

Nesne Yönelimli Analiz ve Tasarım

Nesnelerin Belirlenmesi

Sınıf Semasi

Nesneler Arası Bağıntılar

Modelin Koda Dönüşürülmesi

Örnek Uygulamalar (Veri Toplulukları İle İşlemler),
Örnekler (Sıralama Semasi, Etkinlik Semasi, Durum Semasi)

Nesne Yönelimli Analiz

- * Analiz aşamasının ilk adımı gereksinimlerin belirlenmesidir.
- * Gereksinimler, sistemin düzgün çalışması için yapması beklenenleri ifade eder
- * Gereksinimler ilk olarak kullanıcılarla(paydaşlarla) görüşülerek (gerekli ise standart, yasa, yönetmelik vb. dokümanlar takip edilerek) oluşturulur, ve neler istediği belirlenir.
- * Daha sonra kullanım durumları (use case) belirlenerek gereksinimler ayrıntılandırılır.
- * Sistemin tüm istenenlerle birlikte, beklenmeyen ya da istenmeyen durumları da göz önüne alınır.
- * Kullanım durumları analiz edilerek nesneler, nesne üyeleri ve nesneler arasındaki bağıntılar belirlenir.

Nesnelerin Belirlemelesi

- * NYP gerçek dünyayı daha doğru modelleyebilmek için geliştirilmiştir.
- * Gerçek hayatı gevremizde gördüğümüz her şey nesnedir.
 - * insan, ürün, sipariş, fatura, öğrenci, ekran, televizyon, masa, sıra, bisiklet, araba, köpek v.s.
- * Nesnelerin durum ve davranışları bulunur.
 - * İnsan için durumlar; adı, yaşı, boyu v.s. iken davranışlar; öğrenmek, anlamak, uyumak, konuşmak, koşmak v.s.
 - * Bisiklet için durumlar; rengi, o anki vitesi, hızı, tekerlek sayısı, vites sayısı v.s. iken davranışlar; fren yapması, hızlanması, yavaşlaması, vites değiştirmesi v.s.
- * Yazılım nesneleri de durum ve davranışlara sahiptir.
 - * Nesnelerin durumları nitelik (özellik) olarak da adlandırılır ve üye değişkenler ile ifade edilir.
 - * Davanışlar ise üye fonksiyonlar/methodlar/yöntemler ile temsil edilir.

Nesnelerin Belirlenmesi

- * Gereksinimlerden nesnelerin belirlenmesi işlemi için; nesne modelleme, alan modelleme, kavramsal modelleme gibi isimler kullanılır.
- * Nesneler belirlenirken, analiz aşamasında oluşturulan gereksinim Listesi (kullanım durumlarının ayrıntıları) üzerinde metinsel analiz yapılır (Abbott's teknigi).
- * Gereksinim Listesindeki isim ve isim tamlamaları sınıf adaylarıdır. Özellikleri ya da davranışları bulunan isim ya da isim tamlamaları sınıf olabilir.
- * Gereksinim Listesindeki eylemler nesnelerin üye yöntemlerini ifade edebilir.
- * Bu işlem sadece bir başlangıçtır ve yinelemeli bir süreçtir.
- * Gereksinim Listesi inceledikçe yeni sınıflar, özellikler, yöntemler, sınıflar arası bağlantılar belirlenir. Sınıflar değiştirilebilir, niteliğe dönüşebilir, kullanılmayabilir. Nitelikler sınıfa dönüşebilir vb.
- * Sınıflar belirlendikten sonra tasarım ayrıntıları içermeyen Analiz seviyesi Sınıf Şeması (kavramsal model) oluşturulur.

Nesnelerin Belirlenmesi

Nesne adayları (Craig Larman):

- * Fiziksel ya da elle tutulabilir nesneler (ürün, insan, bilgisayar, klavye, ekran ...)
- * Yerler (okul, bina, yerleşke, sınıf, oda ...)
- * İşlemler (transactions) (para çekme..., kayıt yaptırma...)
- * Roller (yönetici, öğrenci, personel, kayıtlı kullanıcı v.b.)
- * Harici sistemler (Veritabanları, sürücüler, banka bilgi sistemi, web servisleri...)
- * Kuruluşlar (okul, işyeri, firma...)
- * Olaylar (ActionListener, ActionEvent, KeyListener, KeyEvent...)
- * Loglar ...

Nesnelerin doğru olarak belirlenmesi için en etkili yol paydaşlarla görüşmektir (beyin fırtınası).

ATM Sistemindeki Nesnelerin Belirlenmesi

* ATM sistemi aşağıdaki donanım bileşenlerinden oluşur:

- * Tuş takımı; ile veri girişi yapılır.
- * Ekrان; kullanıcıya mesaj görüntüler.
- * Para bölmesi; para alma ve para verme işleminden sorumludur.
- * Kart bölmesi; banka kartını alır ve doğrular.

* Müsteriler hesaplarından banka bilgi sistemini kullanarak; para çekme, para yatırma, bakiye görüntüleme vb. işlemlerini yapabilmelidirler.

* Banka görevlileri ATM sisteminin bakımını (para doldurma, sıkışan/yutulan kartları alma v.b.) yapmalıdır.

* Tüm işlemlerde güvenlik gereksinimleri göz önünde bulundurulmalıdır (yetki kontrolü).

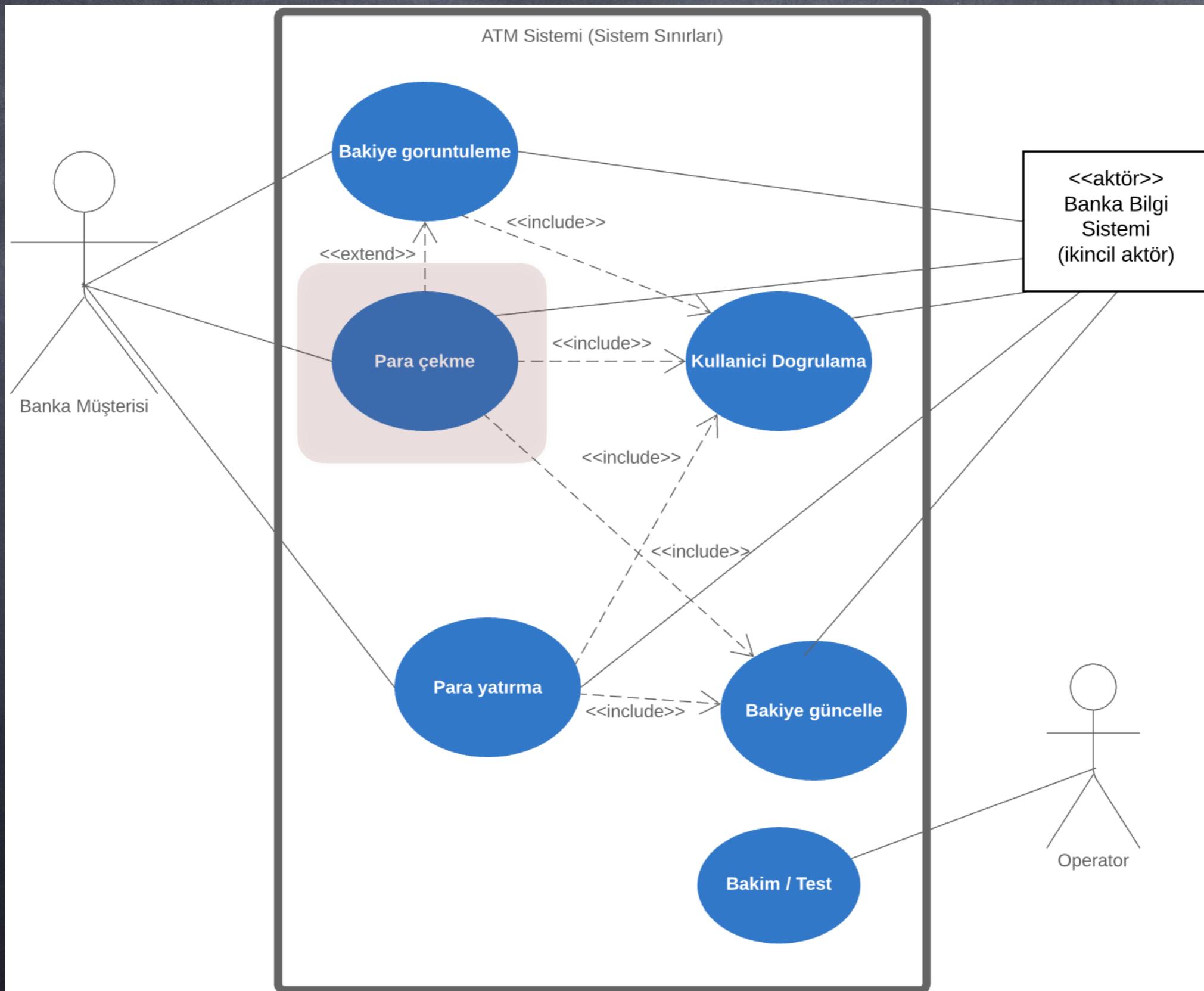
* Sistem, Banka Bilgi Sistemiyle gevrimiçi çalışmalıdır. Kullanıcılar Banka Bilgi Sisteminde yer alan hesaplarına erişerek işlemlerini gerçekleştirirler.

Not: Sarı renkliler nesne (sınıf), yeşil renkliler yöntem adaylarıdır.

Aynı cümlede bulunan nesneler bağıntılıdır.

ATM Sistemindeki Nesnelerin Belirlenmesi

Para Çekme Kullanım Durumu (Use Case)



ATM Sistemindeki Nesnelerin Belirlenmesi

Use Case- "Para Çekme" Olay Akışı (Ana senaryo-başarılı)

1. ATM sistemi Ekrana, müşteriden "kartını kart bölmeye takmasını" isteyen mesaj yazdırır.
2. Kart sahibi kartını kart bölmeye takar.
3. ATM sistemi kart doğrulamasını yapar.
4. Ekrana şifre girilmesini isteyen mesaj gönderilir.
5. Tuş takımını kullanarak girilen şifreyi alır.
6. Kullanıcı doğrulama ve yetki kontrolü için banka bilgi sistemine istek gönderilir.
7. Banka sistemi erişim isteğini kabul eder ve gerekli bilgileri gönderir.
8. ATM sistemi ekrana, çekilecek tutarları listeler.
9. Kart sahibi çekilecek tutarı girer.
10. ATM sistemi kartı çıkartır.
11. Kart sahibi kartı alır.
12. ATM sistemi parayı ve makbuzu verir.
13. Kart sahibi parayı ve makbuzu alır.

Not: Kırmızı renkli işlem basamakları Kullanıcı Doğrulama (erişim denetimi) kullanım durumunu ifade eder.

ATM Sistemindeki Nesnelerin Belirlenmesi

"Para Çekme" Kullanım Durumu Olay Akışı (Alternatif akışlar)

A1. Kart doğrulanamadı (3)

4. Kart bölmesi kartı çıkartır.

5. İşlem sonlandırılır.

A2. Yanlış şifre (6)

8. 3 den az kez yanlış ise yeniden gir.

8. 3 kez yanlış girilmiş ise kart yükülür.

9. İşlem sonlandırılır.

A3. Yeterli para yok (9)

10. ATM de yeterli bakiye yoksa yeni kutar girmesi istenir.

11. Hesapta yeterli bakiye yoksa yeni kutar girmesi istenir.

A4. Banka kartı alınmadı(10)

11. 30 sn. boyunca alınmadı ise kart yükülür

12. Gerekli bilgilendirme yapılır

13. İşlem sonlandırılır

A5. Para alınmadı (12)

13. 30 sn. boyunca alınmadı ise para yükülür

14. Gönderilen ve yükulan para karşılaştırılır.

15. Gerekli ise hesap güncellenmesi yapılır

12. Gerekli bilgilendirme yapılır

13. İşlem sonlandırılır

...

UML - Sınıf Diyagramı

- * Yapısal gösterim şekillerindendir.
- * Tasarlanan sistemin sınıflar ve arayüzler seviyesinde statik yapısını gösterir.
- * Sınıf ve arayüzlerin, özelliklerini, üyelerini, kısıtlarını ve bağıntılarını ığırır.

GeometrikSekil

GeometrikSekil sınıfının basit gösterim

GeometrikSekil

x:double
y:double
renk:String

konumDegistir()
alanHesapla()
toString()

GeometrikSekil sınıfının analiz seviyesi gösterimi

GeometrikSekil

- x:double
- y:double
- renk:String="Beyaz"
+ toplamSekilSayisi: int

+ konumDegistir(double,double):void
+ alanHesapla() : double
+ toString() : String

+ : public
- : private
: protected
altı çizili: statik üye

GeometrikSekil sınıfının gerçekleme seviyesi gösterimi

GeometrikSekil

private
x:double
y:double
renk:String="Beyaz"

public
konumDegistir(double,double):void
alanHesapla() : double
toString() : String

GeometrikSekil sınıfının gerçekleme seviyesi gösterimi

GeometrikSekil

- x:double
- y:double
- renk:String="Beyaz"
+ toplamSekilSayisi: int

+ konumDegistir(double,double):void
+ alanHesapla()
+ toString()

Soyut (Abstract) Sınıf

<<interface>> Sekil

+ konumDegistir(double,double):void
+ alanHesapla()

Arayüz (Interface)

Modelin Koda Dönüşürülmesi / Sınıf

```
package cc.ders6;

public class GeometrikSekil {

    private double x;
    private double y;
    private String renk;
    public static int toplamSekilSayisi;

    public void konumDegistir(double x, double y){
        ....
    }

    ....
}

}
```

GeometrikSekil

- x:double
- y:double
- renk:String="Beyaz"
+ toplamSekilSayisi: int

+ konumDegistir(double,double):void
+ alanHesapla() : double
+ toString() : String

+ : public
- : private
: protected
altı çizili: statik üye

GeometrikSekil sınıfının
gerçekleme seviyesi gösterimi

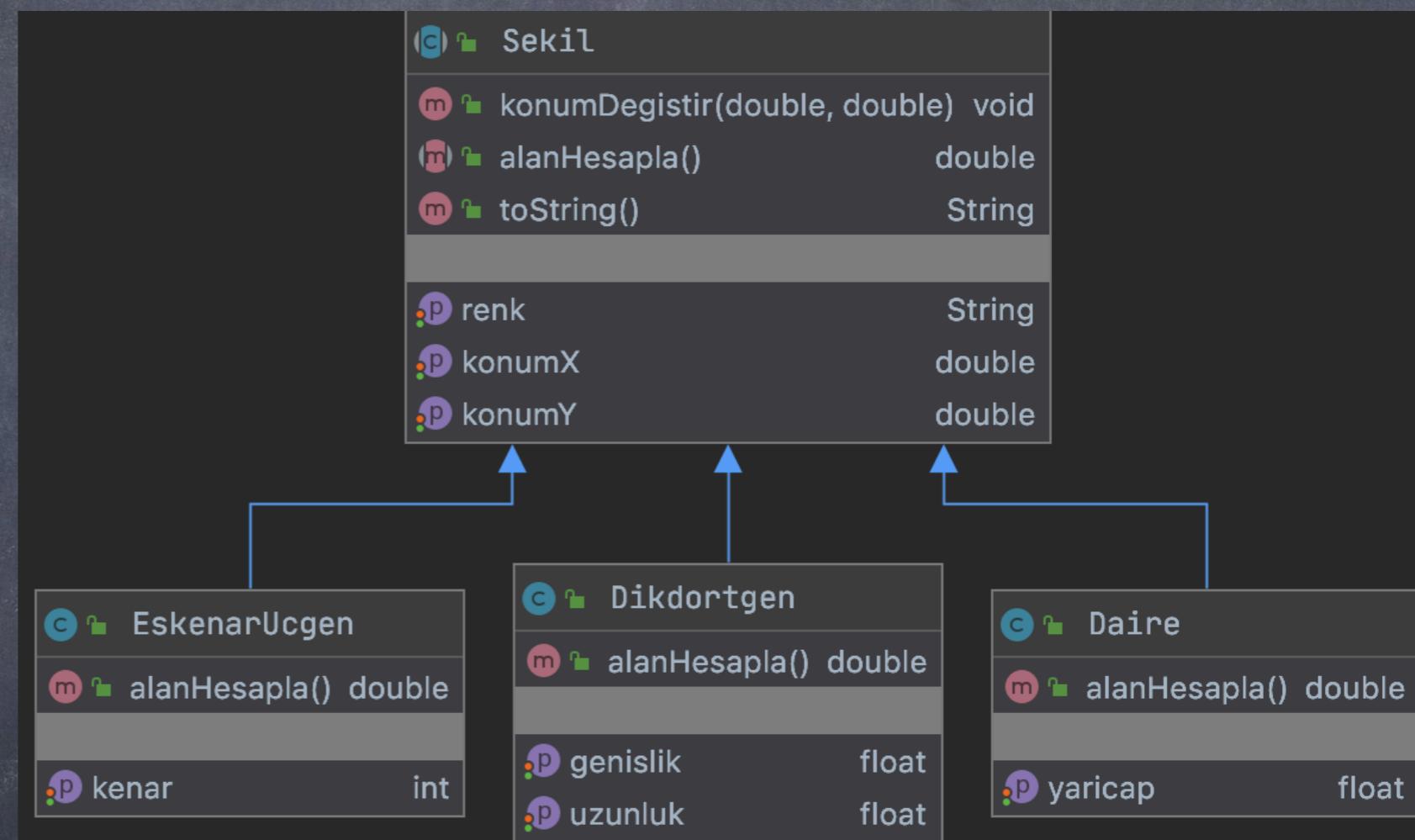
Veri Topluluklarıyla İşlemeler

[https://github.com/celalceken/NesneYonelimiAnalizVeTasarimDersiUygulamalari/tree/master/Ders4/
VeriTopluluklariUzerindeIslemler](https://github.com/celalceken/NesneYonelimiAnalizVeTasarimDersiUygulamalari/tree/master/Ders4/VeriTopluluklariUzerindeIslemler)

Sınıf Diyagramı - Elementler arası bağlantılar

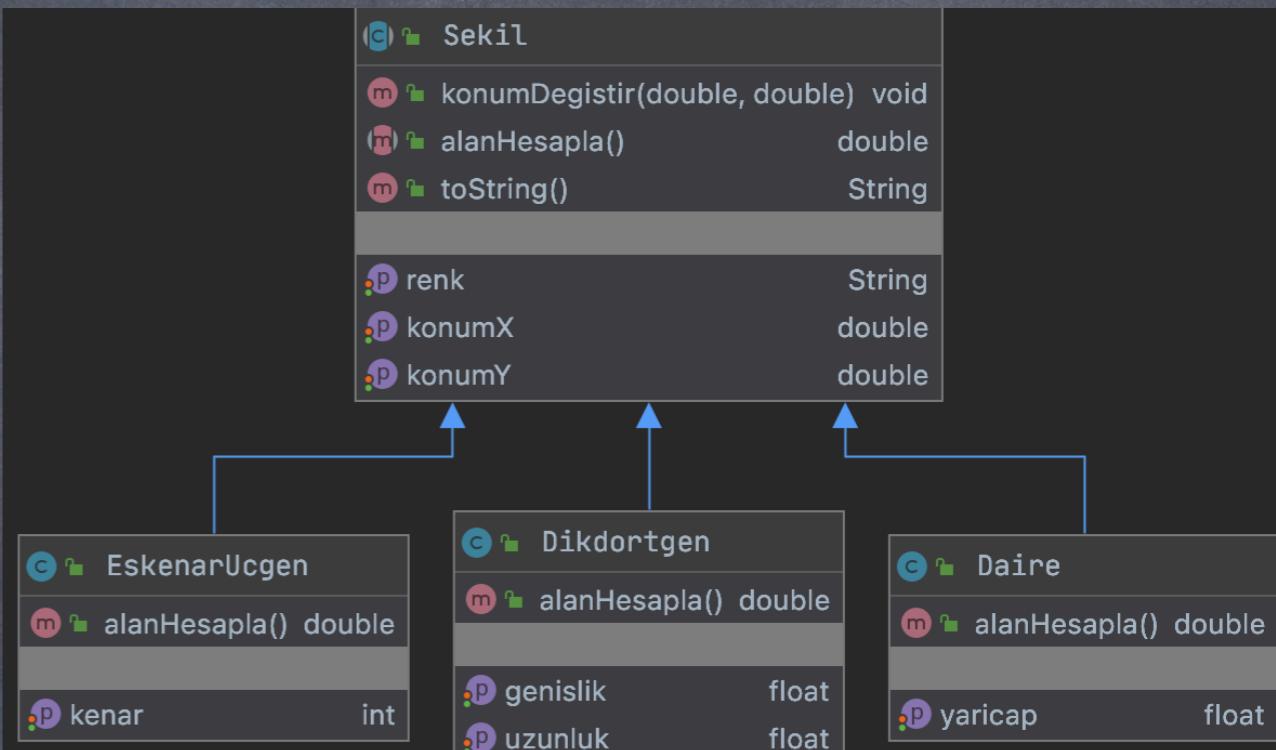
Kalıtım Bağlığı

- * Benzer varlıklar arasında olmalıdır.
- * "is a", "is kind of"
- * Türetilmiş sınıf (Derived class) Temel Sınıfin (base class) üyelerine kalıtım yoluyla sahip olur.
- * Kod tekrar kullanımı sağlanır



Sınıf Diyagramı - Elementler arası bağlantılar

```
abstract public class Sekil { //Soyut yöntem sadece soyut sınıflar  
    private double konumX;  
    private double konumY;  
    private String renk;  
  
    public Sekil(double konumX, double konumY, String renk) {  
        this.konumX = konumX;  
        this.konumY = konumY;  
        this.renk = renk;  
    }  
  
    public double getKonumX() { return konumX; }  
  
    public void setKonumX(double konumX) { this.konumX = konumX; }
```



```
public class Daire extends Sekil {  
  
    private float yaricap;  
  
    public Daire(double konumX, double konumY, String renk, float yaricap) {  
        super(konumX, konumY, renk);  
        this.yaricap = yaricap;  
    }  
  
    public float getYaricap() { return yaricap; }  
  
    public void setYaricap(float varicap) { this.varicap = varicap; }
```

Sınıf Diyagramı - Elementler arası bağlantılar

"Dependency" Bağlığı

- * Bir elementin (istemci- ClassA) diğer elementi (tedarikçi-ClassB) kullanımını ifade eder.
 - * "uses"
 - * ClassA içerisindeki bir yöntem ClassB yi kullanır.
- SınıfA

----->

SınıfB
- C Fatura

m toplamTutariHesapla() double

m toString() String

p siparis Siparis

C FaturaServisi

m faturaOlustur(Fatura) String
- ```
public class FaturaServisi {

 public String faturaOlustur(Fatura fatura){

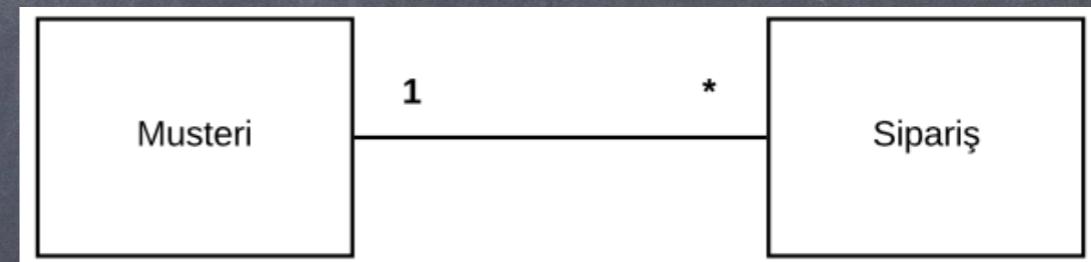
 String faturaXML=<XML>"+fatura.toString();
 return faturaXML;
 }

}
```

# Sınıf Diyagramı - Elementler arası bağlantılar

## "Association" Bağlığı

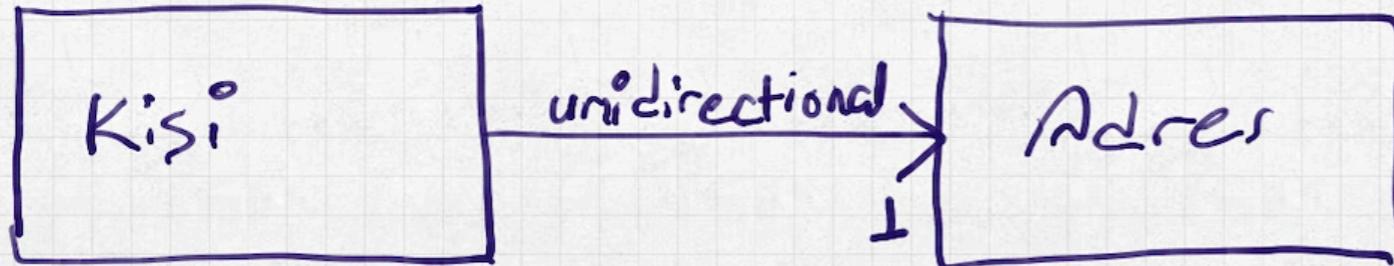
- \* Bir nesne diğer nesneyi üye (özellik) olarak icerir.
- \* "has a"
- \* "Bidirectional" (iki yönlü) ve unidirectional (tek yönlü) olabilir.
- \* "Aggregation" ve "Composition" bağlantıları "Association" bağlantısının özel halleridir.



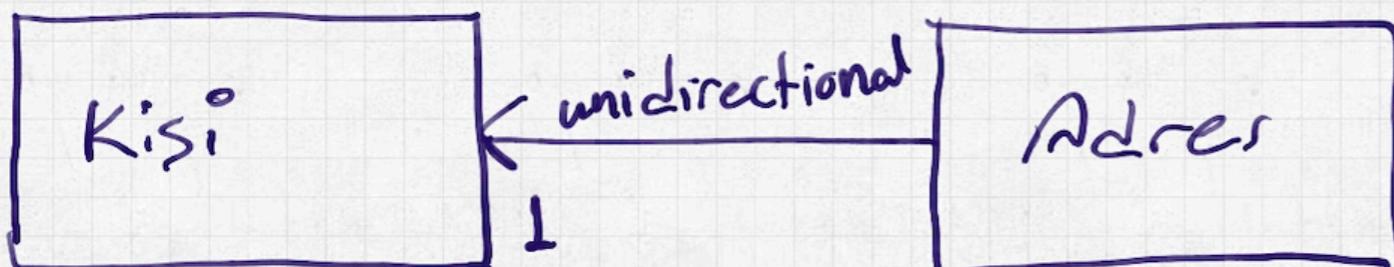
- \* 1 Musteri nesnesi çok sayıda Siparis nesnesini (referansını) icerir.

# Sınıf Diyagramı - Elementler arası bağlantılar

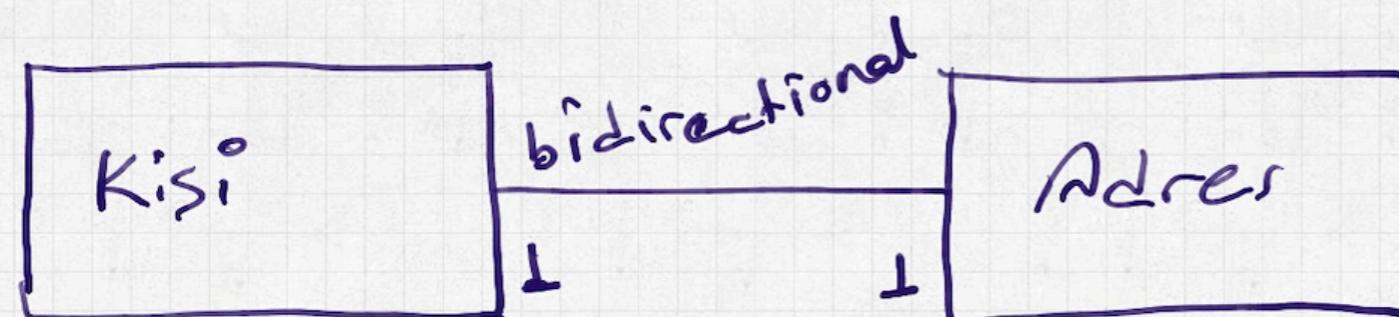
"Association" Bağlığı



```
class Kisi
{ private Adres adres;
};
```



```
class Adres
{ private Kisi kisi;
};
```

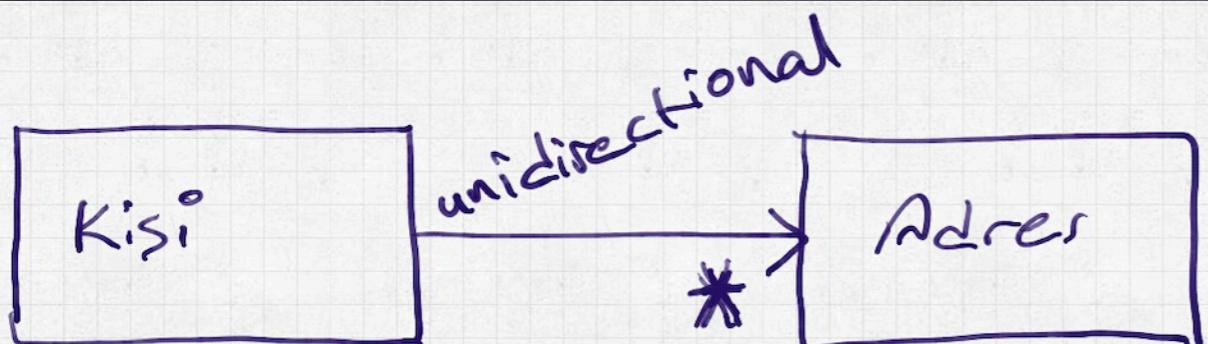


```
class Kisi
{ private Adres adres;
};

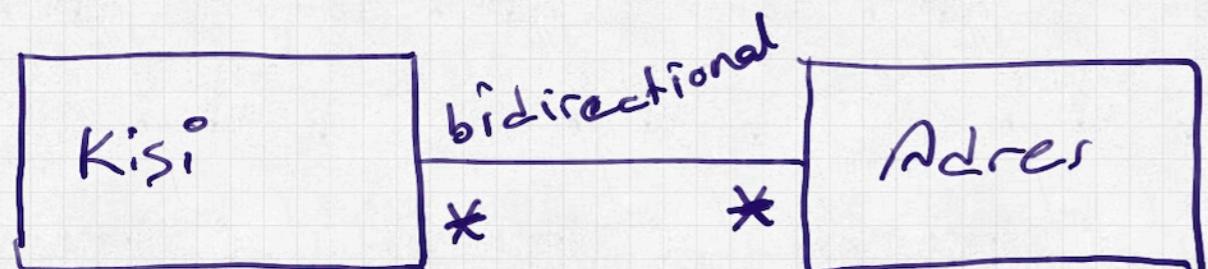
class Adres
{ private Kisi kisi;
};
```

# Sınıf Diyagramı - Elementler arası bağlantılar

"Association" Bağlığı



```
class Kisi
{
 private List<Adres> adresler;
```



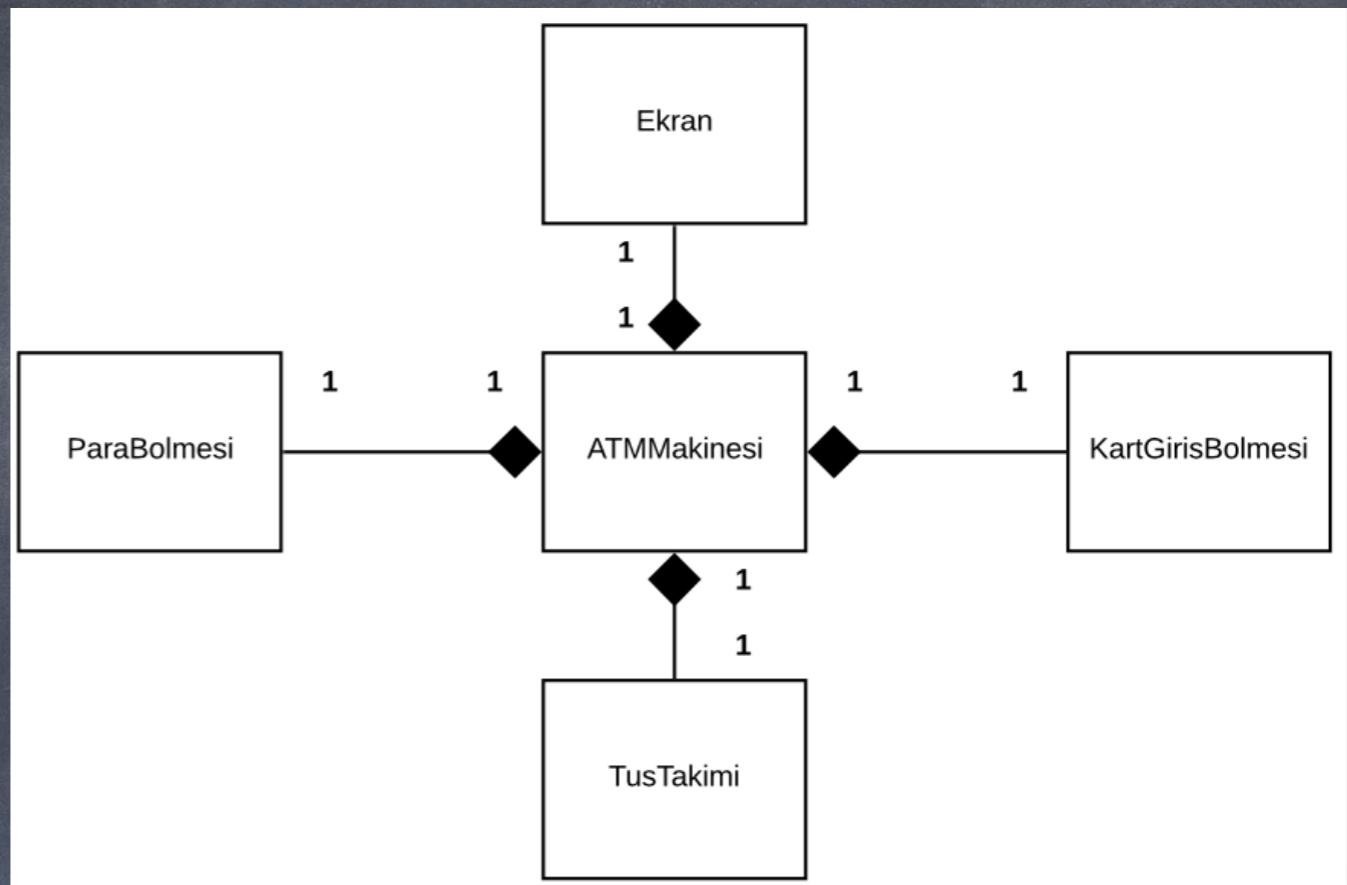
```
class Kisi
{
 private List<Adres> adresler;
}

class Adres
{
 private List<Kisi> kisiler;
}
```

# Sınıf Diyagramı - Elementler arası bağıntılar

## "Composition" Bağıntısı

- \* Bütün parça bağıntısı (whole/part)
- \* Bütün, işini yapabilmek için parçaya ihtiyaç duyar.
- \* Parça nesneler bütün içerisinde yer alırlar. parçaları bütün oluşturur ve yok eder.
- \* "is part of" (whole/part)
- \* Her parça en çok 1 bütün içerisinde bulunur. Bir büttünde çok sayıda parça yer alabilir.



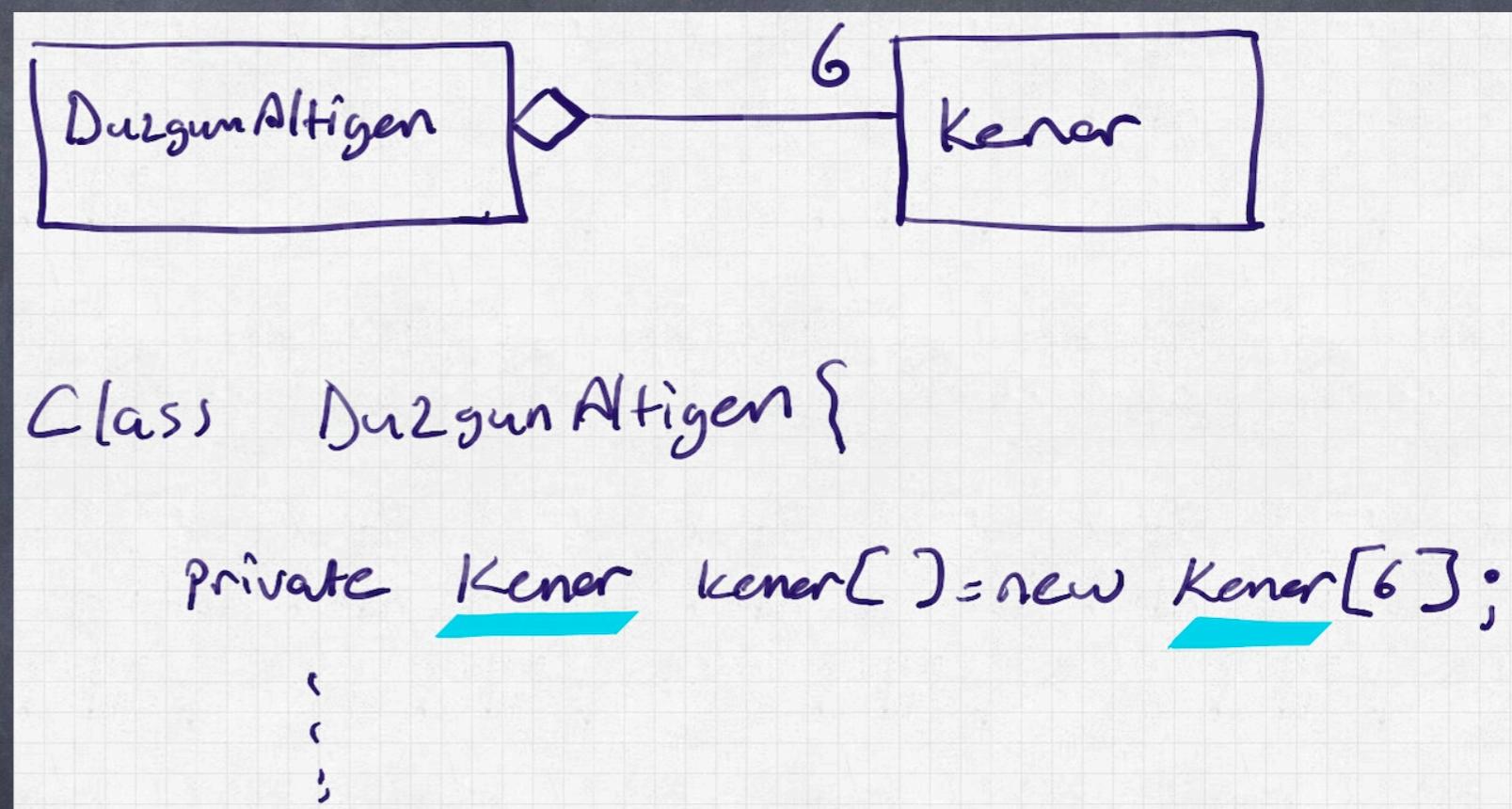
```
public class ATMMakinesi
{
 private Ekran ekran;
 private TusTakimi tusTakimi;
 private ParaBolmesi paraBolmesi;
 private KartGirisBolmesi kartBolmesi;

 public ATM() {
 ekran=new Ekran();
 tusTakimi=new TusTakimi();
 paraBolmesi=new ParaBolmesi();
 kartBolmesi=new KartGirisBolmesi();
 }
}
```

# Sınıf Diyagramı - Elementler arası bağlantılar

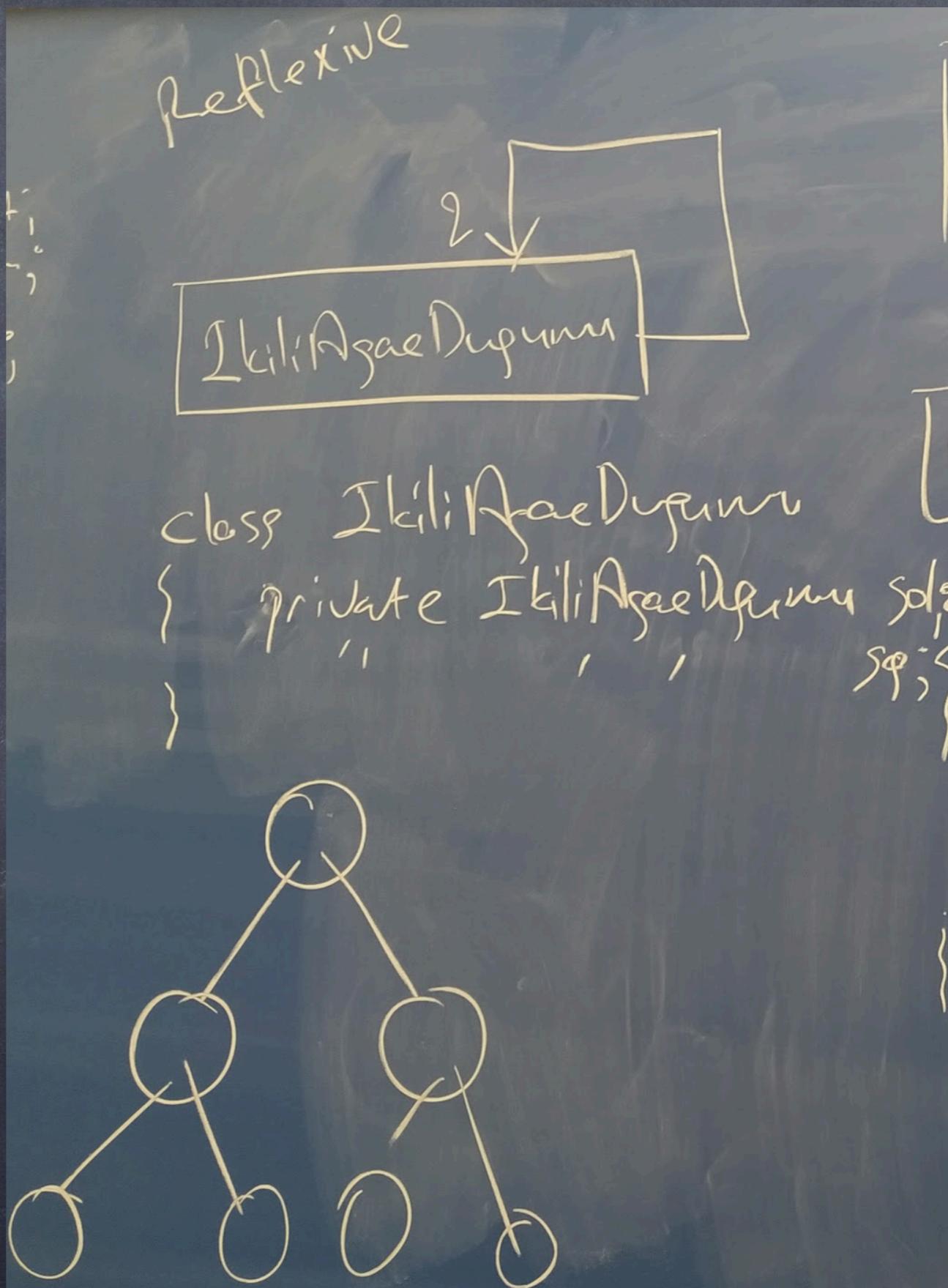
## "Aggregation" Bağıntısı

- \* Bütün parça bağıntısı (whole/part)
- \* Bütün, işini yapabilmek için parçaya ihtiyaç duyar.
- \* Parça nesneler bütün içerisinde yer alırlar. parçalar başka bütün içerisinde de kullanılabilir. Bütün yok edildiğinde parçanın yok edilmesi gerekmeyebilir (başka bütün içerisinde kullanılıyor olabilir)
- \* "consists of"



# Sınıf Diyagramı - Elementler arası bağlantılar

\* "Reflexive" Bağlantısı



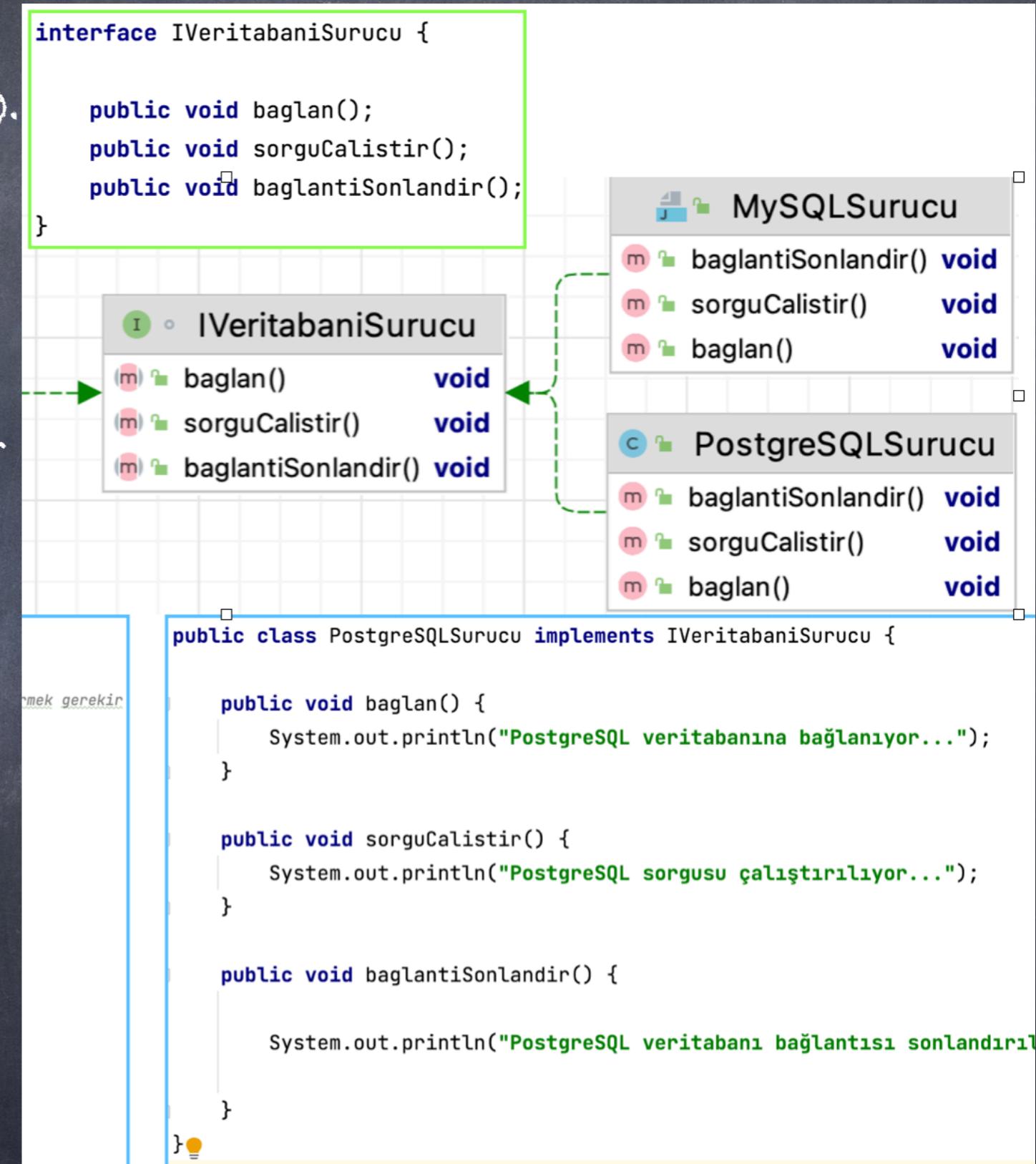
# Sınıf Diyagramı - Elementler arası bağlantılar

## "Interface"

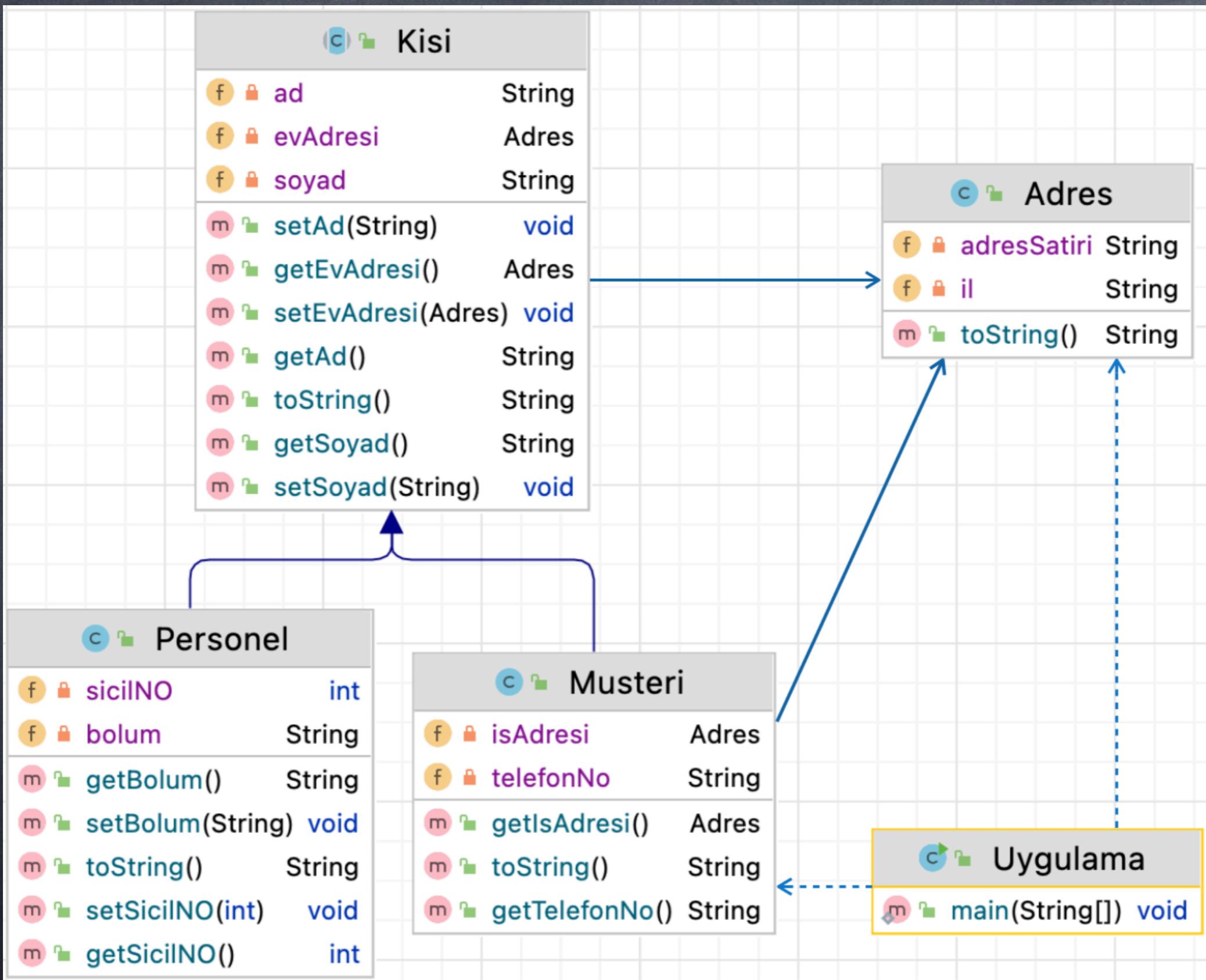
- \* Bir sınıfın yapmasını istediğimizi belirtmek için kullanırız ("is capable of"). Ne yapması gereği belirtilir, nasıl ile ilgilenilmez.

Arayüzler sayesinde:

- \* gerçeklemeler(sınıflar) henüz ortada yok iken istemci kod içerisinde kullanılabilir
- \* gereğekleme ve istemci modüller aynı anda farklı kodlayıcılar tarafından oluşturulabilir (takım çalışması)
- \* yeni özellik eklendiğinde istemci kod değiştirilmez (sözleşme yapılır)
- \* istemci kod, gereğekleme kısmındaki değişikliklerden etkilenmez (bağımlılığın zayıflatılması)
- \* aynı istemci kod farklı modüller tarafından defalarca kullanılabilir (kod tekrar kullanımı-code reuse)

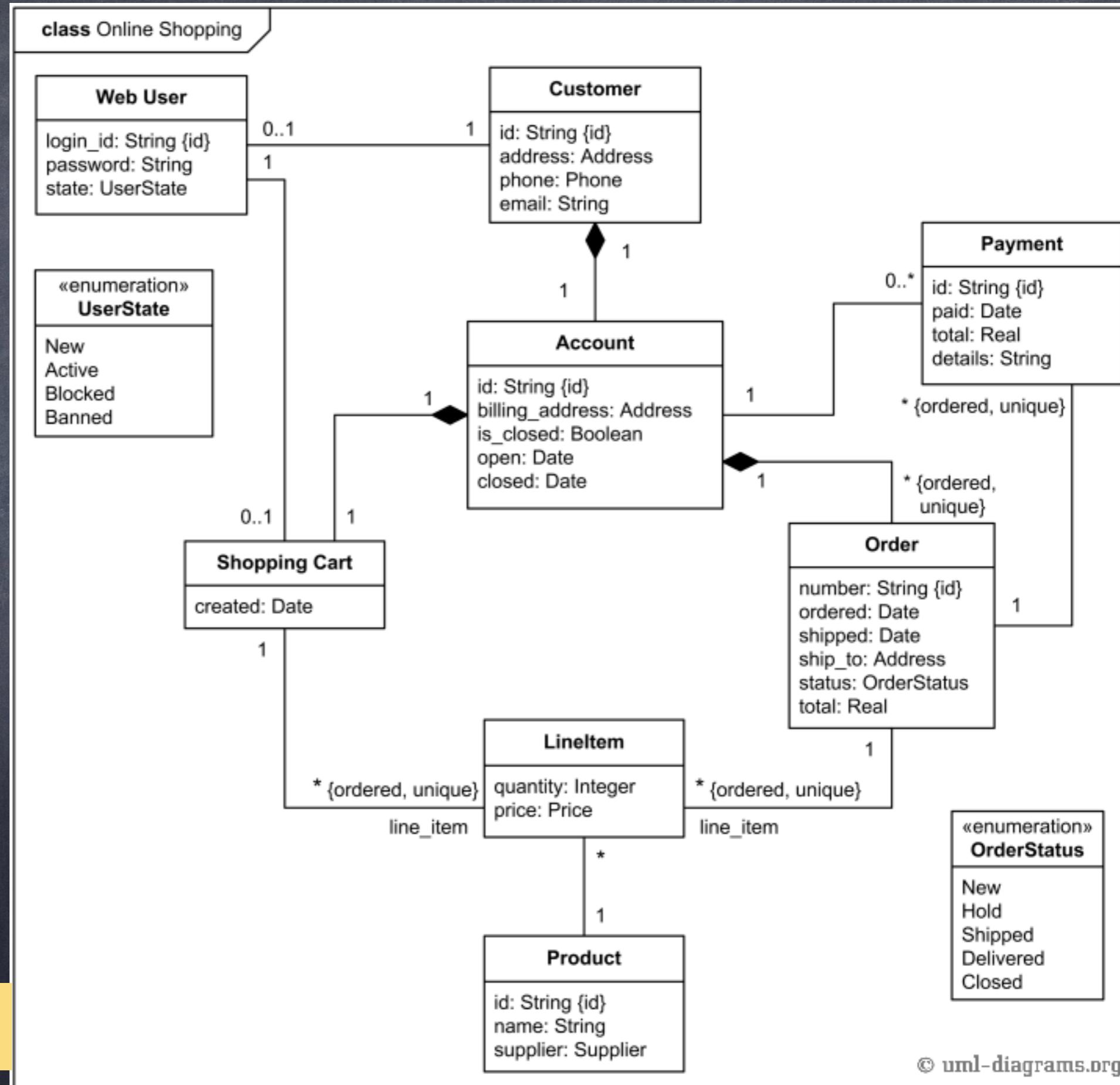


# Örnek Uygulama: Elementler arası Bağıntılar



# Örnek: Çevrimiçi Alış Veriş Sistemi Sınıf Semasi

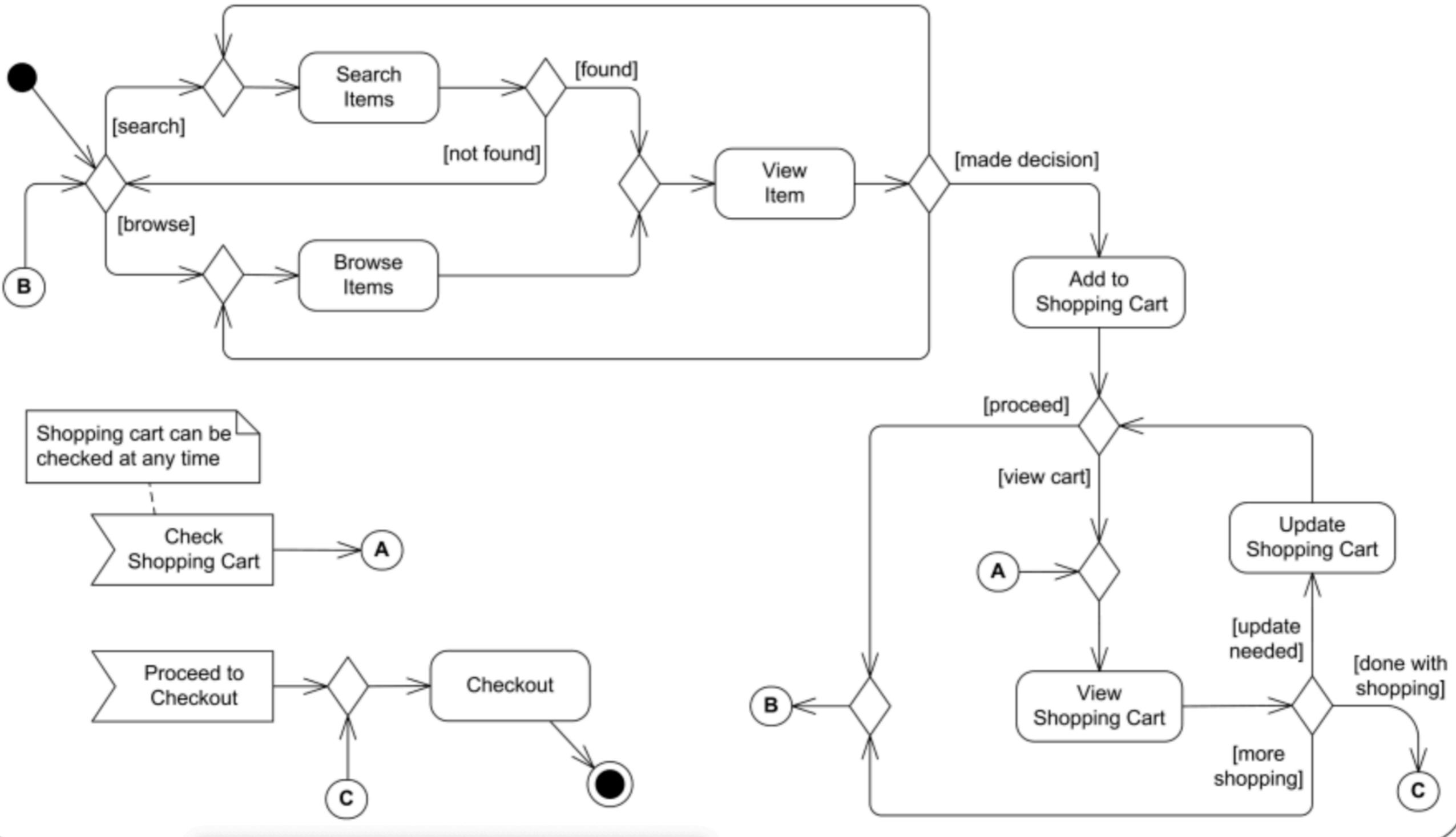
- \* Her müşterinin eşsiz bir "id" si ve sadece bir hesabı vardır.
- \* Hesaba ait bir alış veriş kartı (sepeti) (internet üzerinde alışveriş için) ve siparişler (klasik alışveriş, telefon, yüz yüze vb.) (o ya da çok) vardır.
- \* Sipariş ve alış veriş kartı alışveriş kalemlerine sahiptir.
- \* Alış veriş kalemleri mutlaka bir ürünle ilgilidir. Ürünler çok sayıda alış veriş kalemiyle ilişkili olabilir.



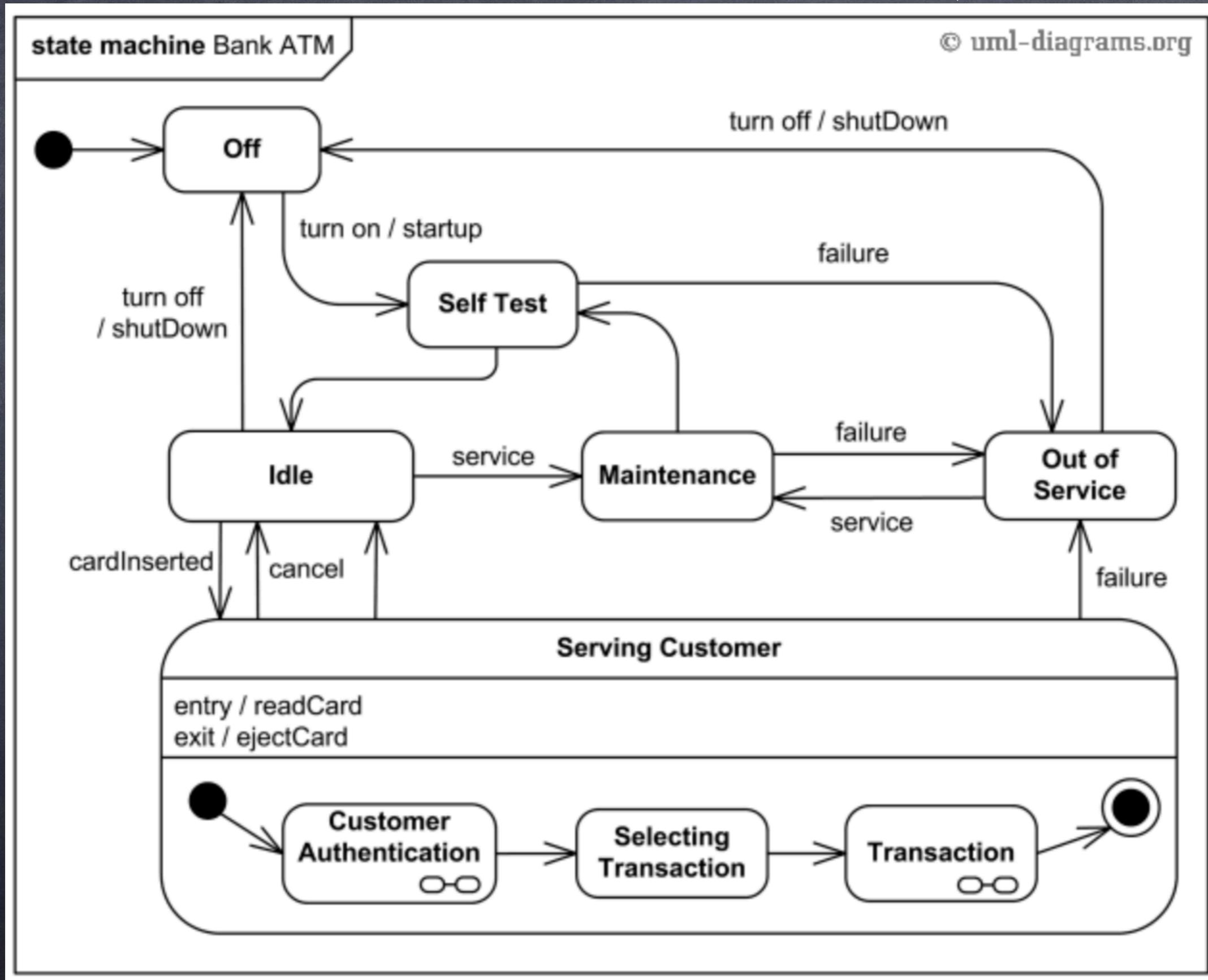
# Örnek: Gevrimiçi Alış Veriş Sistemi Etkinlik Şeması

## Online Shopping

© uml-diagrams.org



# Örnek: ATM Durum Makinası Diyagramı



# Örnek: E-Ticaret Uygulaması

Belirtilen gereksinimlere (iş kurallarına) göre nesneleri belirleyerek sınıf şemasını oluşturunuz.

## Senaryo

Elektronik ticaret yazılımının ihtiyacı olan verilerin yönetimi için bir veritabanı geliştirilmesi isteniyor. İş kuralları verilen bu veritabanının VB diyagramını ve ilişkisel şemasını oluşturunuz.

## İş Kuralları

- Bu veritabanında her müşteriye bir numara verilerek müşterinin TC Kimlik numarası, adı, soyadı, yaşadığı ili saklanması düşünülüyor.
- İllerin plaka numarası ve adı saklanır.
- Her siparişe bakan bir satış temsilcisi mevcuttur. Satış temsilcilerinin TCKimlikNo, ad ve soyad bilgileri mevcuttur.
- Ürünlerin (kişisel bilgisayar, telefon vb.) kodu, adı, fiyatı ve stok miktarlarının saklanması gerekmektedir.
- Ürünlerin kategorileri (bilgisayar, ev elektroniği, kozmetik vb.) mevcuttur.
- Müşterilerin ürün siparişleri saklanarak her bir siparişe bir fatura kesilmesi sağlanmalıdır.
- Her siparişin eşsiz bir sipariş numarası ve sipariş tarihi mevcuttur.
- Sipariş edilen bir ürünün sipariş adedi ve birim fiyatı (kişiye özel indirim v.s. nedeniyle ürün tablosundaki fiyat farklılığı gösterebilir) da kaydedilmelidir.
- Faturaların fatura numarası, tarih ve fatura adresi bilgileri saklanmalıdır. (Toplam fiyat hesaplanabilir ya da saklanabilir).
- Siparişler bir kargo firması tarafından iletilir. Kargo firmasının kodu, adı, adresi bilgileri yer alır. Her kargo firmasında siparişlerden sorumlu bir yetkili yer alır.
- Bir ürünün yalnızca bir kategorisi mevcuttur. Bir kategori çok sayıda ürünün kategorisi olabilir.
- Bir siparişte en az bir ürün bulunur. Ancak çok sayıda ürün de bulunabilir. Bir ürün çok sayıda siparişte yer alabilir.
- Bir müşteri çok sayıda sipariş verebilir. Bir sipariş yalnızca bir müşteri tarafından verilebilir.
- Bir siparişin yalnızca bir faturası olabilir. Bir fatura yalnızca bir siparişin faturası olabilir.
- Bir sipariş ile yalnızca bir satış temsilcisi ilgilendir. Bir satış temsilcisi çok sayıda sipariş ile ilgilenebilir.
- Bir müşteri yalnızca bir ilde yaşayabilir. Bir ilde çok sayıda müşteri yaşayabilir.
- Bir sipariş yalnızca bir kargo firması tarafından iletilir. Bir kargo firması çok sayıda sipariş iletebilir.