

# BİL 201

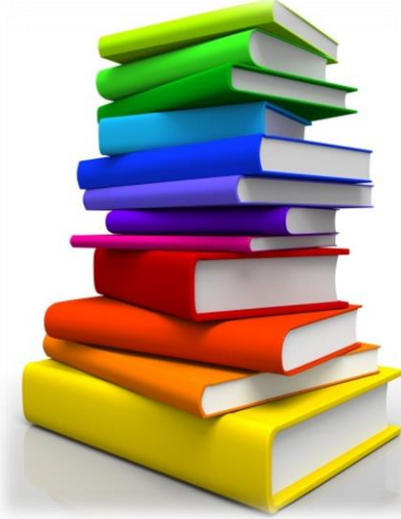
# NESNE YÖNELİMLİ PROGRAMLAMA (NYP)

**DERS #3**

**Öğretim Üyesi: Doç. Dr. Deniz Kılınç**

# BÖLÜM 3 – Nesne Yönelim Programlamaya Giriş

- Bu bölümde aşağıdaki konular anlatılacaktır
  - Nesneye Yönelik Programlamaya Giriş
  - Soyutlama, Saklama
  - Nesneye Yönelik Programlamanın Farklılıkları
  - Nesne Kavramı
  - Sınıf Kavramı
  - Sınıf Oluşturulması
  - Örnek Sınıflar



# Nesne Yönelimli Programlamaya Giriş

- Nesne yönelimli programlama (OOP), (**O**bject **O**riented **P**rogramming) nesneyi merkezine alan bir bilgisayar **programlama yaklaşımıdır**.
- Nesne yönelimli programlama terminolojisindeki “**object/nesne**” ve “**oriented/yönelimli/yönelik**” kavramları ilk olarak 1966 veya 1967 yılında NYP’nin babası olarak bilinen **Alan Kay** tarafından kullanılmıştır.
- NYP paradigmasının bazı özelliklerini destekleyen ilk programlama dili olan **Simula’dan** sonra, Alan Kay, Dan Ingalls, Adele Goldberg ve arkadaşları tarafından NYP paradigmasını tam destekleyen programlama dili **Smalltalk’ın** ilk versiyonu 1969-1972 yılları arasında **Xerox PARC’ta** geliştirilmiştir.

simula



# Nesne Yönelimli Programlamaya Giriş

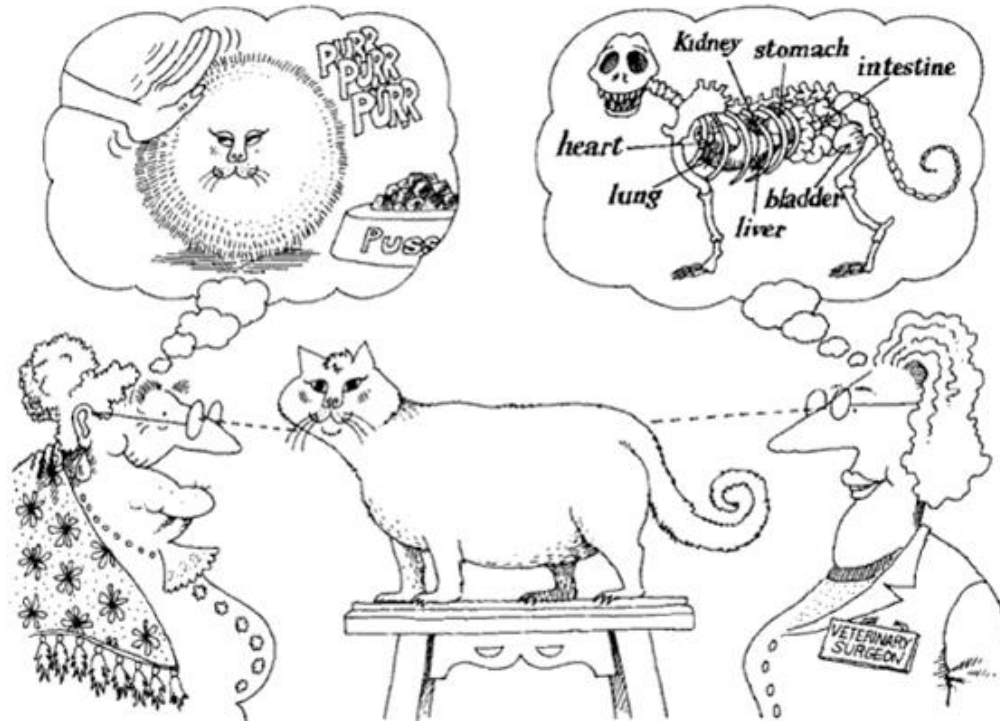
- 90'lı yıllarda programlama dillerine olan destek artınca *nesneye yönelik programlama* **etkin** ve **yaygın** olarak kullanılmaya başlandı.
- Nesneye yönelik programlamanın temel kavramları;
  - Büyük yazılımlar geliştirmeyi kolaylaştıran **Soyutlama (Abstraction)**,
  - Yazılımları değiştirmeyi ve korumayı kolaylaştıran **Saklama/Sarmalama (Encapsulation)**,
  - Yazılımları genişletilebilir kılan **Sınıf Hiyerarşisi** ve **Kalıtım (Inheritance)**,
  - Kalıtım hiyerarşisi içerisinde farklı işlemler yapan (fakat) aynı isimdeki özellik veya metotların kullanımına **Çok Biçimlilik (Polymorphism)**

# Nesne Yönelimli Programlamaya Giriş



# Soyutlama

Önemli özelliklere **odaklanabilmek** için ayrıntıları  
göz ardı etme sürecidir.



# Soyutlama (devam...)

- Soyutlama temel olarak **veri** ve **kontrol** soyutlaması olarak yapılır.
  1. Bir veri tipinin nasıl yapılandırıldığının ayrıntılarını göz ardı etmemize izin veren soyutlama tarzına “**veri soyutlaması**” denir.
  2. **Kontrol soyutlaması** ise yapısal programlama ile gelen altprogram, fonksiyon gibi kavramlar üzerinde yapılan soyutlamadır.

**Örnek:** Bir **Kisi** nesnesinde, kişinin yaşını tutan bir **tamsayı değişkeni** olan **yas** değişkenini ele alalım.

- Bu **yas** değişkenine, programın çalıştırma anında, "-10" değerinin atanmasını kimse engelleyemez. İşte burada nesne yönelimli programlamanın getirdiği görünürlük ve özellik tanımlama gibi yetenekler kullanarak "yas" değerine gerçek bir değer girilmesi garanti edilebilir. Buna **veri soyutlaması** adı verilir.

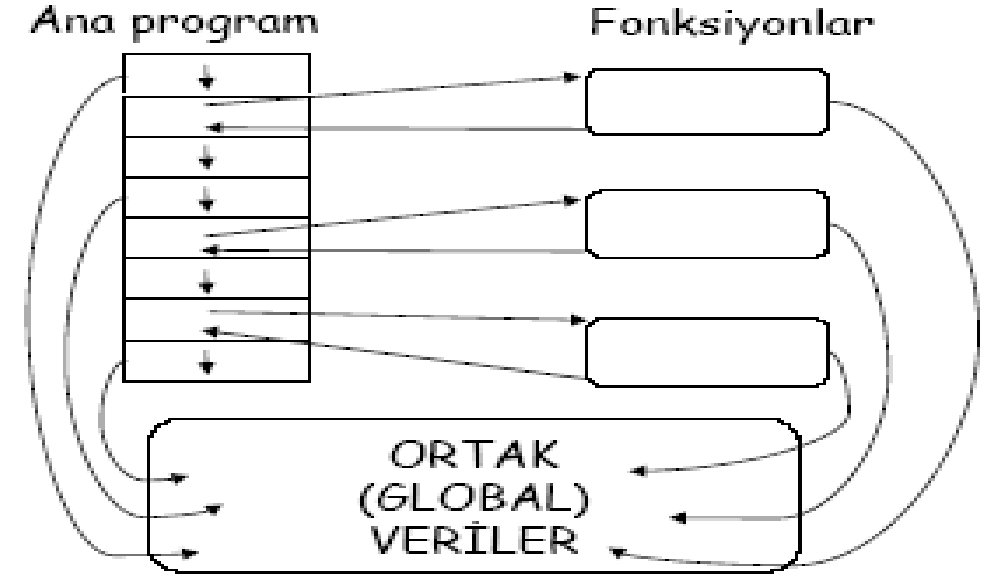
# Saklama/Sarmalama (Encapsulation)

- Gerçek hayatta **information hiding** yöntemiyle sıkça karşılaşırız.
- Örneğin arabanın deposuna ne kadar yakıt kaldığını öğrenmek için depoya baktığımızda göremeyiz.
- Bunun yerine gerekli bilgiyi bize aracın **gösterge panelindeki yakıt göstergesi** vermektedir.
- Saklama işlemi Sınıflar kullanılarak gerçekleştirilir.



# Yapısal Programlama vs. Nesne Yönelimli Programlama

- Yapısal Programlamada sadece bir soruna odaklı farklı fonksiyonlar yazılır ve **sadece o sorun** için fonksiyonlar bulunur.
- Dolayısıyla yapısal programlama yaklaşımıyla oluşturulmuş bir program binlerce ayrı isimde tanımlanmış değişken ve yüzlerce farklı fonksiyona sahiptir.
- En ufak işlem için bile **ayrı fonksiyon oluşturulması gerekir** ve bu programın **karmaşıklığını** arttırır.



**Gerçek dünyadaki sistemler sadece fonksiyonlardan oluşmaz.**  
Sistemin gerçeğe yakın bir modelini bilgisayarda oluşturmak zordur.

# Yapısal Programlama vs. Nesne Yönelimli Programlama

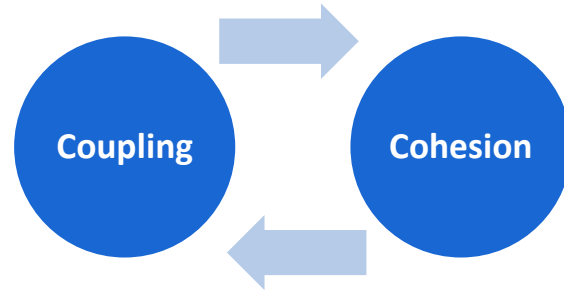
- Nesne yönelimli programlama yaklaşımı **doğaldır**.
- Nesnelerden oluşan bir dünya düşünmek **basittir**.
- Nesne Yönelimli Programlama yaklaşımı yapısal programlama yaklaşımının **daha genişletilmiş ve gerçeğe yakın versiyonudur**.
  - Değişkenler ve metotlar burada da **kullanılır** fakat burada **nesne** üzerine **odaklanılır**.
- Yapısal programlamadaki **büyük ve karmaşık** sistemleri tasarlamak yerine, *birbiriyle iletişim ve etkileşim halinde nesnelerin* olduğu bir dünya kolayca tasarlanabilir.
  - Bu nesnelerin çalışması için ayrıca bir mantık oluşturmaya gerek kalmaz.
- Nesne yönelimli programlama yaklaşımında, projenin erken safhalarında projeyi tamamlamak için **gerekli** olan **tüm nesnelerin** tanımlanması, nesneleri tanımlayan **sınıfların yaratılması** yatar. Böylece yaratılan **her nesne** kendi verisini kendinde barındırır ve **diğer nesnelerin kendisinden istediği görevleri yerine getirir**.

# Yapısal Programlama vs. Nesne Yönelimli Programlama

**Dikkat:** Nesne Yönelimli Programlama yanlış bir tasarımla **KARMAŞIKLIĞI** arttırabilir.

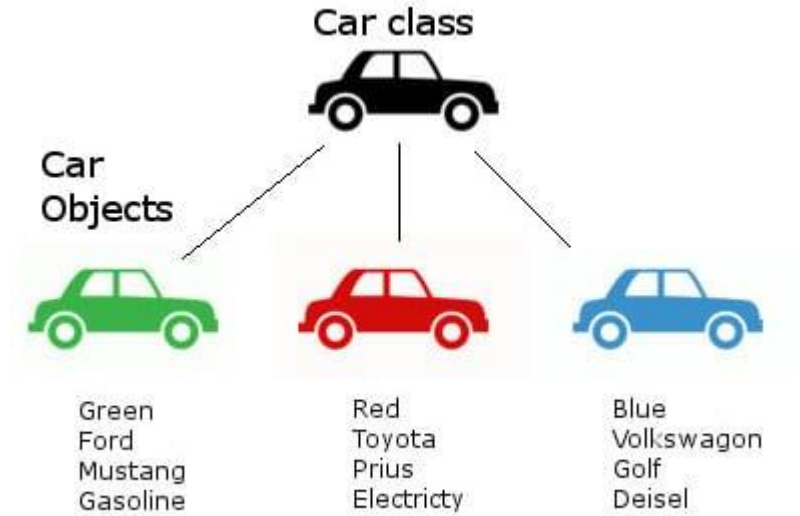
## Reçete:

1. Her zaman **Temiz Kodlama** (Clean Code) tarafında kalmalıyız.
2. **Tasarım Prensiplerinden** (Design Principles) ve **Tasarım Kalıplarından** (Design Patterns) uzaklaşmamalıyız.



# Nesne Kavramı ilk Tanım

- Nesne, **niteliklere/özelliklere** ve **davranışlara** sahip **somut** bir **varlıktır**.
- Bir nesnenin **niteliği/özelligi** onun sahip olduğu **özelliklerdir**.
  - Nesnenin bir **niteliğinin değeri** o nesnenin durumunu oluşturur.
- Bir nesnenin **davranışları** onun yapabildiği / onunla yapılabilen işlemlerdir /operasyonlardır.



# Örnek 1: Sipariş verme yazılımı

## Yapısal Programlama Çözümü

- Müşteri ve sipariş bilgileri ayrı değişkenlerde tutulurlar. Diziler kullanılır.
- Bu değişkenler üzerinden işlemler yapılır (sipariş ekle, tutar hesaplama, ödeme gerçekleştir ... ).

## Nesne Yönelimli Programlama Çözümü

- Programın içereceği nesnelerden birisi “**Siparis**” nesnesi olur.

# Örnek 1: Sipariş verme yazılımı (devam..)

Siparis nesnesi şunları içerir:

## **Nitelik/özellik olarak;**

- Sipariş Tarihi
- Müşteri
- Durumu

## **Davranış olarak;**

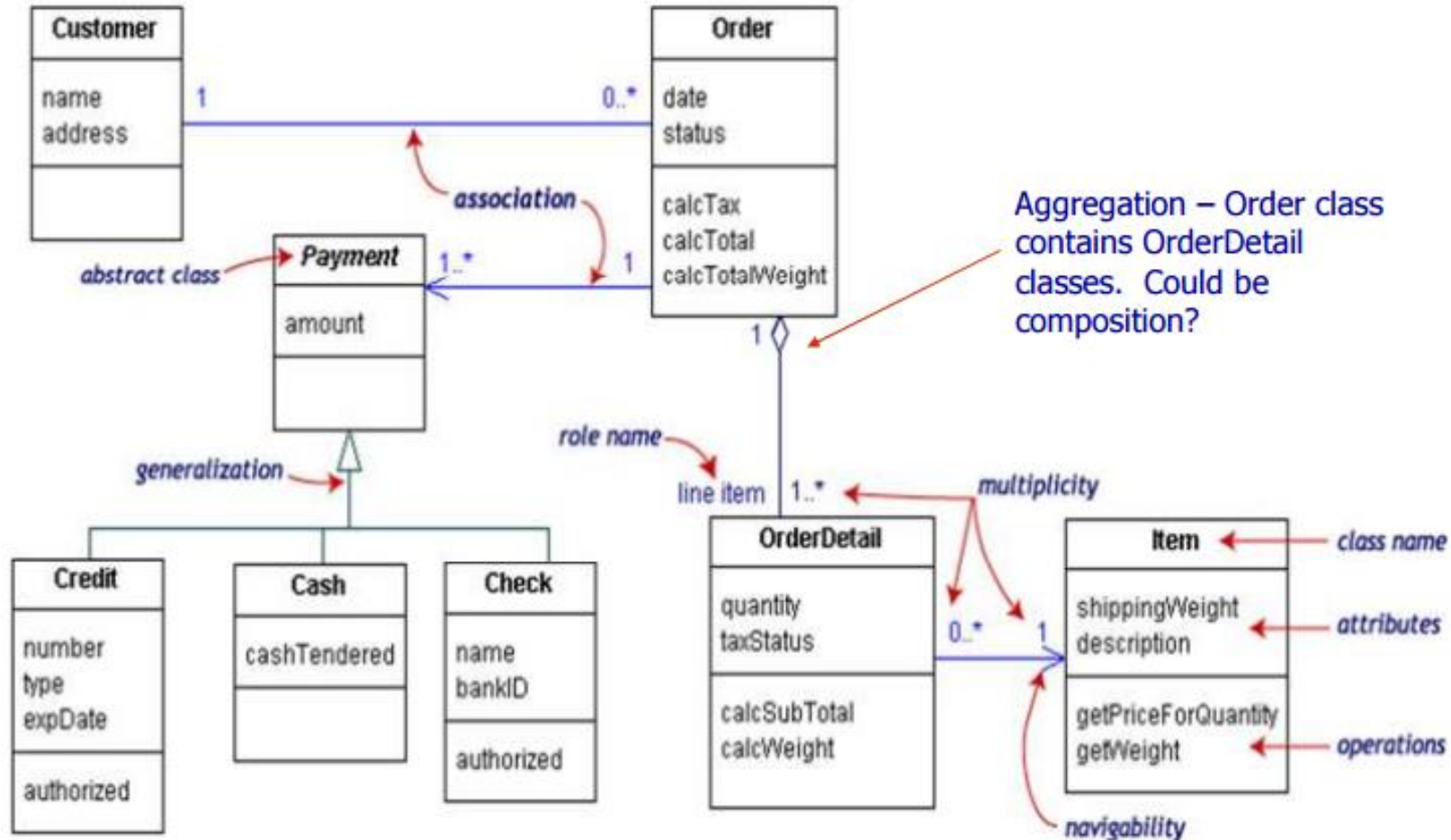
- Oluştur
- Öde
- Kalem (Ürün) Ekle / Çıkart
- Sevk Et

# Örnek 1: Sipariş verme yazılımı (devam..)

- Sipariş verme yazılımının tasarımı için gereken tüm nesneler **düşünülmeli** ve nesnelerin **nitelikleri / özellikleri** ve davranışları tanımlanmalıdır.
- Başlıca nesneleri şunlar olmalıdır :
  - **Müşteri** Nesnesi,
  - **Ödeme** Nesnesi,
  - **Ürün** Nesnesi,
  - **Sipariş Detayı** Nesnesi

*Önemli: Aslında bu nesnelerin hemen hepsinin karşılığı olan bir veritabanı nesnesi (tablo) olmalı değil mi?*

# Örnek 1: Sipariş verme yazılımı (devam..)





# Örnek 1: Sipariş verme yazılımı (devam..)

- Her iki programlama yaklaşımıyla (yapısal ve nesne yönelimli) da **doğru bir yazılım** üretilebilir?
- Aynı zamanda iki programlama yaklaşımıyla da **yeniden kullanılabilir** (kısmen) modüller üretilebilir.
- İki programlama yaklaşımındaki önemli farklılıklar:
  - Yazılımcının projenin erken safhalarındaki **planlama ve düşünme farklılığı**
  - Üretilen programların **karmaşıklıkları**
  - Yeni ihtiyaçların ve **düzenlemelerin sisteme adapte edilmesindeki kolaylık**

# Örnek 2: Trafik Simülasyon

**Amaç:** Bir şehrin trafik karışıklığını gidermek için gerçek zamanlı trafiği simüle eden bir simülasyonun yapılması gerekmektedir.

Gerekli olan nesneler :

- Arac Nesnesi,
  - Yaya Nesnesi,
  - Trafik İşaretçileri Nesnesi
- Bu nesnelerin her biri kendi bilgilerini saklar ve kendi davranışlarına sahip olur.
  - Böylelikle nesnelerin yardımcılığıyla bir şehrin trafiği gerçek zamanlı olarak simüle edilebilir.

# Örnek 2: Trafik Simülasyon (devam...)

## Arac nesnesi

### Nitelikler

- Plaka Bilgisi
- Sürücü bilgisi
- Anlık Hızı Bilgisi

### Davranışlar

- Anlık Hız Değiştirme Metodu
- Trafik Kuralı Kontrol Metodu

## Yaya nesnesi

### Nitelikler

- Kimlik Numarası
- Anlık Hızı

### Davranışlar

- Anlık Hız Değiştirme Metodu
- Trafik Kuralı Kontrol Metodu

# Sınıf ve Nesne Kavramı

Dünyayı anlayabilme kabiliyetimizin çoğu

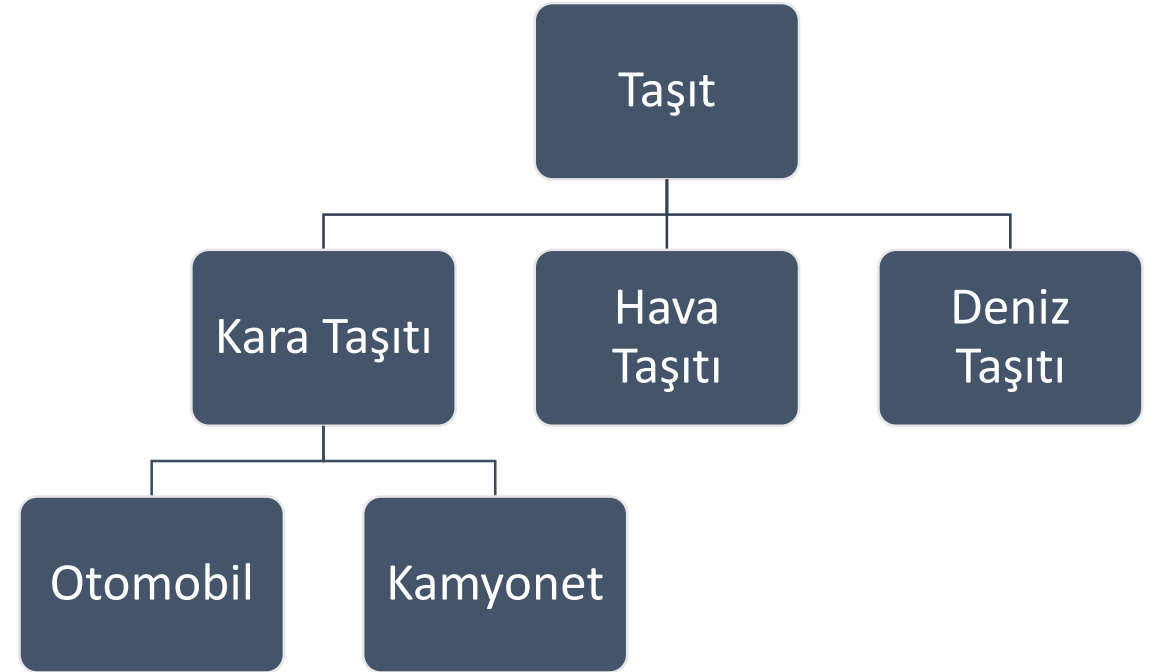
*nesneleri ve olayları*

sınıflar halinde kategorilendirebilmemizden gelmektedir.

- Örneğin; Çocukken “**hayvan**” kavramını, “hayvan” kelimesini öğrenmeden çok öncesinde biliyordunuz. Hayvanlarla ilk karşılaşmanız köpeğinizle veya sokak kedisi ile veya bir çiftlikteki keçi ile olmuş olabilir. **Konuşmayı öğrendikçe, tüm hayvanlar için aynı terimleri kullanıyorduk.**
- Tecrübelendikçe, **hayvanlar** arasındaki farkları anlayabildik.
- Mesela bir **köpek** ile **koyunun** arasındaki farkı, sonrasında ise yavruları ile yetişkinleri arasındaki fark gibi...

# Sınıf ve Nesne Kavramı (devam...)

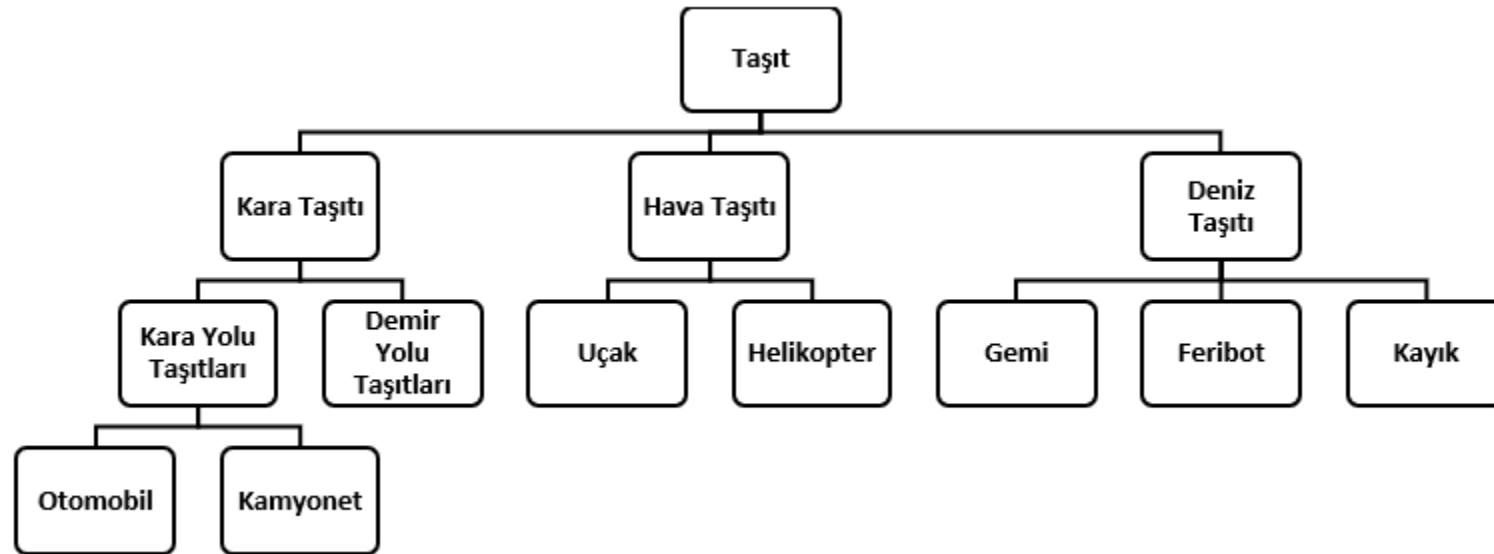
- Aynı şekilde **Taşıtlar** sınıfını anlamamız, **Kara taşıtları** ile **Deniz ve Hava taşıtları** arasındaki benzerlikleri görmemize yardımcı oldu.
- *Kara taşıtları sınıfını öğrendikten sonra **Otomobil** ile **Kamyonet** arasındaki benzerlikleri daha rahat kavrayabildik.*



*Yukarıdan aşağı incelediğimizde taşıtlar genel olarak kara, hava ve deniz olmak üzere 3 alt sınıfa ayrılırlar. Bir kara taşıtı ise genel bir taşıtın tüm özelliklerini barındırır.*

# Sınıf ve Nesne Kavramı (devam...)

- **Sınıf:** NYP yaklaşımında sınıflar varlıkların ortak özellik ve davranışlarını (operasyonları) tanımlamamızı ve onları *soyutlamamızı* sağlayan özel veri türleridir (*kullanıcı tanımlı veri türü – user defined data type*)



# Sınıf ve Nesne Kavramı (devam...)

- **Nesne** daha genel bir sınıfın **somutlaşmış** bir örneğidir.
- **Nesneler** gerçek dünyadaki sınıfların birer örneğidir (**instance**).
- Örneğin; Sizin arabanız daha genel bir sınıf olan **Araba** sınıfının **somutlaşmış** bir örneğidir. Bir nesne, bulunduğu sınıfın somutlaşmış bir örneğidir.
- **Canlı cisimleri** de “**nesne**” olarak düşünebilirsiniz:
  - Ev *bitkiniz*,
  - Balığınız,
  - Aile bireyleri birer nesnedir.

# Sınıf ve Nesne Kavramı (devam...)

**Örneğin:** Bir **araba** sınıfı ile ilgili hangi özellikleri ve davranışları biliyoruz?

**Düşünelim...**



# Araba Sınıfı Özellikler ve Metotlar

## • Özellikler

- Marka: string
- Model: int
- Renk: string
- YakitTuru: eYakitTur
- Fiyat: Decimal
- Motor: motor\_sinifi

## • Metotlar / Davranışlar

- SatinAl()
- Sat()
- Calistir(): boolean
- Sur()
- Durdur(): boolean



- Farklı bu kadar çok metodu tek sınıfta barındırmak mantıklı değil.
- Bir sınıfa fazla sorumluluk yüklemiş oluyoruz.
- Her sınıfın tek sorumluluğu olmalı.
- **Single Responsibility Principle**

# Araba Sınıfı UML Sınıf Diyagramı Gösterimi

Araba	Sınıf, Interface Adı
Marka: string Model: short Renk: string YakitTuru: eYakitTur Fiyat: Decimal Motor: motor_sinifi	Üyeler ve Özellikler
SatinAl() Sat() Calistir(): boolean Sur() Durdur(): boolean	Metotlar / Davranışlar

# Sınıf ve Nesne Kavramı (devam...)

- Nesnelerin üyeleri, sınıf içerisinde kullanılan diğer değişkenler ile karışmaması için **field (üye değişkenler)** olarak adlandırılır.
- Nesnelerin **özellikleri (property)** arka planda üye değişkenleri ile ilişkilendirilirler.
- Nesneler üzerinden üye değişkenlerine erişilemez (**private**) ancak özelliklere erişilebilir (**public**).
- Genellikle; **üyeler** küçük harfle, **özellikler** BÜYÜK harfle başlar.
- Nesnenin örnek değişkenlerinin içerikleri, o nesnenin **durumunu** belirler.

# Sınıf ve Nesne Kavramı (devam...)

Araba sınıfına ait 2 nesnenin **durumu (state)**

Araba1
Marka: Renault Megane Model: 2009 Renk: Beyaz YakitTuru: eDizel Fiyat: 28.500 TL Motor: motor_nesnesi1
SatinAl() Sat() Calistir() Sur() Durdur()

Araba2
Marka: Citroen C5 Model: 2012 Renk: Füme YakitTuru: eBenzin Fiyat: 41.500 TL Motor: motor_nesnesi2
SatinAl() Sat() Calistir() Sur() Durdur()

# Sınıf Oluşturmak

- Sınıfları tanımlarken öncelikle **bir sınıf başlığı/tanımı** yaratılmalıdır. Sınıf başlığı üç parçadan oluşur:

1. İsteğe bağlı bir **erişim belirleyici (access modifier)**,
2. **class** anahtar kelimesi,
3. Sınıfın adı

`[erişim belirleyici] class [sınıfAdı]`

Örneğin;     **public class** Araba

- ❗ Sınıf adları nesnelerin tipini tanımladığı için genellikle tek bir addan oluşur.

# Sınıf Oluşturmak (devam...)

- **Araba** sınıfını tanımlarken kullanılan **public** anahtar sözcüğü **sınıf erişim belirleyicisidir**.
- Oluşturulan sınıfın amacına uygun olan **erişim belirleyicileri** kullanılır.

Sınıf Erişim Belirleyicisi	Açıklama
Public (UML → +)	Sınıfa erişim sınırsızdır
Protected (UML → #)	Sınıfa erişim bulunduğu sınıf ve <b><i>bu sınıftan türetilen sınıflar</i></b> ile sınırlıdır.
Internal – Varsayılan (UML → ~)	Sınıfa erişim ait olduğu assembly/exe ile sınırlıdır.
Private (UML → -)	Sınıfa erişim ait olduğu sınıf ile sınırlıdır.

# Sınıf Oluşturmak (devam...)

- Sınıf başlıkları tanımlanırken kıvrıcık parantezler ( “{}”) içerisinde sınıf gövdeleri tanımlanmalıdır.

## Örneğin;

```
public class Araba
```

```
{
```

```
//üye değişkenleri, özellikleri ve metotları buraya yazılır
```

```
}
```

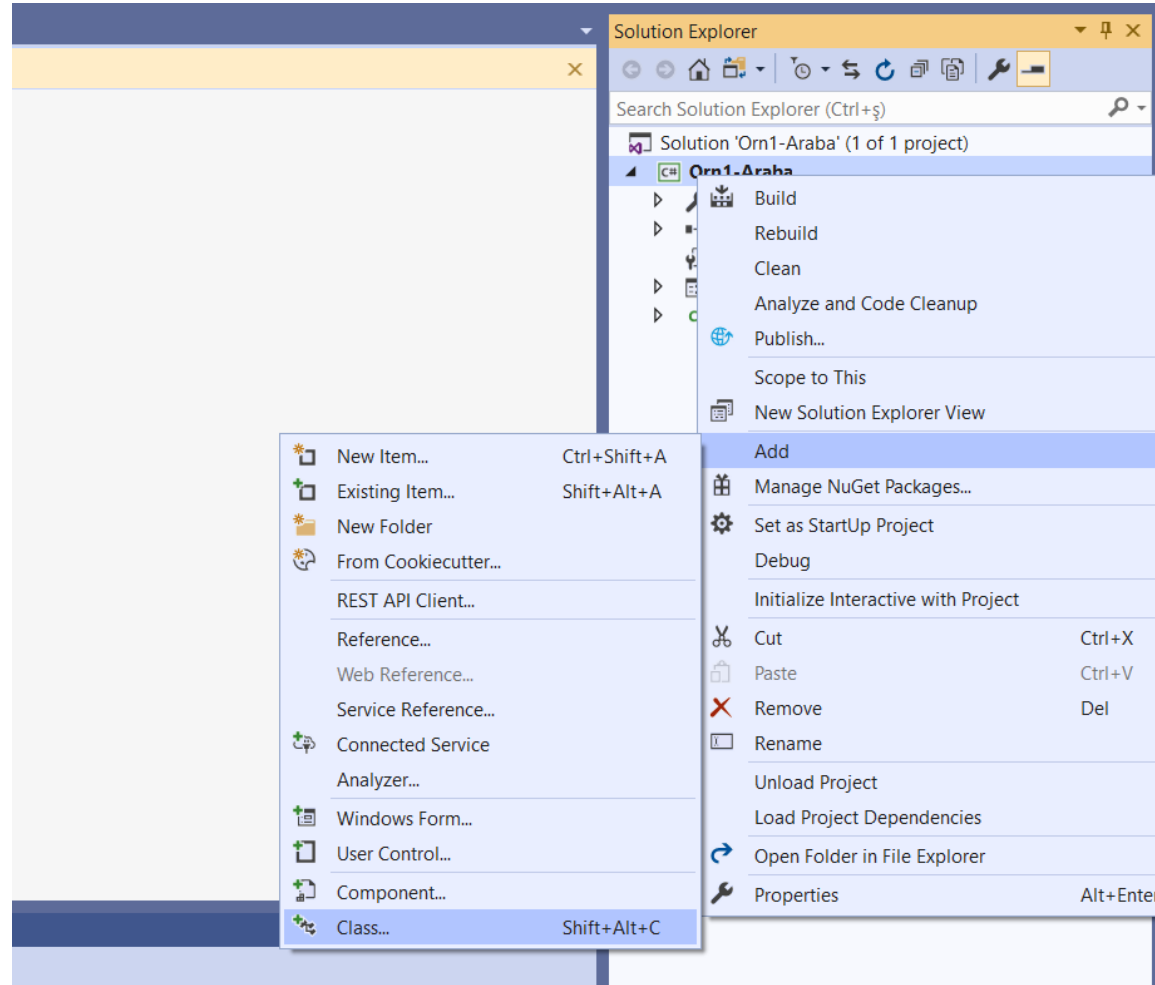
# Örnek 1: Araba Sınıfının Oluşturulması

- Araba sınıfını oluşturarak, sınıftan araba isimli bir nesne oluşturalım.
- Erişim belirleyicilere dikkat edelim.

Araba
+Marka: string +Model: short +Renk: string -fiyatKDVsiz: Decimal
+SatinAl() +Sat() #FiyatHesapla()



# Örnek 1: Araba Sınıfının Oluşturulması (devam...)



# Örnek 1: Araba Sınıfının Oluşturulması (devam...)

```
public class Araba
{
    //Üyeler
    private decimal fiyatKDVsiz;
    //Özellikler
    public string Marka { get; set; }
    public short Model { get; set; }
    public string Renk { get; set; }
    //Davranışlar
    public void SatinAl()
    { //Todo: Satın alma kodları
    }
    public void Sat()
    { //Todo: Satış kodları
    }
    protected void FiyatHesapla()
    { //Todo: Fiyat hesaplama kodları
    }
}
```

```
private void BtnTest_Click(object sender, EventArgs e)
{
    Araba araba = new Araba();
    araba.Marka = "Citroen C5";
    araba.Model = 2017;
    araba.Renk = "Kırmızı";
    araba.Sat();
    araba.SatinAl();
    MessageBox.Show("Araba markası: " +
        araba.Marka + "\n" +
        "Modeli: " + araba.Model +
        "\n" +
        "Rengi: " + araba.Renk);
}
```

**fiyatKDVsiz** üye değişkenine ve **FiyatHesapla** metoduna neden erişemedik?

# Örnek 2: DortgenPrizma Sınıfının Oluşturulması

DortgenPrizma
-en: float
-boy: float
-yukseklık: float
+HacimHesapla(float _en, float _boy, float _yukseklık): float

# Örnek 2: DortgenPrizma Sınıfının Oluşturulması (devam...)

```
public class DortgenPrizma
{
    private float en;
    private float boy;
    private float yukseklik;
    public float HacimHesapla(float pEn,
                              float pBoy,
                              float pYukseklik)
    {
        en = pEn;
        boy = pBoy;
        yukseklik = pYukseklik;
        return (en * boy * yukseklik);
    }
}
```

## Sınıfın Kullanımı

```
private void BtnTest_Click(object sender,
EventArgs e)
{
    DortgenPrizma dp = new DortgenPrizma();
    float hacim = dp.HacimHesapla(4, 5, 2);
    MessageBox.Show("Hesaplanan hacim: " +
                    hacim.ToString());
}
```

# Örnek 3: KimlikBilgileri Sınıfının Oluşturulması

KimlikBilgileri
+TCKimlikNo: long
+Ad: string
+Soyad: string
#DogumYeri: string
+DogumTarihi: DateTime
+KimlikBilgisiOlustur():string

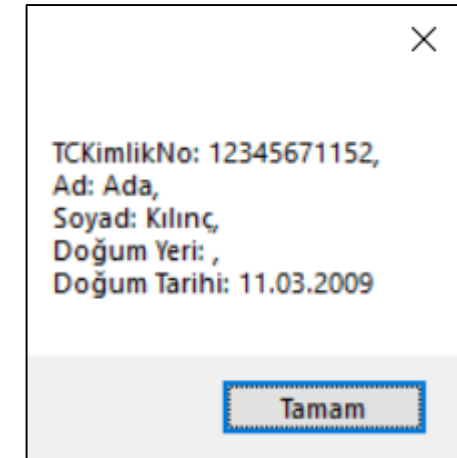
## Örnek 3: KimlikBilgileri Sınıfının Oluşturulması (devam...)

```
public class KimlikBilgileri
{
    public ulong TCKimlikNo;
    public string Ad;
    public string Soyad;
    protected string DogumYeri;
    public DateTime DogumTarihi;
    public string KimlikBilgisiOlustur()
    {
        return "TCKimlikNo: " + TCKimlikNo.ToString() + ", " +
            "\nAd: " + Ad + ", " +
            "\nSoyad: " + Soyad + ", " +
            "\nDoğum Yeri: " + DogumYeri + ", " +
            "\nDoğum Tarihi: " + DogumTarihi.ToShortDateString();
    }
}
```

# Örnek 3: KimlikBilgileri Sınıfının Oluşturulması (devam...)

## Sınıfın Kullanımı

```
private void BtnTest_Click(object sender, EventArgs e)
{
    KimlikBilgileri kb = new KimlikBilgileri();
    kb.Ad = "Ada";
    kb.Soyad = "Kılınç";
    kb.TCKimlikNo = 12345671152;
    kb.DogumTarihi = Convert.ToDateTime("11.03.2009");
    MessageBox.Show(kb.KimlikBilgisiOlustur());
}
```



Doğum yeri ?  
Sadece aile içi 😊

# Örnek 4: Öğrenci Sınıfının Oluşturulması

Oğrenci
+OgrNo: ulong
+ProgramTuru: EProgramTur (Lisans, YL, Doktora)
+Birim: string
+Bolum: string
+KimlikBilgisi: <b>KimlikBilgileri</b>
+OğrenciBilgisiOlustur():string

- Bir sınıf diğer sınıflardan yaratılmış bir **nesne üye barındırabilir**.
- Örneğin, ad, soyad, adres vb. gibi üye değişkenlerine sahip **KimlikBilgileri** sınıfından yaratılan bir nesne **Oğrenci** sınıfının üyesi olabilir.



# Örnek 4: Öğrenci Sınıfının Oluşturulması (devam...)

```
public class Öğrenci
{
    public ulong OgrNo;
    public EProgramTur ProgramTuru;
    public string Birim;
    public string Bolum;
    public KimlikBilgileri KimlikBilgisi;
    public string ÖğrenciBilgisiOlustur()
    {
        string bilgi;
        bilgi = "Öğrenci No: " + OgrNo +
            "\nProgram Türü: " + ProgramTuru +
            "\nBirim: " + Birim +
            "\nBölüm: " + Bolum +
            "\nKimlik No: " + KimlikBilgisi.TCKimlikNo +
            "\nAd: " + KimlikBilgisi.Ad +
            "\nSoyad: " + KimlikBilgisi.Soyad +
            "\nDoğum Yeri: " +
            "\nDoğum Tarihi: " +
            KimlikBilgisi.DogumTarihi.ToShortDateString();
        return bilgi;
    }
}
```

```
public enum EProgramTur
{
    Lisans,
    YuksekLisans,
    Doktora
}
```

## Örnek 4: Öğrenci Sınıfının Oluşturulması (devam...)

```
private void BtnOgrenciTest_Click(object sender, EventArgs e)
{
    Ogrenci ogrenci = new Ogrenci();
    ogrenci.OgrNo = 1892322;
    ogrenci.ProgramTuru = EProgramTur.Lisans;
    ogrenci.Birim = "Mühendislik ve Mimarlık Fakültesi";
    ogrenci.Bolum = "Bilgisayar Mühendisliği";
    /*
    ogrenci.KimlikBilgisi = new KimlikBilgileri();
    ogrenci.KimlikBilgisi.Ad = "Ada";
    ogrenci.KimlikBilgisi.Soyad = "Kılınç";
    ogrenci.KimlikBilgisi.TCKimlikNo = 12345671152;
    ogrenci.KimlikBilgisi.DogumTarihi = Convert.ToDateTime("11.03.2009");
    */
    KimlikBilgileri kb = new KimlikBilgileri();
    kb.Ad = "Ada";
    kb.Soyad = "Kılınç";
    kb.TCKimlikNo = 12345671152;
    kb.DogumTarihi = Convert.ToDateTime("11.03.2009");
    ogrenci.KimlikBilgisi = kb;
    MessageBox.Show(ogrenci.OgrenciBilgisiOlustur());
}
```

# UML Sınıflar Arası Birliktelik Türleri

## Association

- **İlişkinin en genel hali.**
- Nesneler tamamen bağımsız yaşamlara sahiptirler.
- Sahiplik yok (**ownership**).
- Ok ile gösterilir.

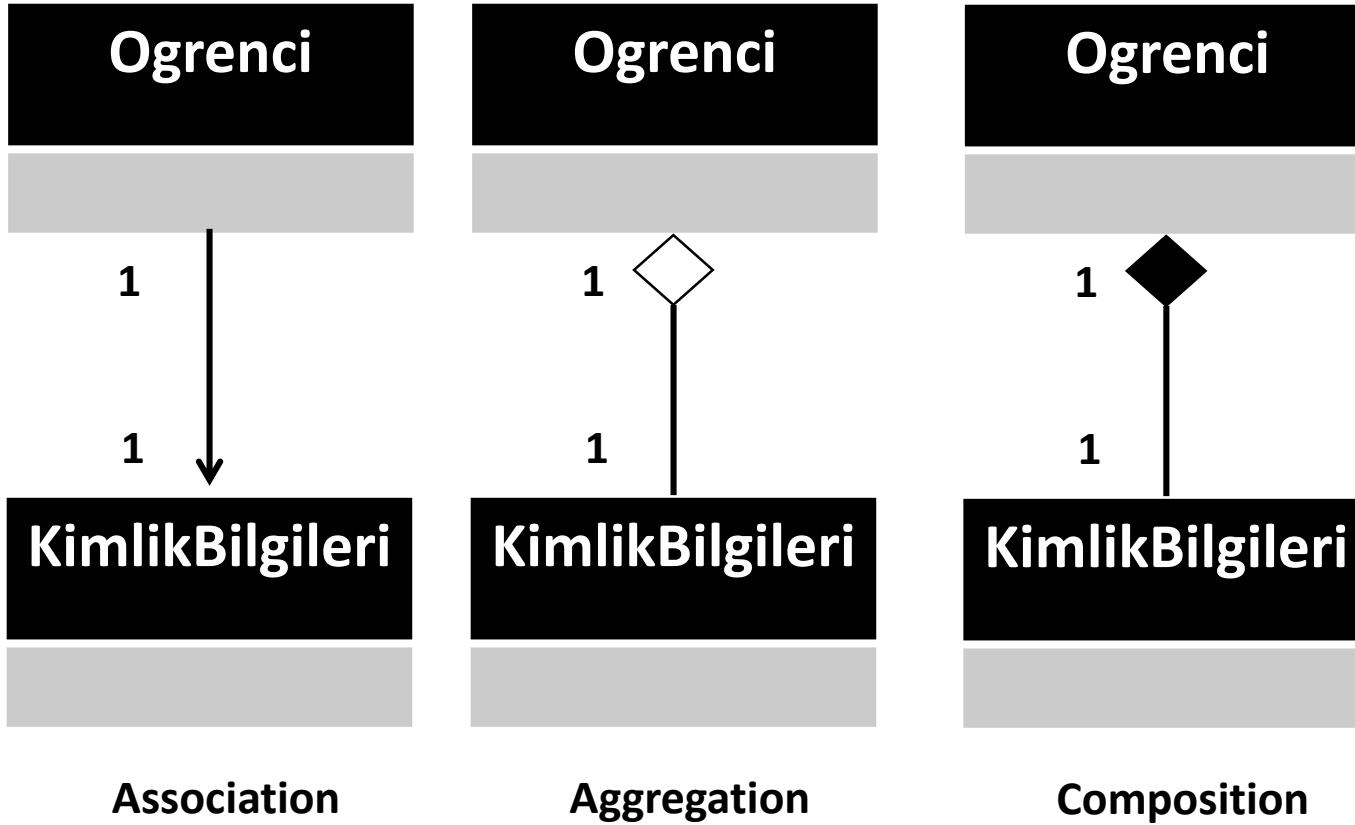
## Aggregation

- **Is-part-of:** Bir parçası olmak.
- Nesneler bağımsız yaşamlara sahiptirler.
- Yine de bir sahiplik söz konusudur (**ownership**).
- İçi boş elmas ile gösterilir.

## Composition

- **Is-made-of:** Daha güçlü bir parçası olmak.
- Nesneler birlikte oluşturulur ve birlikte yok edilirler.
- Sahiplik için temelidir.
- İçi dolu siyah elmas ile gösterilir.

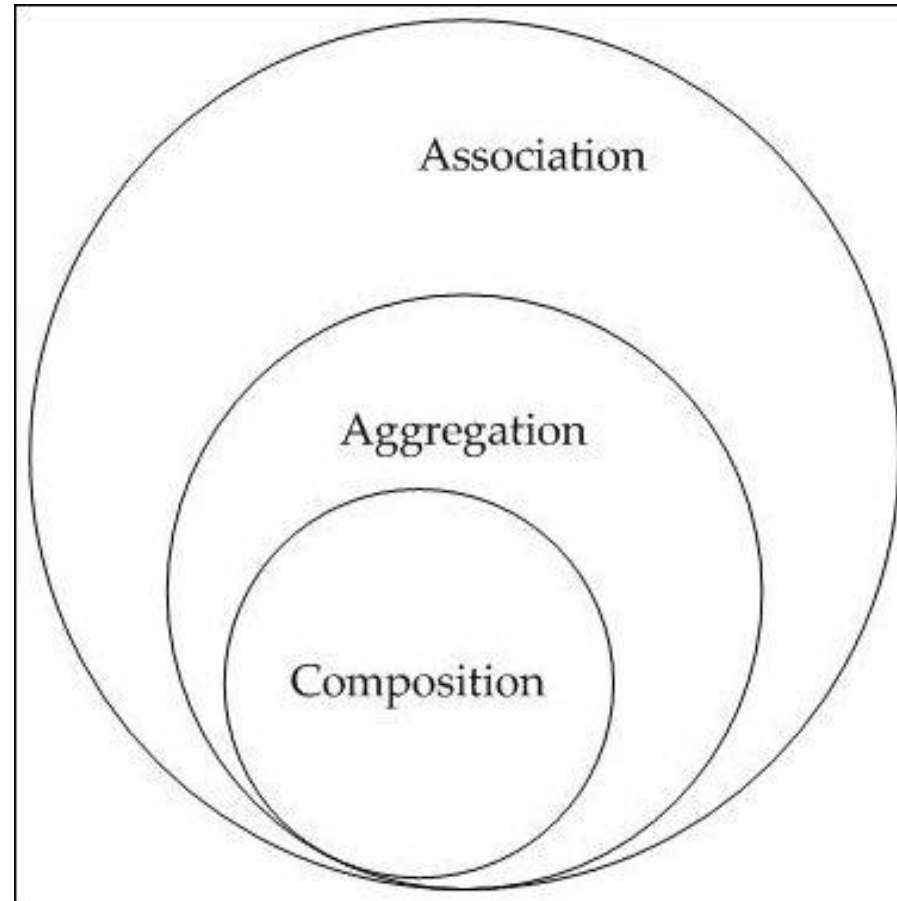
# UML Sınıflar Arası Birliktelik Türleri



## Dikkat

- **Coupling** ve **cohesion**'a dikkat ediyoruz.
- **Coupling:** Nesneler arası bağımlılık.
- **Cohesion:** Sınıfın üyeleri arasındaki benzerlik. Tek iş yapma.
- **Düşük coupling olmalı**
- **Yüksek cohesion olmalı.**
- Çok fazla composition ve aggregation coupling'i (genelde) düşürür.

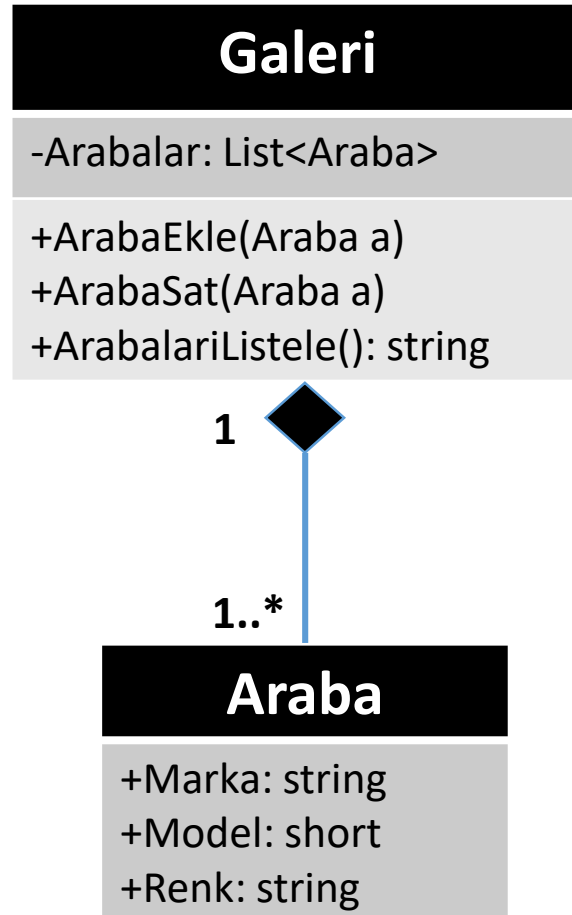
# UML Sınıfllar Arası Birliktelik Türleri



# Uygulama: Araba Galerisi

- **Gereksinim:** Bir araba galerisi uygulaması yapmak istiyoruz. Galeriye arabalar eklenip, çıkartılabilir (satılabilir), listeleme yapılabilir. Arabaların marka, model ve renk bilgisi olması yeterli. Galeriye ait farklı bilgiler (firma bilgileri vb.) isteğe bağlı eklenebilir.
- **Yaklaşım:** Nesne yönelimli mantıkla, soyut düşünelim.
- **Sorular soralım:**
  - Soru 1:** Gereksinimler tam mı? (**Requirements Gathering and Analysis**)
  - Soru 2:** Hangi sınıflara (varlıklara) ihtiyacımız var? (**Design-Dev**)
  - Soru 3:** Sınıfların birbirleri ile ilişkisi nasıl kurulmalı? (**Design-Dev**)
  - Soru 4:** Nasıl bir ekran ara yüzü tasarlayalım? (**Design-UI**)

# Uygulama: Araba Galerisi (devam...)

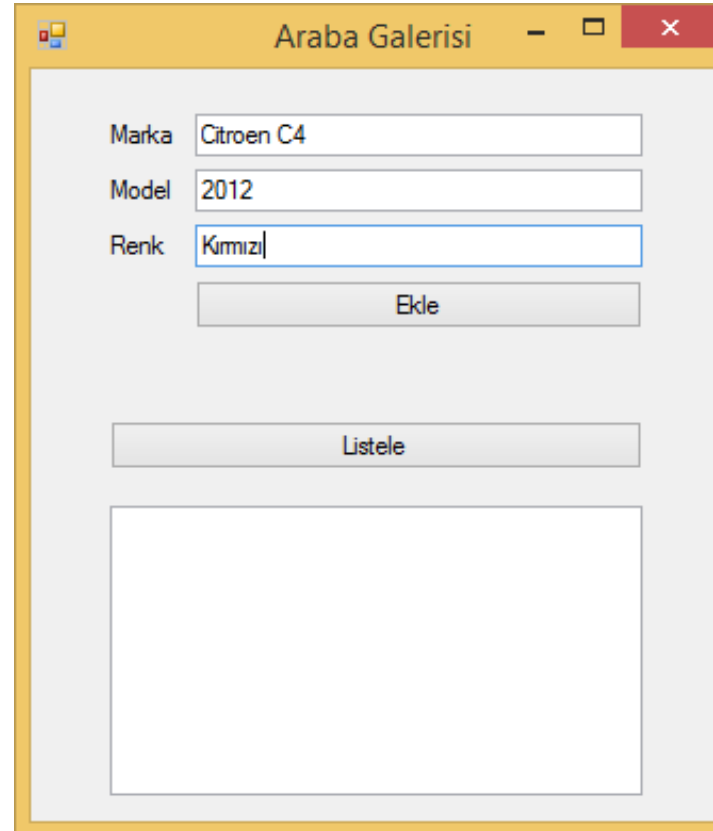


**Soru 2:** Hangi sınıflara ihtiyacımız var?

**Soru 3:** Sınıfların birbirleri ile ilişkisi nasıl kurulmalı?

# Uygulama: Araba Galerisi (devam...)

**Soru 4:** Nasıl bir ekran ara yüzü (UI) tasarlayalım?



## Notlar

- Listelenen kayıtları görmek için önce bir **textbox (Multiline)** kullanalım.
- Daha sonra bir **DatagridView** kullanalım.



# Yararlanılan Kaynaklar

- Sefer Algan , HER YÖNÜYLE C# , Pusula Yayıncılık, İstanbul, 2003
- Volkan Aktaş, HER YÖNÜYLE C# 5.0 , Kodlab Yayıncılık, İstanbul, 2013
- Milli Eğitim Bakanlığı "Nesne Tabanlı Programlama", 2012

# İyi Çalışmalar...

**Doç. Dr. Deniz Kılınç**

[deniz.kilinc@bakircay.edu.tr](mailto:deniz.kilinc@bakircay.edu.tr)

[drdenizkilinc@gmail.com](mailto:drdenizkilinc@gmail.com)

[www.denizkilinc.com](http://www.denizkilinc.com)