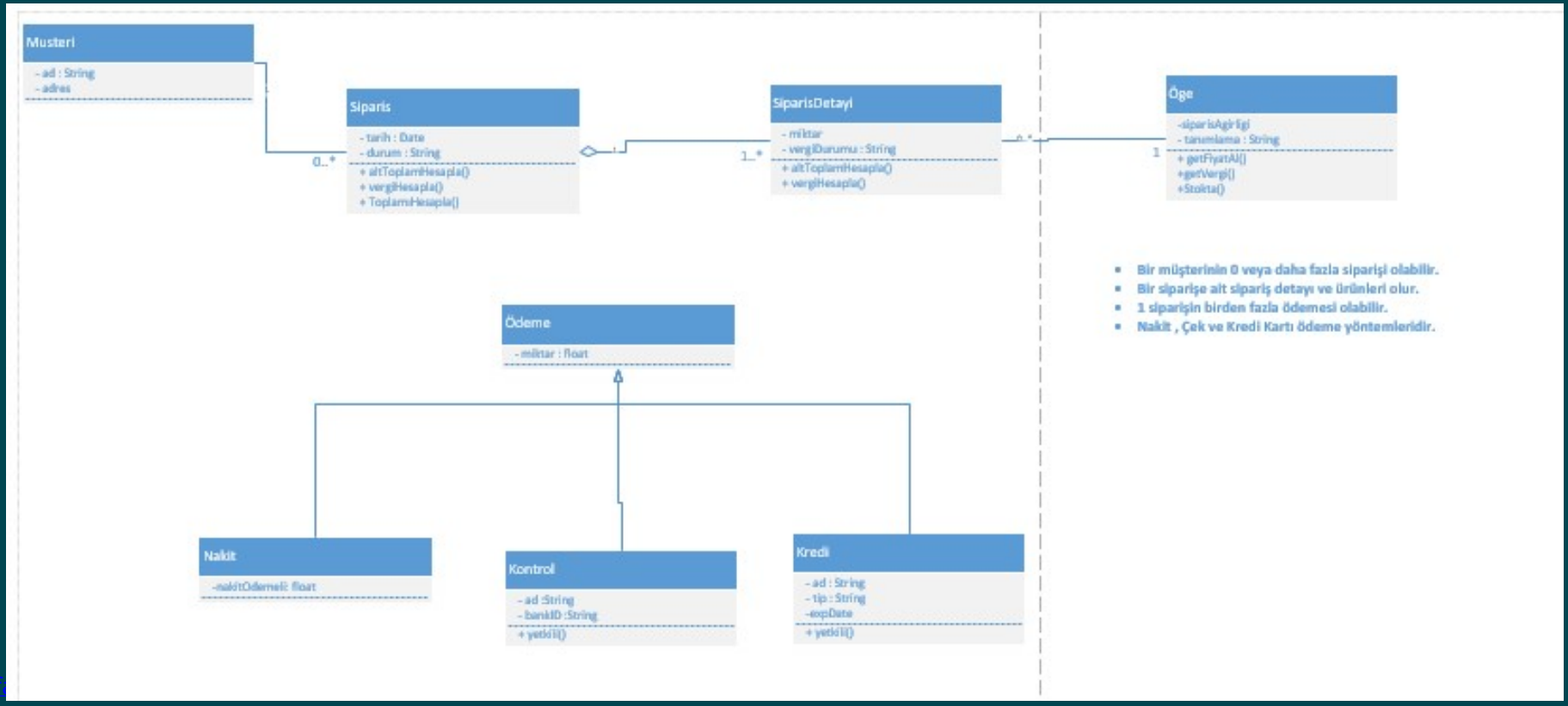
E-Bilet Sistemi Sınıf Diyagramı



ATM CLASS DİAGRAM



Sipariş Yönetim Sistemi Örnek



Gündem Sayfa

Online Sınav Sistemi Sınıf Diyagramı