# OOP

Nesneye Dayalı Analiz

# Tarihçe

Nesneye yönelik paradigma, yeni bir programlama yaklaşımı ilk konseptinden şeklini alırken, tasarım ve analiz yöntemlerine ilgi çok daha sonra geldi.

- İlk nesne yönelimli dil, 1960 yılında Norveç Bilgi İşlem Merkezi'ndeki araştırmacılar tarafından geliştirilen Simula (gerçek sistemlerin simülasyonu) idi.
- 1970 yılında, Alan Kay ve Xerox PARK'daki araştırma grubu, Dynabook'u programlamak için Dynabook ve ilk saf nesne yönelimli programlama dili (OOPL) - Smalltalk adlı bir kişisel bilgisayar yarattı.
- 1980'lerde Grady Booch, programlama dili Ada için bir tasarım sunan ve Nesne Yönelimli Tasarım başlıklı bir makale yayınladı. Bundan sonraki baskılarda fikirlerini eksiksiz bir nesne yönelimli tasarım yöntemine genişletti.

- 1990'larda, Coad nesne yönelimli yöntemlere davranışsal fikirleri dahil etti.
- Diğer önemli yenilikler James Rumbaugh'ın Nesne Modelleme Teknikleri (OMT) ve Ivar Jacobson'ın Nesne Yönelimli Yazılım Mühendisliği (OOSE) idi.

# Nesneye Dayalı Analiz

- Nesneye Yönelik Analiz (OOA), etkileşimli nesnelerden oluşan bir yazılım sisteminin nesne modeli açısından yazılım mühendisliği gereksinimlerini belirleme ve yazılım özelliklerini geliştirme prosedürüdür.
- Nesne yönelimli analiz ile diğer analiz biçimleri arasındaki temel fark, nesne yönelimli yaklaşımda gereksinimlerin hem verileri hem de işlevleri birleştiren nesnelerin etrafında düzenlenmesidir. Sistemin etkileşimde olduğu gerçek dünya nesnelerinden modellenmiştir. Geleneksel analiz metodolojilerinde, iki özellik - fonksiyonlar ve veriler - ayrı olarak ele alınır.

- Grady Booch, OOA'yı “Nesne yönelimli analiz, problem alanının kelime haznesinde bulunan sınıflar ve nesneler açısından gereksinimleri inceleyen bir analiz yöntemidir” olarak tanımlamıştır.

  Nesne yönelimli analizde birincil görevler (OOA) -

- Nesneleri belirleme
- Nesne modeli diyagramı oluşturarak nesneleri düzenleme
- Nesnelerin iç kısımlarını veya nesne niteliklerini tanımlama
- Nesnelerin davranışını, yani nesne eylemlerini tanımlama
- Nesnelerin nasıl etkileşime girdiğini açıklamak

  OOA'da kullanılan ortak modeller kullanım durumları ve nesne modelleridir.

# Nesneye Dayalı Tasarım

- Nesneye Yönelik Tasarım (OOD), nesne yönelimli analiz sırasında üretilen kavramsal modelin uygulanmasını içerir. OOD'da, teknolojiden bağımsız olan analiz modelindeki kavramlar uygulama sınıfları üzerinde haritalanır, kısıtlamalar belirlenir ve arayüzler tasarlanır, böylece çözüm alanı için bir model ortaya çıkar, yani sistemin nasıl olacağına dair ayrıntılı bir açıklama yapılır. somut teknolojiler üzerine inşa edilmiştir.

  Uygulama detayları genellikle şunları içerir –
- Sınıf verilerinin yeniden yapılandırılması (gerekirse),
- Yöntemlerin, yani iç veri yapılarının ve algoritmaların uygulanması,
- Kontrolün uygulanması ve
- Ortaklıkların uygulanması.

  Grady Booch, nesneye yönelik tasarımı “nesneye yönelik ayrıştırma sürecini kapsayan bir tasarım yöntemi ve tasarımdaki sistemin hem statik hem de dinamik modellerinin hem mantıksal hem de fiziksel olarak tasvir edilmesi için bir gösterim” olarak tanımlamıştır.

# Nesne yönelimli programlama

- Nesneye yönelik programlama (OOP), modülerlik ve yeniden kullanılabilirliğin avantajlarını birleştirmeyi amaçlayan nesnelere (hem veri hem de yöntemlere sahip) dayalı bir programlama paradigmasıdır. Genellikle sınıf örnekleri olan nesneler, uygulamaları ve bilgisayar programlarını tasarlamak için birbirleriyle etkileşimde bulunmak için kullanılır.
- Nesneye yönelik programlamanın önemli özellikleri –
  - Program tasarımında aşağıdan yukarıya yaklaşım
  - Nesneler etrafında düzenlenen, sınıflar halinde gruplandırılmış programlar
  - Nesnenin verileri üzerinde çalışacak yöntemlerle verilere odaklanın
  - Nesneler arasındaki fonksiyonlar aracılığıyla etkileşim
  - Mevcut sınıflara özellikler ekleyerek yeni sınıflar oluşturarak tasarımın yeniden kullanılabilirliği
- Nesne yönelimli programlama dillerinin bazı örnekleri C ++, Java, Smalltalk, Delphi, C #, Perl, Python, Ruby ve PHP'dir.

  Grady Booch, nesne yönelimli programlamayı “programların, her biri bir sınıfın örneğini temsil eden ve tüm sınıfları devralma ilişkileri yoluyla birleşmiş bir sınıf hiyerarşisinin üyesi olan, işbirliğine dayalı nesne koleksiyonları olarak organize ettiği bir uygulama yöntemi olarak tanımlamıştır. ”.

# Nesneler ve Sınıflar

- Nesneler ve sınıflar kavramları içsel olarak birbiriyle bağlantılıdır ve nesne yönelimli paradigmanın temelini oluşturur.

# Object-Nesne

- Bir nesne, fiziksel ya da kavramsal bir varlığa sahip olabilen nesne yönelimli bir ortamda gerçek dünyadaki bir unsurdur. Her nesnenin şu özellikleri vardır –
  - Onu sistemdeki diğer nesnelerden ayıran kimlik.
  - Bir nesnenin karakteristik özelliklerini ve aynı zamanda nesnenin sahip olduğu özelliklerin değerlerini belirleyen durum.
  - Nesnenin durumundaki değişiklikler bakımından gerçekleştirdiği harici olarak görülebilen etkinlikleri temsil eden davranış.
- Nesneler, uygulamanın ihtiyaçlarına göre modellenebilir. Bir nesne, müşteri, araba, vb. Gibi fiziksel bir varlığa sahip olabilir; veya proje, süreç vb. gibi maddi olmayan bir kavramsal varlık.

# Class - Sınıf

- Bir sınıf, ortak davranış sergileyen aynı karakteristik özelliklere sahip nesneler koleksiyonunu temsil eder. Ondan yaratılabilecek nesnelerin planını veya açıklamasını verir. Bir nesnenin sınıfın bir üyesi olarak yaratılmasına anında örnekleme denir. Böylece nesne bir sınıf örneğidir.

Bir sınıfın bileşenleri -

- Sınıftan başlatılacak nesneler için bir öznitelik kümesi. Genel olarak, bir sınıfın farklı nesneleri, niteliklerin değerlerinde bir miktar farklılığa sahiptir. Nitelikler genellikle sınıf verileri olarak adlandırılır.

- Sınıftaki nesnelerin davranışını gösteren bir işlemler kümesi. İşlemlere ayrıca işlevler veya yöntemler de denir.

# Örnek

- İki boyutlu bir uzayda geometrik şekil dairesini temsil eden basit bir sınıf, Çember olarak düşünelim. Bu sınıfın özellikleri şu şekilde tanımlanabilir -

  x – koordinat, merkezin x koordinatını belirtir
  y – koordinat, merkezin y-koordinatını belirtir
  a, dairenin yarıçapını belirtmek için

  Operasyonlarından bazıları aşağıdaki gibi tanımlanabilir -

  findArea (), alanı hesaplama yöntemi
  findCircumference (), çevreyi hesaplama yöntemi
  scale (), yarıçapı artırma veya azaltma yöntemi

  Örnekleme sırasında, en azından bazı nitelikler için değerler atanır. My_circle adlı bir nesne yaratırsak, durumunu göstermek için x-coord: 2, y-coord: 3 ve a: 4 gibi değerler atayabiliriz. Şimdi, eğer işlem scale () ölçeklendirme faktörü 2 olan my_circle üzerinde gerçekleştirilirse, a değişkeninin değeri 8 olur. Bu işlem my_circle durumunda bir değişiklik getirir, yani nesne belirli davranışlar sergiler.

# Encapsulation and Data Hiding

- Kapsülleme ve Veri Gizleme

# Encapsulation - Kapsülleme

- Kapsülleme, bir sınıf içinde hem nitelikleri hem de metotları birleştirme işlemidir. Kapsülleme yoluyla, bir sınıfın iç detayları dışarıdan gizlenebilir. Sınıftaki öğelere yalnızca sınıf tarafından sağlanan arabirim aracılığıyla dışarıdan erişilmesine izin verir.

# Data Hiding – Veri Gizleme

- Tipik olarak, bir sınıf, verilerine (niteliklerine) yalnızca sınıf yöntemleriyle erişilebilecek ve doğrudan dış erişimden izole edilebilecek şekilde tasarlanmıştır. Bir nesnenin verilerini yalıtmak için bu işleme veri gizleme veya bilgi gizleme denir.

# Örnek

- Circle sınıfında, veri gizleme, sınıf dışından görünmez nitelikler sağlayarak ve sınıf verilerine erişmek için sınıfa iki yöntem daha ekleyerek dahil edilebilir.

  setValues (), x-coord, y-coord ve a'ya değer atama yöntemi
  getValues (), x-coord, y-coord ve a değerlerini alma yöntemi

  Burada, my_circle nesnesinin özel verilerine, Circle sınıfı içine alınmamış herhangi bir yöntemle doğrudan erişilemez. Bunun yerine setValues () ve getValues () yöntemleriyle erişilebilir.

# İleti geçişi - İleti geçişi

- Herhangi bir uygulama, uyumlu bir şekilde etkileşime giren birkaç nesne gerektirir. Bir sistemdeki nesneler mesaj iletmeyi kullanarak birbirleriyle iletişim kurabilir. Bir sistemin iki nesnesi olduğunu varsayalım: obj1 ve obj2. Nesne1 nesne2'nin yöntemlerinden birini yürütmek istiyorsa nesne2'ye bir mesaj gönderir.

  Mesaj iletmenin özellikleri :
- İki nesne arasında geçen mesaj genellikle tek yönlüdür.
- İleti iletme, nesneler arasındaki tüm etkileşimleri sağlar
- Mesaj iletme esasen sınıf yöntemlerini çağırmayı içerir
- Farklı süreçlerdeki nesneler mesaj aktarımında rol oynayabilir.

# Inheritance – Miras - Kalıtım

- Kalıtım, yeteneklerini artırarak ve geliştirerek mevcut sınıflardan yeni sınıflar yaratılmasına izin veren mekanizmadır. Mevcut sınıflara temel sınıflar / ebeveyn sınıfları / süper sınıflar, yeni sınıflar ise türetilmiş sınıflar / alt sınıflar / alt sınıflar olarak adlandırılır. Alt sınıf, süper sınıfın izin verdiği sürece, süper sınıfın özelliklerini ve yöntemlerini miras alabilir veya türetebilir. Ayrıca, alt sınıf kendi niteliklerini ve yöntemlerini ekleyebilir ve süper sınıf yöntemlerin herhangi birini değiştirebilir. Kalıtım “bir - bir” (“is – a”) ilişkisini tanımlar.

# Örnek

- Bir memeli sınıfından, İnsan, Kedi, Köpek, İnek vb. Gibi çeşitli sınıflar türetilebilir. İnsanlar, kediler, köpekler ve inekler, memelilerin kendine has özelliklerine sahiptir. Ek olarak, her birinin kendine özgü özellikleri vardır. Bir ineğin "is - a" memeli olduğu söylenebilir.

# Miras Çeşitleri

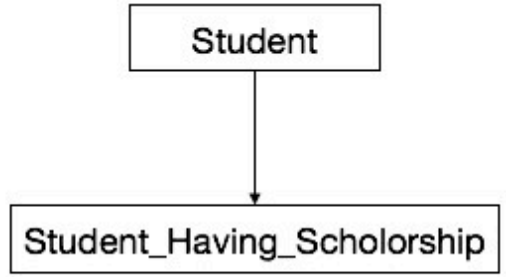
**Tek Kalıtım** - Bir alt sınıf, tek bir süper sınıftan türetilir.

**Çoklu Kalıtım** - Bir alt sınıf birden fazla süper sınıftan türemiştir.

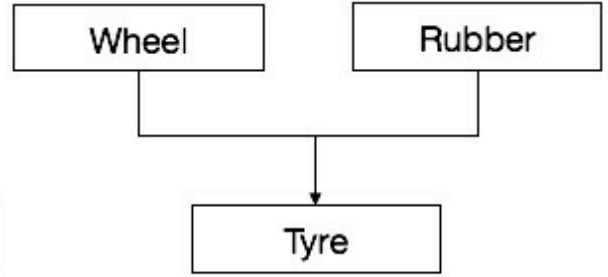
**Çok Düzeyli Kalıtım** - Bir alt sınıf, sırayla başka bir sınıftan türetilen bir üst sınıftan türer.

**Hiyerarşik Miras** - Bir sınıf, her biri bir ağaç yapısını oluşturmak için devam eden bir takım alt seviyelere sahip olabilecek birkaç alt sınıfa sahiptir.

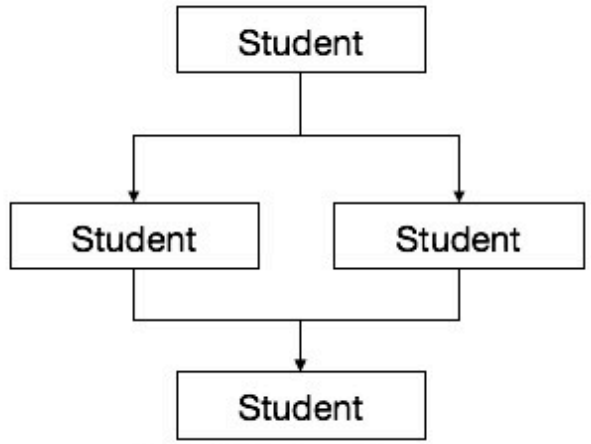
**Hibrit Kalıtım** - Bir kafes yapısı oluşturacak şekilde çoklu ve çok seviyeli kalıtımın bir kombinasyonu.



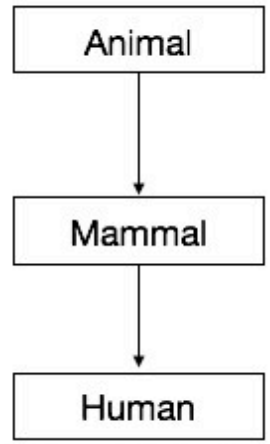

***Single Inheritance***



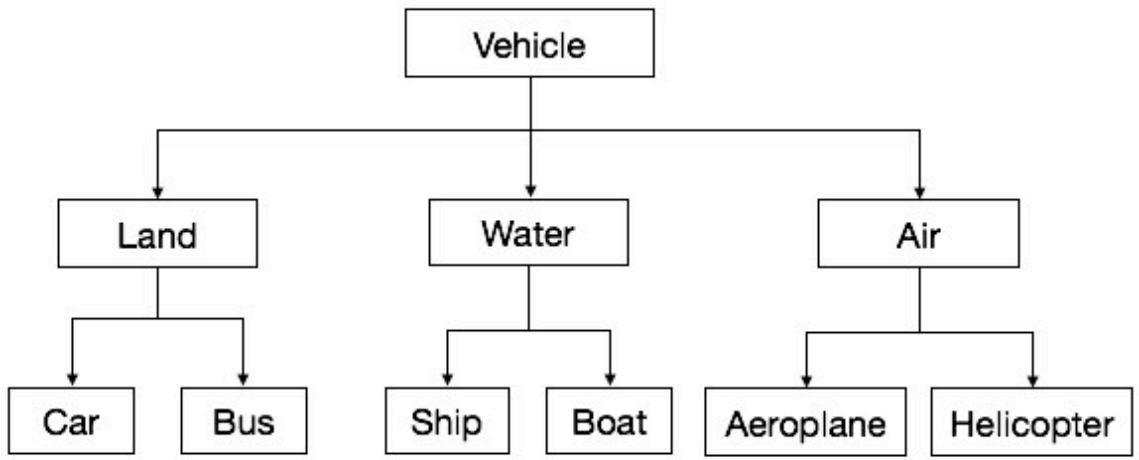

***Multiple Inheritance***




***Hybrid Inheritance***




***Multi-level Inheritance***




***Hierarchical Inheritance***

# Polymorphism

- Polimorfizm, aslında birden fazla form alma yeteneği anlamına gelen Yunanca bir kelimedir. Nesneye yönelik paradigmada, polimorfizm, üzerinde çalıştıkları örneğe bağlı olarak işlemleri farklı şekillerde kullanmak anlamına gelir. Polimorfizm, farklı iç yapıya sahip nesnelerin ortak bir dış arabirime sahip olmasını sağlar. Polimorfizm, kalıtımın uygulanması sırasında özellikle etkilidir.

# Örnek

- Her biri findArea () yöntemiyle birlikte, Circle ve Square olmak üzere iki sınıf düşünelim. Sınıflardaki yöntemlerin adı ve amacı aynı olsa da, iç uygulama, yani, alan hesaplama prosedürü her sınıf için farklıdır. Circle sınıfı bir nesne, findArea () yöntemini çağırdığında, işlem, çemberin alanını, Square sınıfının findArea () yöntemiyle çakışma olmadan bulur.

# Genelleme ve Özelleştirme

- Genelleme ve Özelleştirme, alt sınıfların süper sınıflardan miras aldığı sınıflar arasındaki ilişkiler hiyerarşisini temsil eder.

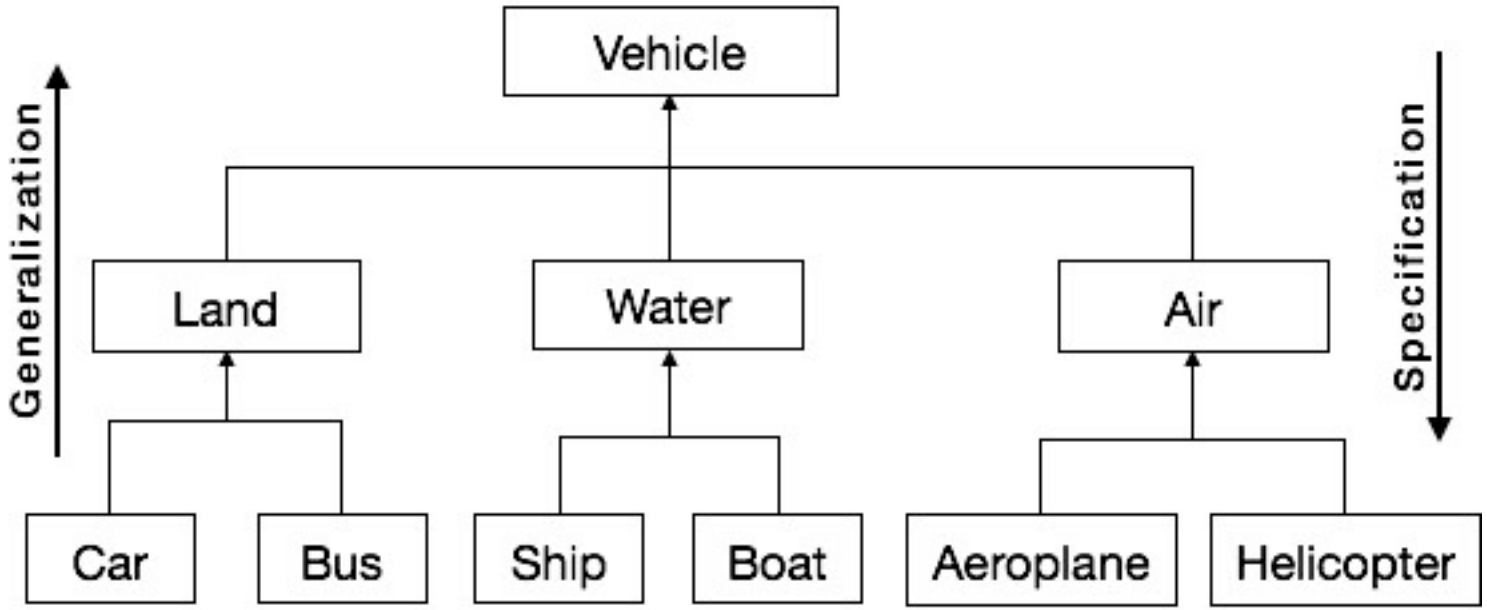
# Genelleme

- Genelleme sürecinde, sınıfların ortak özellikleri daha yüksek bir hiyerarşi düzeyinde bir sınıf oluşturmak için birleştirilir, yani genelleştirilmiş bir süper sınıf oluşturmak için alt sınıflar birleştirilir. “Bir - bir - tür” (“is – a – kind – of”) ilişkisini temsil eder. Örneğin, “araba bir tür kara aracıdır” veya “gemi bir tür su aracıdır”.

# Özelleştirme

- Özelleştirme, genellemenin tam tersi işlemidir. Burada nesne gruplarının ayırt edici özellikleri, mevcut sınıflardan özel sınıflar oluşturmak için kullanılır. Alt sınıfların süper sınıfın özel versiyonları olduğu söylenebilir.

# Aşağıdaki şekilde genelleme ve özelleşme örneği gösterilmektedir.



# Bağlantı ve İlişki

- Bağlantı ve ilişki, nesneler ve sınıflar arasındaki ilişkiyi kurmanın yollarıdır. İlişkilendirme, sınıfla ilişki kurarken, bağlantılar nesnelere karşılık gelir.

# Bağlantı

- Bağlantı, bir nesnenin diğer nesnelerle işbirliği yaptığı bir bağlantıyı temsil eder. Rumbaugh, bunu “nesneler arasında fiziksel veya kavramsal bir bağlantı” olarak tanımladı. Bir bağlantı yoluyla, bir nesne yöntemleri çağırabilir veya başka bir nesne arasında gezinebilir. Bir bağlantı iki veya daha fazla nesne arasındaki ilişkiyi gösterir.

# İlişki

- İlişki, ortak yapıya ve ortak davranışa sahip bir bağlantılar grubudur. İlişki, bir veya daha fazla sınıfın nesneleri arasındaki ilişkiyi gösterir. Bir bağlantı, bir ilişki örneği olarak tanımlanabilir.