

# NESNE YÖNELİMLİ PROGRAMLAMA(Object Oriented Programming/OOP)

**Öğr. Gör. Celil ÖZTÜRK**

Marmara Üniversitesi

Teknik Bilimler Meslek Yüksekokulu

# İçerik

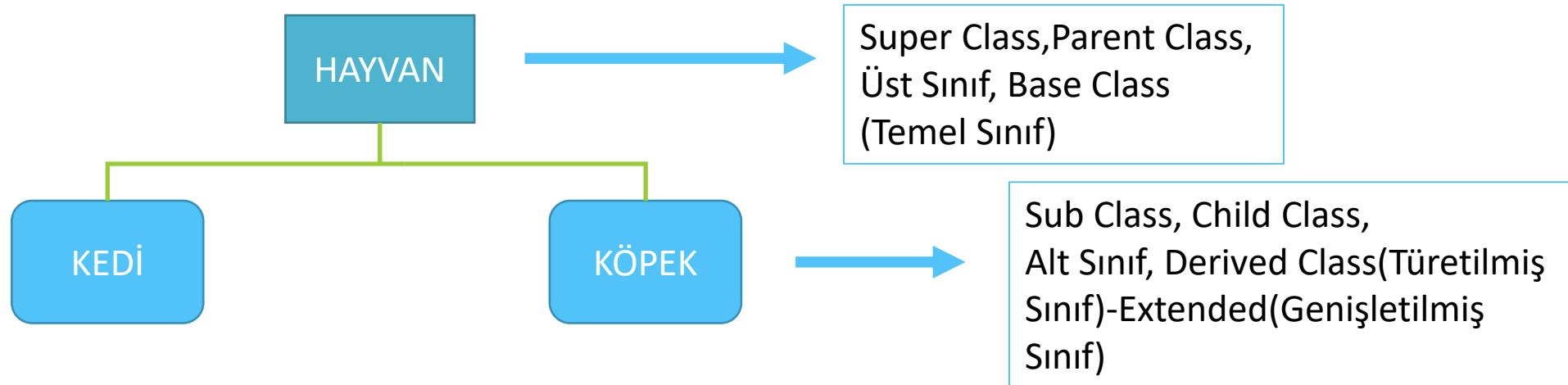
- ✓ Sınıf Kavramı
- ✓ Nesne Kavramı
- ✓ UML Sınıf Diyagramları
- ✓ Nesneye Yönelik Programlamanın Temel İlkeleri
- ✓ Program geliştirmeye giriş

# SINIF(CLASS)

- Sınıflar, verileri tek tek değişkenlerde tutmak yerine, birbirleri ile *ilişkili verileri* bir araya getirmek ve bunlara tek bir **referans** aracılığıyla erişmeyi sağlamaktadır.
- Örnek; Taşıt bir sınıftır. Kişi de bir sınıftır. Çalışan da bir sınıftır. Müşteri de bir sınıftır.
- Ad, soyad, müşteri numarası, çalıştığı departman gibi bilgileri tutmak istediğimiz bir **Müşteri sınıfı** tanımlanabilir.
- Sınıf kavramı, nesneler hakkında bilgi sağladığı için kullanışlıdır.
- Nesneler, özelliklerini ve davranışlarını(metotlar) **Sınıflardan** alır.

# SINIF(CLASS)

- Hayvan sınıfı, kedi ve köpek sınıfındaki ortak özellikleri görememizi sağlar. Hayvan sınıfı, Kedi ve Köpek türleri ortak olarak ad, cinsiyet değerini alabilir. Aynı şekilde farklı hayvan türleri de bu nitelikleri alabilir.
- Hayvan sınıfı, temel alınarak Kedi ve Köpek sınıfları bu sınıftan türetilmelidir.



# NESNE KAVRAMI

- Bir sınıf tanımlandıktan sonra aslında yeni bir veri tipi tanımlanmış olunur. Java dilinde oluşturulan bu yeni veri tipine sahip değişkenlere «nesne» adı verilir.
- Nesne daha genel bir sınıfın somutlaşmış örneğidir.
- Benim arabam, Araba sınıfının somutlaşmış bir örneğidir.
- Bir nesne, bulunduğu sınıfın somutlaşmış bir örneğidir.
- Evdeki hayvanımız, aile bireylerimiz, sahip olduğumuz araçlar, sahip olduğumuz Kripto paralar da birer nesnedir.
- Ayrıca olaylar da birer nesne olabilir.
- Nesneler, özelliklerini ve davranışlarını **Sınıflardan** alırlar.

# NESNE KAVRAMI

- Nesne , niteliklere ve davranışlara sahip somut bir varlıktır.
- Bir nesnenin niteliği onun sahip olduğu özelliklerdir.
- Bir nesnenin davranışları onun yapılabildiği işlemlerdir ve operasyonlardır.
- Örnek,
- Bir Araba sınıfı Nitelik(properties, attribute) olarak renk, fiyat, hız gibi özellikler alabilir.
- Davranış(Behaviours/Methods) olarak Başla(), Dur() davranışlarını alabilir.

# Nesneye Yönelik Programlama !

- Nesneye yönelik 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.
- Gerçek dünya nesnelerden oluştuğundan bu yaklaşım ile sistemin daha gerçekçi bir modeli oluşturulabilir. Program daha anlaşılır olur.

# Nesneye Yönelik Programlama !

- Nesne modellerinin içindeki veriler sadece üye fonksiyonların erişebileceği şekilde düzenlenebilirler. Veri saklama denilen bu özellik sayesinde verilerin herhangi bir fonksiyon tarafından bozulması önlenir.
- Programcılar kendi veri tiplerini yaratabilirler.
- Bir nesne modeli oluşturduktan sonra bu modeli çeşitli şekillerde defalarca kullanmak mümkündür.
- Programları güncellemek daha kolaydır.
- Nesneye dayalı yöntem takım çalışmaları için uygundur.



# UML Sınıf Diyagramları(Unified Modelling Language)

- Nesne-tabanlı yazılım geliştirmede kullanılan standart bir modelleme dilidir.
- Modellerden kod oluşturulabilir (forward engineering).
- Kodlardan model çıkarılabilir (reverse engineering).
- UML, standartlaşmış bir yapı olduğundan dolayı, dili bilenler tarafından okunur ve aynı şekilde yorumlanır.
- Takım çalışmasına birebirdir.
- UML, bir sistemin mimarisini ve detaylarını belgelemede yardımcı olur.
- UML ile gereksinimler daha iyi ifade edilebilir.

# UML Sınıf Diyagramları(Unified Modelling Language)

SINIF ADI
Özellik 1:Tür 1 Özellik 2:....
İslev1 ( ) İslev2 (parametre) İslev3 ():geri dönen değer tipi



Sınıf Adı



Özellikler(Propertyes,Attributes)



Metotlar(Fonksiyonlar)

Ogrenci
+ad: String -ogrenciNo: Integer #dogumTarihi : Date
+BilgiGoruntule ( ) -KayıtEkle (Id): string #Duzenle (): boolean <<Constructor>>+Ogrenci()

## Erişim Belirteci

## Açıklama

Public(UML → +)

Sınıf, değişken veya metota erişim sınırsızdır.

Protected (UML → #)

Sınıf, değişken veya metota erişim bulunduğu sınıf ve bu sınıftan türetilen sınıflar ile sınırlıdır.

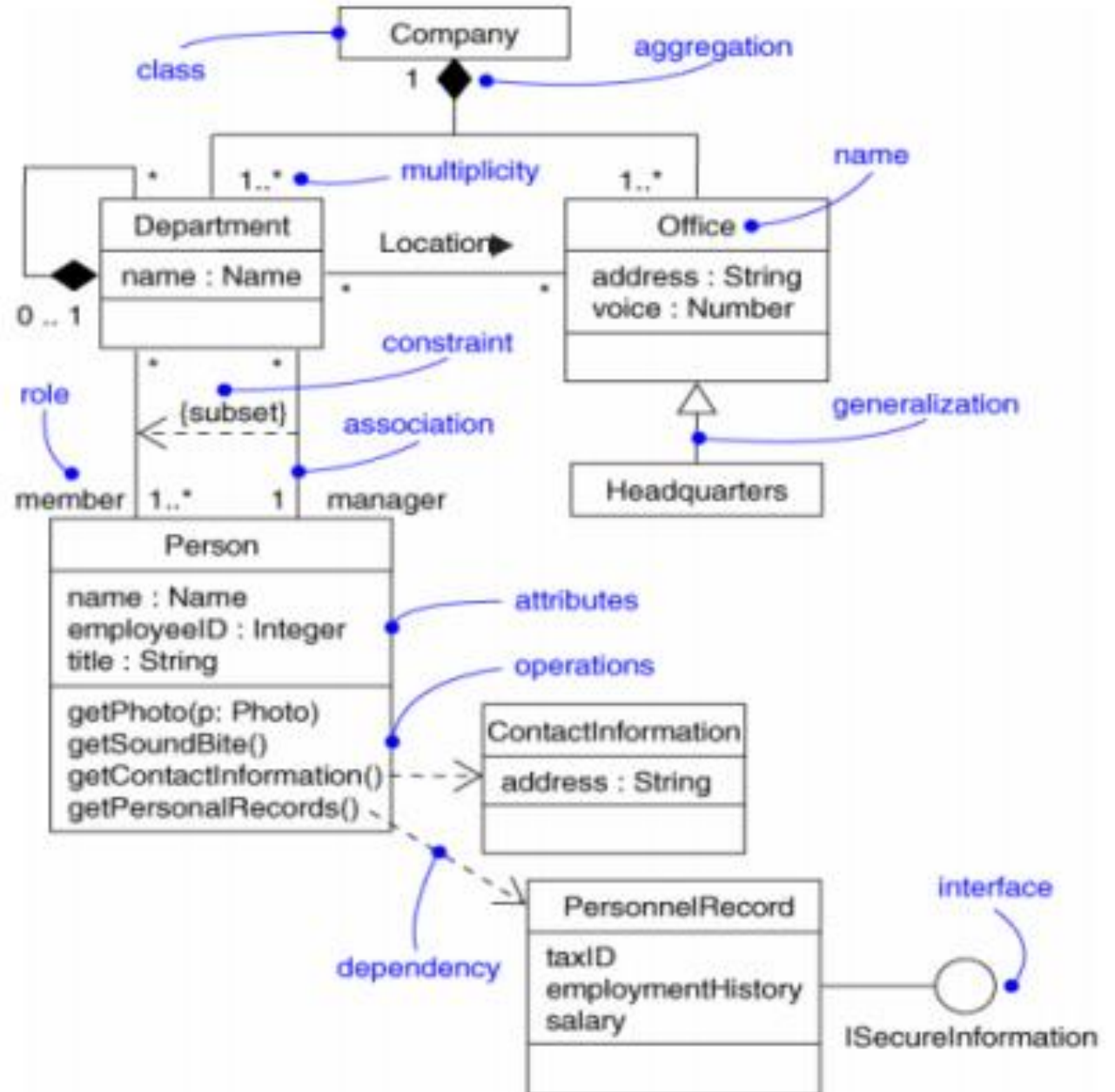
Private (UML → -)

Sınıfa, değişken veya metota erişim ait olduğu sınıf ile sınırlıdır.

Interval (UML → ~)

Sınıfa erişim ait olduğu assembly ile sınırlıdır.

# UML Sınıf Diyagramları(Unified Modelling Language)



# NESNEYE YÖNELİK PROGRAMLAMA İLKELER

- Soyutlama
  - Saklama
- SINIF
- Kalıtım
  - Çok Biçimlilik
- SINIF  
HİYERARŞİSİ

# NESNEYE YÖNELİK PROGRAMLAMA İLKELER

- Büyük programlar yazmayı kolaylaştıran **soyutlama**
- programları değiştirmeyi ve korumayı kolaylaştıran **saklama**
- **Kalıtım** ve **çok biçimliliğe** izin veren ve programları kolayca genişletilebilir kılan **sınıf hiyerarşisi**dir.

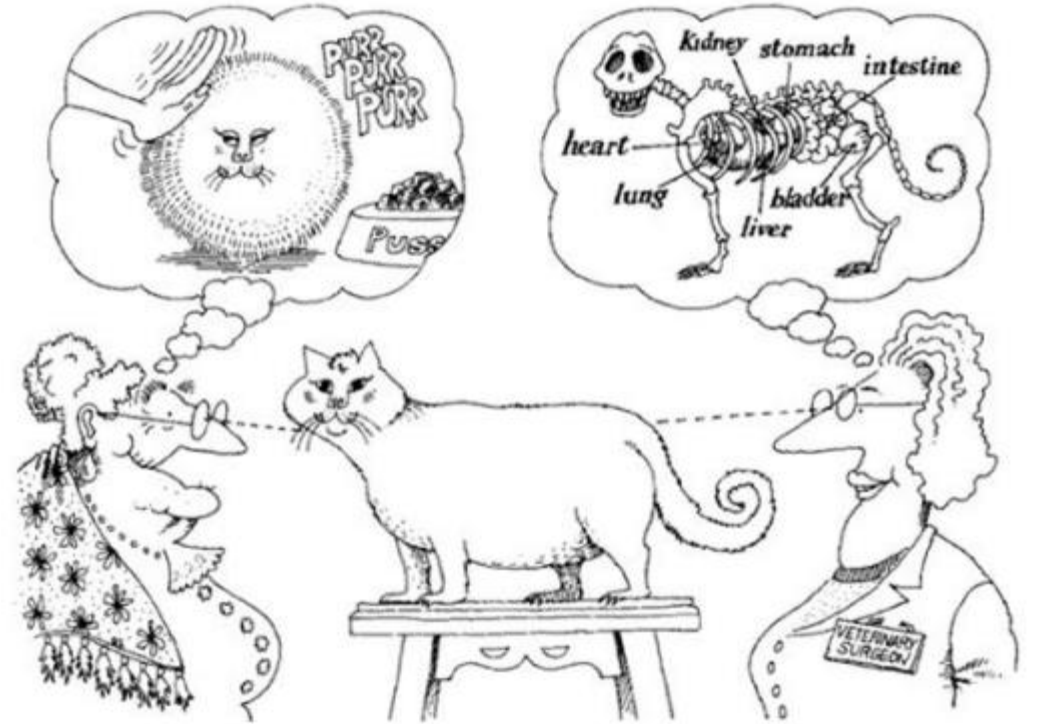
Herhangi bir programlama dilinde bu kavramları uygulayabilirsiniz; fakat, nesneye-yönelik programlama dilleri salt bu amaçla tasarlanmışlardır.

# KALITIM(INHERITANCE)

- Var olan bir sınıftan, sınıfın tüm özelliklerini **kullanabilen** yeni sınıflar türetmek olarak tanımlanabilir.
- Üst sınıfın ayrıntılarının bilinmesine ve kaynak kodun elde edilmesine gerek yoktur.
- Yararları:
  1. **Tekrar Kullanılabilirlik(Reusability !)** : Sistemin genel kısımları önce kodlanır. Daha özel kısımlar, bu genel kısımlardan türetilir. Ortak özelliklerin yeniden yazılmasını gerek kalmaz.
  2. **Çok biçimlilik(Polymorphism)** : ile birlikte kullanıldığında, aynı sınıftan türeyen farklı varlıkların onlara mesaj gönderen nesnelere aynı varlıklar gibi görünmesini sağlar.

# SOYUTLAMA(Abstraction)

- “**Soyutlama**” önemli özelliklere odaklanabilmek için ayrıntıları göz ardı etme sürecidir.
- Geleneksel olarak, bir programlama dili soyutlama yapmaya izin verdiği ölçüde yüksek düzeyli (high-level) kabul edilir.



# SOYUTLAMA(Abstraction)

- Soyutlama, **Veri** ve **Kontrol** soyutlaması olarak yapılır.
- **Veri soyutlaması:** Veri yapıları üzerinde yapılan soyutlamadır. Bir veri tipinin nasıl yapılandırıldığının ayrıntılarını göz ardı edilmesi.(Örn; **yas** değişkenine -10 gibi değerin girilmesi, bu durumun engellenmesi için gerçek bir değerin girilmesinin garanti edilmesi !)
- **Kontrol Soyutlaması:** Veriyi taşıyan değişkenler haricinde; altprogram, yöntem veya işlevlerin soyutlama işlemidir. (Örn, yöntem ile yapılan işlemlerin, bir başka nesne tarafından müdahale edilmeyecek hale getirilmesi.)



# BİLGİ SAKLAMA /SARMALAMA (Encapsulation/Kapsülleme)

- Information Hiding !
- Bir nesne özelliklerinin dışarıdan kullanılmasının sınırlandırılması işlemidir.
- Örnek, **bir aracın deposundaki yakıt miktarını depoya baktığımızda göremeyiz.\* Bunun için araç yakıt gösterge paneli kullanılır.**
- Kapsülleme = Veri Soyutlaması + Kontrol Soyutlaması.
- Sarmalama”, soyutlamayı desteklemek ya da güçlendirmek için bir sınıfın iç yapısının gizlenmesidir.
- Bir sınıfın «arayüzü» o sınıfın ne yapabileceğini, «gerçeklemesi» ise bunu nasıl yapabileceğini gösterir. Gerçek bir sarmalama, verileri fonksiyonlarla gizlemeyi gerektirir.

# BİLGİ SAKLAMA /SARMALAMA (Encapsulation/Kapsülleme)

- Nesnenin içindeki kod, veri veya her ikisi bu nesneye private (özel) veya public (genel) olabilir.
- Private kod ve veriler, sadece o nesneye ait bileşenler tarafından bilinebilir ve erişilebilir. Programında nesneye ait olmayan bileşenleri bu private kod ve verilere erişemez.
- Kod ve verilerimiz public ise, belirli bir nesnenin içinde tanımlansalar da programın diğer parçaları tarafından erişilebilirler.
- Nesnelere ait public elemanları private elemanlara denetlenebilir şekilde erişebilmek amacıyla kullanırız.
- Her ne sebeple olursa olsun, nesne, kullanıcı tarafından tanımlanan tipte bir değişkendir. Nesne tipi tanımladığımızda aslında yeni bir veri tipi oluşturuyoruzdur. Bu veri tipi de bileşik bir değişkendir

# Sınıflar Arası İlişkiler

- ❑ Bağlantı(Association): "has a" Sınıflar arasında hizmet alma-vermeye dayalı bir ilişkidir.
- ❑ Bağ(Link): Nesneler arasındaki bağlantıdır.

## Aggregation(Toplama,Bir araya getirme)

Is part of – Bir parçası olmak,

İçerilen nesneler, kendi başlarına da kullanılırlar.  
Sadece o nesneye ait parçalar değildir.

Gösterim: İçi boş elmas

İçerilen nesneler, kendi başlarına da kullanılırlar.





## Composition(Meydana Gelme)

Is made of – Daha güçlü bir parçası olmak

Birlikte oluşturulup, birlikte yok edilirler.  
Alt parçalar, o nesneyi oluşturmak için  
oluşturulmuşlardır, kendi başlarına kullanılamazlar.

Gösterim: İçi dolu elmas.

# Sınıflar Arası İlişkiler ve UML Gösterimleri

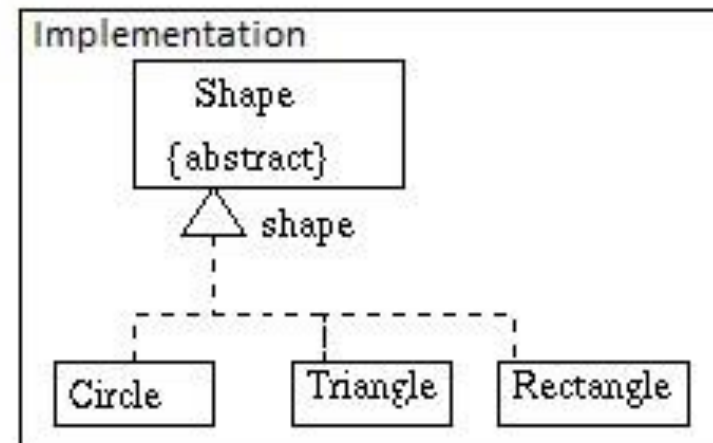
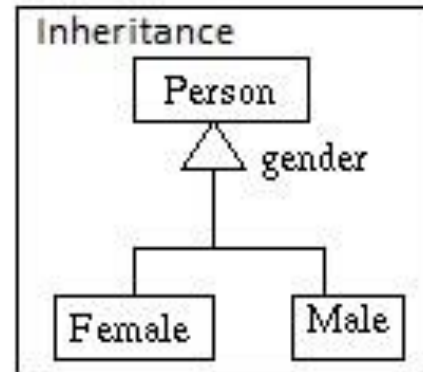
UML symbols	
Association	Symbol
Composition	
Aggregation	
Inheritance	
Implementation	



Composition: every car has an engine.



Aggregation: cars may have passengers, they come a



# Sınıflar Arası İlişkiler

## Sahip Olma (*has a*) İlişkisi (*Aggregation / Composition*)

Bir sınıfın üyeleri sadece hazır veri tipleri (char, int, double....) olmak zorunda değildir. Bir sınıfın üyeleri başka sınıftan tanımlanmış nesneler de olabilirler.

Sahip olma ilişkisinin (*has a relation*) iki ayrı türü vardır:

- **Toplama, bir araya getirme (*Aggregation*)**: İçerilen nesneler (alt parçalar) kendi başlarına da kullanılırlar. Sadece o nesneye ait parçalar değildir.

Örneğin havaalanında uçaklar vardır.



UML sınıf diyagramında bu ilişki ifade edilirken kutucuğun içi boş bırakılır.

- **Meydana gelme (*Composition*)**: Alt parçalar o nesneyi meydana getirmek için oluşturulmuşlardır; kendi başlarına kullanılmazlar.

Örneğin otomobilin motoru vardır.



UML sınıf diyagramında bu ilişki ifade edilirken kutucuğun içi doldurulur.

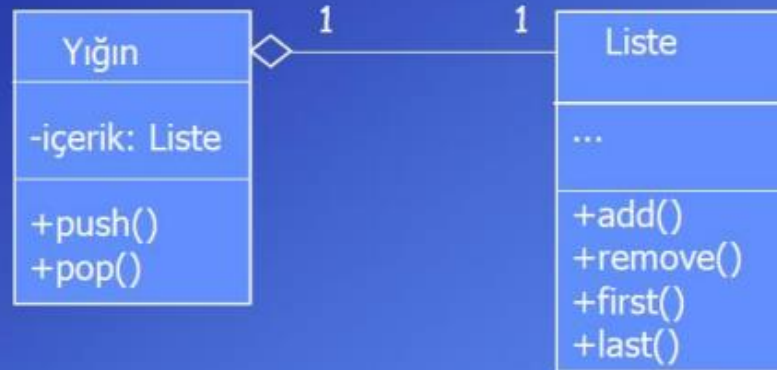
# Temsil(Delegation)

## Temsil (*Delegation*)

Nesneye dayalı programlamanın önemli kavramlarından biri de temsildir (*delegation*).

Bir sınıf/nesne bazı hizmetleri yerine getirmek için başka varlıkları (sınıf/nesne) görevlendirebilir. Diğer bir deyişle kendisini temsil etmesini sağlar.

Örnek: Bir yığın sınıfı tasarlanırken bu sınıfın içine daha önceden varolan bir liste sınıfı yerleştirilebilir.



Yığın sınıfı push işlemi için Liste sınıfının add hizmetini kullanacaktır.

Temsil görevli nesne bulununcaya kadar birden fazla aşama sürebilir.

# Polymorphism(Çok Biçimlilik)

- NYP' da ise oluşturulan nesnelerin gerektiğinde başka bir nesne gibi davranabilmesine denir.
- Çok biçimlilikle programdaki her nesne kendi davranışını değiştirmeden, kalıtım hiyerarşisine göre farklı biçimlerde görülebilir.
- Genel anlamı ile bir adın birbiriyle alakalı fakat teknik açıdan farklı iki veya daha fazla amaç için kullanılabilmesi yeteneğidir.

# JAVA Program Yapısı

- Bir Java programı, bir ya da birden fazla sınıftan oluşur.
- Bir sınıf, bir ya da birden fazla metod içerir.
- Bir metod, program deyimlerini içerir.
- Kural 1. Java kaynak programlarının uzantısı .java olmalıdır.
- Kural 2. Programa verilen dosya adı ile programda tanımlanan class adı aynı olmalıdır.

**YANLIŞ**

Kaynak kod dosyası "myTest.java" olarak kaydedilmiş olsun,

Sınıf tanımı da aşağıdaki gibi olsun:

```
class test { ... }
```



# JAVA Program Yapısı

- Kural 3. Her deyimin sonuna (;) konulmalıdır.
- Kural 4. Derleyicinin hata iletisi, kaynak programdaki hatayı kategorik olarak bildirir ve yerini işaret eder. Bu nedenle hata belirtilen yerde aranmalıdır.
- Kural 5. Bir java uygulaması her zaman main olarak isimlendirilen metodu içermelidir. Çünkü main metodu program akışını denetler

# JAVA Program Yapısı

```
public class MyProgram
```

```
{
```

sınıf başlığı

sınıf gövdesi

Açıklamalar neredeyse her yerde yer alabilir.

```
}
```

# JAVA Program Yapısı

```
// class yani sınıf hakkında açıklamalar
public class MyProgram
{
    // metod hakkında açıklamalar
    public static void main (String[] args)
    {
    }
}
```

metod başlığı

metod gövdesi

# Programa Açıklama Ekleme

- Bir kaynak programın yıllar sonra programcısı değişmiş olsa bile, program üzerinde güncelleme yapılmasını kolaylaştıran ayrıntıların tanımlandığı, satır içi belgelere Açıklama denir. Programın çalışmasına herhangi bir etkisi bulunmamaktadır.
- `//` Satır sonuna kadar yazılanları derleyici işlemez
- `/*` Çoklu satır açıklamaları için kullanılan bu işaretlerin arasındakiler derleyici tarafından işlenmez `*/`  
`/**`
  - \* Bu sentaks biçimi Javadoc açıklamaları için
  - \* kullanılır,
  - \* HTML biçimlendirme özelliklerini sunar.`*/`

# JAVA Adlandırma Kuralları

- Değişken adları bir harf, (\_) ya da (\$)simgesiyle başlamalıdır. Digit ile başlayamaz.
- (\_) karakteri ile başlayan adların ikinci karakterleri harf olmalıdır.
- Java büyük küçük harf ayrımı yapar. SicilNo ile sicilNo derleyici tarafından farklı adlar olarak görülür ve farklı bellek alanları ayrılır.
- Değişken, metod ve arayüz adları küçük harfle, sınıf adlarını ise büyük harfle başlatmak bir gelenektir. Yine sabitleri büyük harfle yazmakta bu geleneğe dahildir. Örneğin;  
title case for class names - Lincoln  
upper case for constants - MAXIMUM

# JAVA Adlandırma Kuralları

- Kullanılan ad birden çok sözcükten oluşuyorsa, sözcükler arasına (\_) konmalı ya da ilk sözcükten sonrakileri büyük harfle başlatmalıdır.
- Kullanılacak karakter sayısı için Java'nın bir sınırlaması yoktur. Ancak kullanılan dosya sisteminin limitlerine uyulmalıdır.
- FAT 32 ya da NTFS dosya sistemini kullanan Windows işletim sisteminde yol (path) uzunluğu 260 karakterdir.
- Aynı kapsama (scope) alanında olan değişkenlere farklı isimler verilmelidir. Ancak istenirse iki metodun yerel değişkenleri aynı ismi alabilir.
- Reserve edilmiş olan sözcükler tanımlayıcı ad olarak kullanılamaz.

# JAVA'da REZERVE EDİLMİŞ SÖZCÜKLER

- abstract boolean break byte case catch char class const continue default  
do double else enum extends false final finally float for goto if  
implements import instanceof int interface long native new null package  
private protected public return short static strictfp super switch  
synchronized this throw throws transient true try void volatile while

# JAVA'da Boşluk

- Java'da sekme ve boşluk sadece bir boşluk karakteri olarak geçerlidir. Fazlası derleyici tarafından gözardı edilir.
- Programda verilen girinti ve boşluk ile yapılan düzenlemeler, Java derleyicisi tarafından önemsenmez, ancak programın anlaşılabilirliği ve geliştirilebilirliği açısından önemlidir.



# Kaynaklar

- Yrd. Doç. Dr. Deniz KILINÇ-Celal Bayar Üniversitesi / Nesneye Yönelik Programlama Ders Notları
- Yrd. Doç. Dr. Erbil Akbay –Marmara Üniversitesi / Object Oriented Programming 1-2 Ders notları
- YAZILIM GEREKSİNİM ANALİZİ (SOFTWARE REQUIREMENTS ANALYSIS) Yrd. Doç. Dr. Fatih YÜCALAR
- [www.yazilimcildunyasi.com](http://www.yazilimcildunyasi.com)
- NESNEYE YÖNELİK PROGRAMLAMA Yrd.Doç.Dr. Zeynep ORMAN [ormanz@istanbul.edu.tr](mailto:ormanz@istanbul.edu.tr)
- Nesne Yönelimli Programlama I 2. Ders Nesneye Yönelik Programlama Temel İlkeleri ve Java'ya Giriş Dr. Öğr. Üyesi Aysun ALTIKARDEŞ
- [https://medium.com/@tugrulbayrak/uml-class-diagramlari-4c3bb7e9cc4c#:~:text=UML\(Unified%20Modeling%20Language\)%20nedir,Yaz%C4%B1l%C4%B1%20bir%20dil%20de%C4%9Fildir.&text=Structure%20Diagrams\(Yap%C4%B1sal%20Diyagramlar\)%2C,sistemde%20bulunmas%C4%B1%20gerekenler%20i%C3%A7in%20kullan%C4%B1l%C4%B1r.](https://medium.com/@tugrulbayrak/uml-class-diagramlari-4c3bb7e9cc4c#:~:text=UML(Unified%20Modeling%20Language)%20nedir,Yaz%C4%B1l%C4%B1%20bir%20dil%20de%C4%9Fildir.&text=Structure%20Diagrams(Yap%C4%B1sal%20Diyagramlar)%2C,sistemde%20bulunmas%C4%B1%20gerekenler%20i%C3%A7in%20kullan%C4%B1l%C4%B1r.)
- [https://medium.com/@tugrulbayrak/uml-class-diagramlari-4c3bb7e9cc4c#:~:text=UML\(Unified%20Modeling%20Language\)%20nedir,Yaz%C4%B1l%C4%B1%20bir%20dil%20de%C4%9Fildir.&text=Structure%20Diagrams\(Yap%C4%B1sal%20Diyagramlar\)%2C,sistemde%20bulunmas%C4%B1%20gerekenler%20i%C3%A7in%20kullan%C4%B1l%C4%B1r.](https://medium.com/@tugrulbayrak/uml-class-diagramlari-4c3bb7e9cc4c#:~:text=UML(Unified%20Modeling%20Language)%20nedir,Yaz%C4%B1l%C4%B1%20bir%20dil%20de%C4%9Fildir.&text=Structure%20Diagrams(Yap%C4%B1sal%20Diyagramlar)%2C,sistemde%20bulunmas%C4%B1%20gerekenler%20i%C3%A7in%20kullan%C4%B1l%C4%B1r.)
- <https://itexpertsconsultant.wordpress.com/2016/04/22/difference-between-associationaggregationcomposition-and-inheritance/>
- <https://gelecegiyazanlar.turkcell.com.tr/konu/android/egitim/android-101/nesne-ve-nesne-yonelimli-programlama-teorisinin-4-temel-ozelligi>