

# Veritabanı Yönetim Sistemleri

---

**Dr. Öğr. Üyesi Ahmet Arif AYDIN**

**UML**

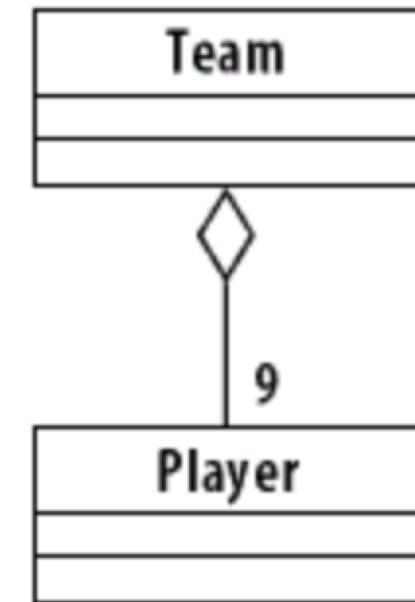
# Bölüm Soruları

---

- ER modeli nedir?
- Hangi amacı gerçekleştirmek için kullanılır?
- Varlık seti nedir?
- İlişki seti (relationship set) nedir?
- Nitelik nedir? Nitelik çeşitleri nelerdir?
  - Multivalued, derived, atomic
- Toplam katılım, zayıf varlıklar, kalıtım, is-a

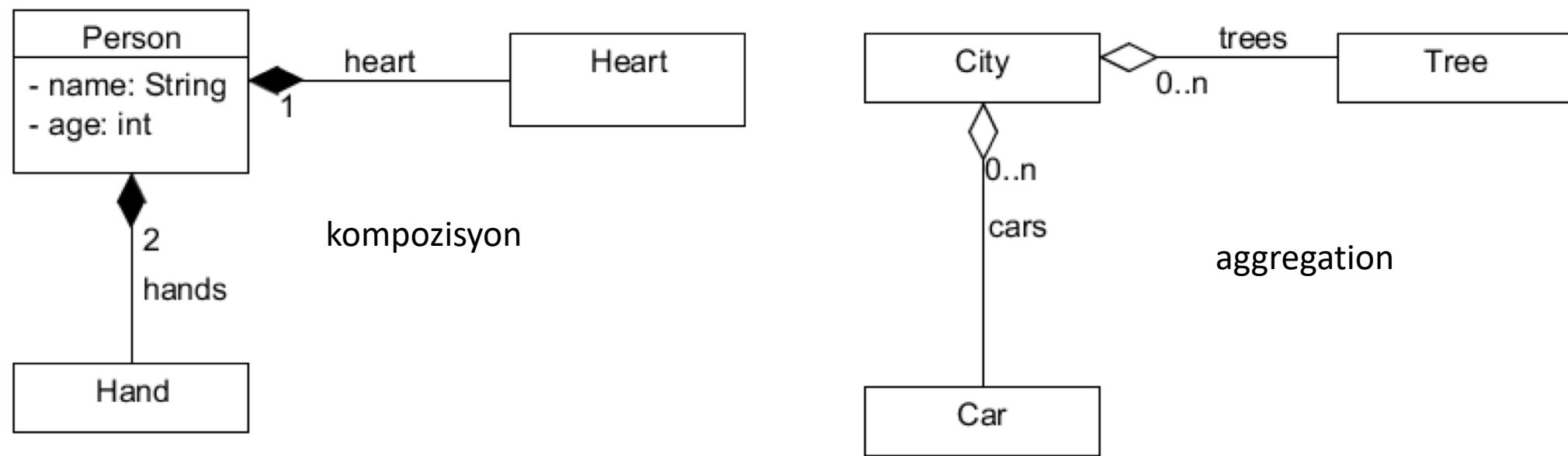
# Kümeleme (aggregation)

- Kümeleme varlıklar arasında 'has-a' ilişkisini bulunmaktadır.
- Parça(part) bütün(whole) ilişkisini ifade eder.
- Parça ve bütün arasında güçlü bir ilişki yoktur.
- Parçalar ayrı olarak başka bir sistem içinde bulunabilir
- Araba ve parçaları (motor, aynalar, tekerler, ...)



# Kompozisyon (Composition)

- Varlıklar arasında bulunan ayrılmaz ve **güclü** bir parça(part) bütün(whole) ilişki kompozisyon ile tanımlanır
- Parçalar ayrılamazlar ve başka bir sistem içinde düşünülemezler



# ER Model ile Kavramsal Tasarım: Problemler

---

ER Modelinin tasarım aşamasında aşağıdaki sorular ortaya çıkabilir:

- Bir kavram **varlık** (entity) olarak mı yoksa **nitelik** (attribute) olarak mı tanımlanmalı?
- Bir kavram **varlık** (entity) olarak mı yoksa **ilişki** (relation) olarak mı tanımlanmalı?
- ER modelinde bulunacak olan varlık ve ilişki setleri hangileridir?

# ER Model ile Kavramsal Tasarım

---

**Belirlenen ihtiyaçlar çerçevesinde tasarım yapılır**

- Bir varlık setinin nitelikleri belirlenirken ihtiyaca göre bir nitelik varlık olarak tanımlanabilir.
- bir akademik personelin ikametgah adresi için tek bir nitelik yetebilirken
  - detaylı aramaların yaptırılabilmesi için **address** varlık seti oluşturulabilir
  - posta kodu, şehir, mahalle ve başka istenilen nitelikler eklenebilir
  - **address** varlık seti akademik personel varlık seti ile ilişkilendirilir.

# ER Model ile Kavramsal Tasarım

---

- Farklı kişiler tarafından aynı problemin çözümü için farklı tasarımlar ve farklı ER Modelleri sunabilirler.
- ER model tasarımında ön plana çıkan hedefler:
  - veritabanına kaydedilecek bilgilerin tekrarlanması
  - veriye zamanında hızlı bir biçimde erişim
  - sorguların makul zaman içerisinde tamamlanması

# ER Model ile Kavramsal Tasarım

---

- Büyük firmalarda yapılan tasarımlar birden fazla kişinin çalışmasını (geliştirici-DB yöneticisi- Kullanıcılar) gerektirir.
- Zaman içerisinde tespit edilen tasarım eksiklileri tekrarlamalı (iterative) olarak giderilir.
- Yüksek seviyeli tasarım farklı uzmanlık alanlarına sahip olan kişilerin anlayabileceği üst seviyede olmalıdır ki her bir katılımcı kendi bilgi birikimi sayesinde tasarıma katkı sağlayabilsin.

# UML: Unified Modeling Language

---

A picture is worth a thousand words !

- 'Bir resim binlerce kelime değerindedir'
- UML'in önemini ve işlevsellliğini açıklamaktadır
- UML yazılım sistemlerinden öte diğer sistemlerin tasarım aşamalarında da kullanılabilir.

# UML: Unified Modeling Language

---

Bir yazılım sisteminin

tasarım ihtiyaçlarını tanımlama aşamasından

teslimatına kadar gecen evrelerde yazılımı

*tanımlamak, görselleştirmek ve dokumantasyon için*

yaygın olarak kullanılan standart dil UML'dir.

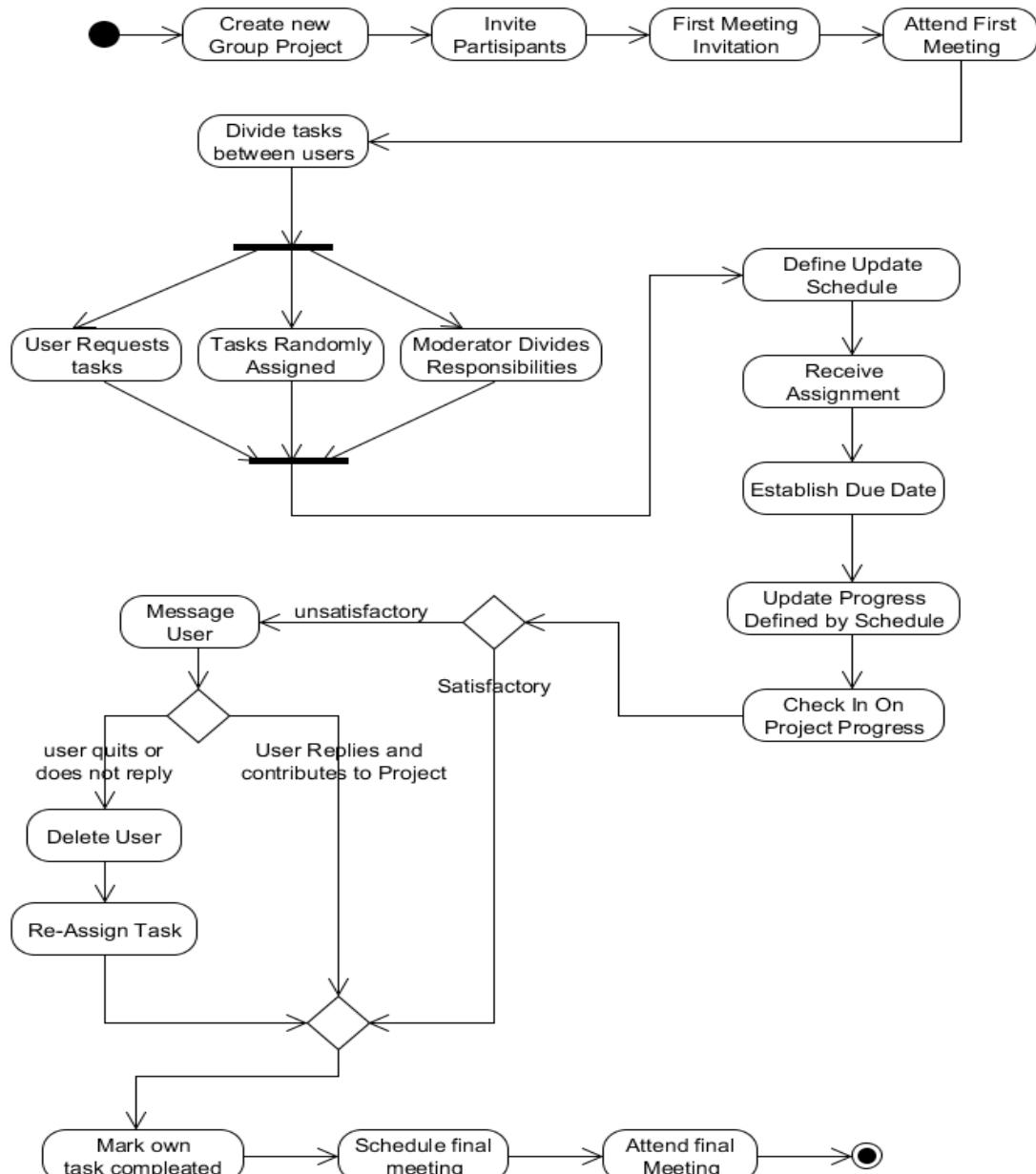
# UML: Unified Modeling Language

---

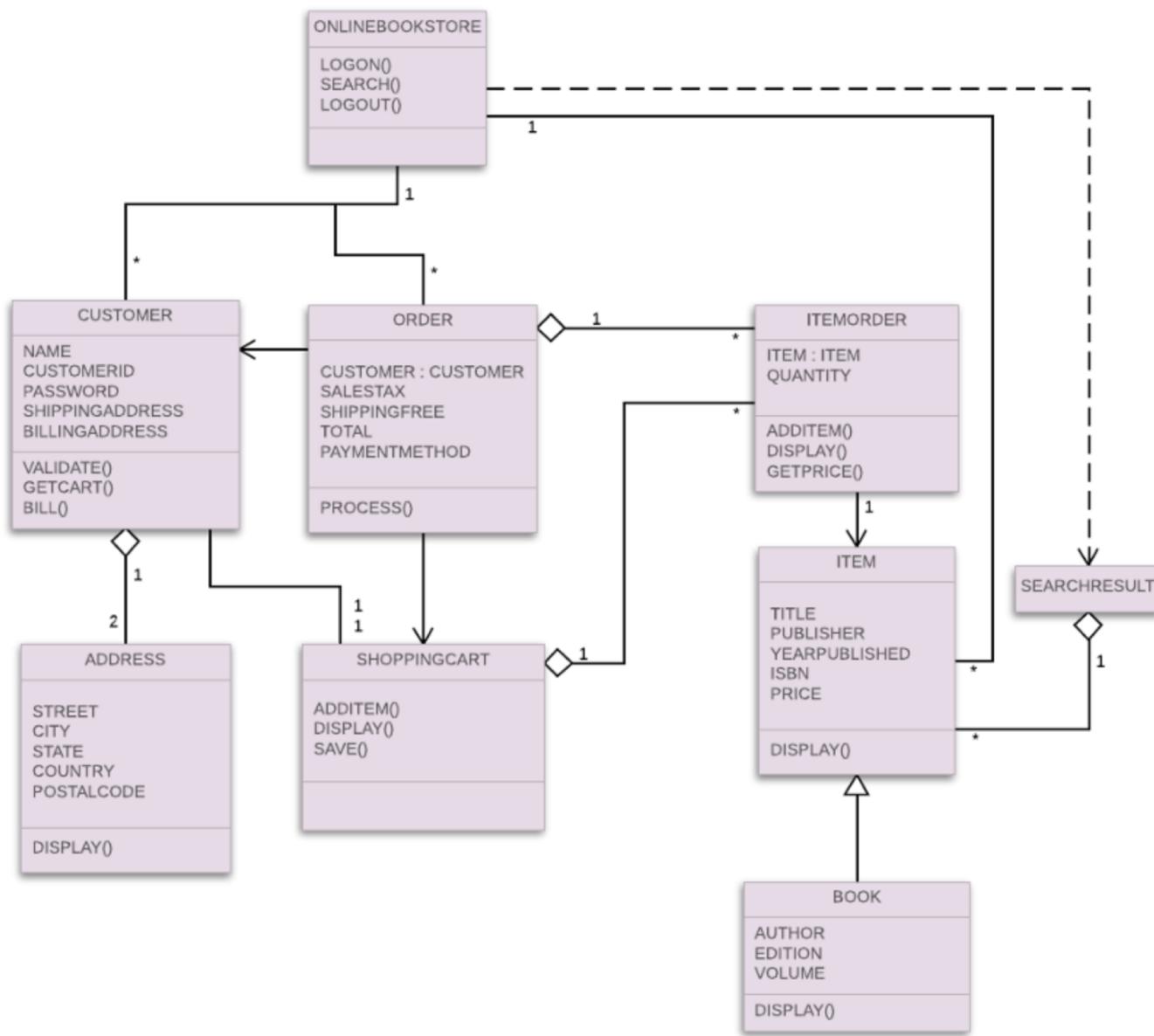
- 1997 yılında OMG (Object Management Group) tarafından geliştirilmiştir.
- UML **resimsel** olarak bir yazılımın taslağını ortaya koymaktadır.
- UML digramlarını program koduna çevirebilmektedir.

[https://www.tutorialspoint.com/uml/uml\\_overview.htm](https://www.tutorialspoint.com/uml/uml_overview.htm)

# UML: Unified Modeling Language (Aktivite Diagramı)

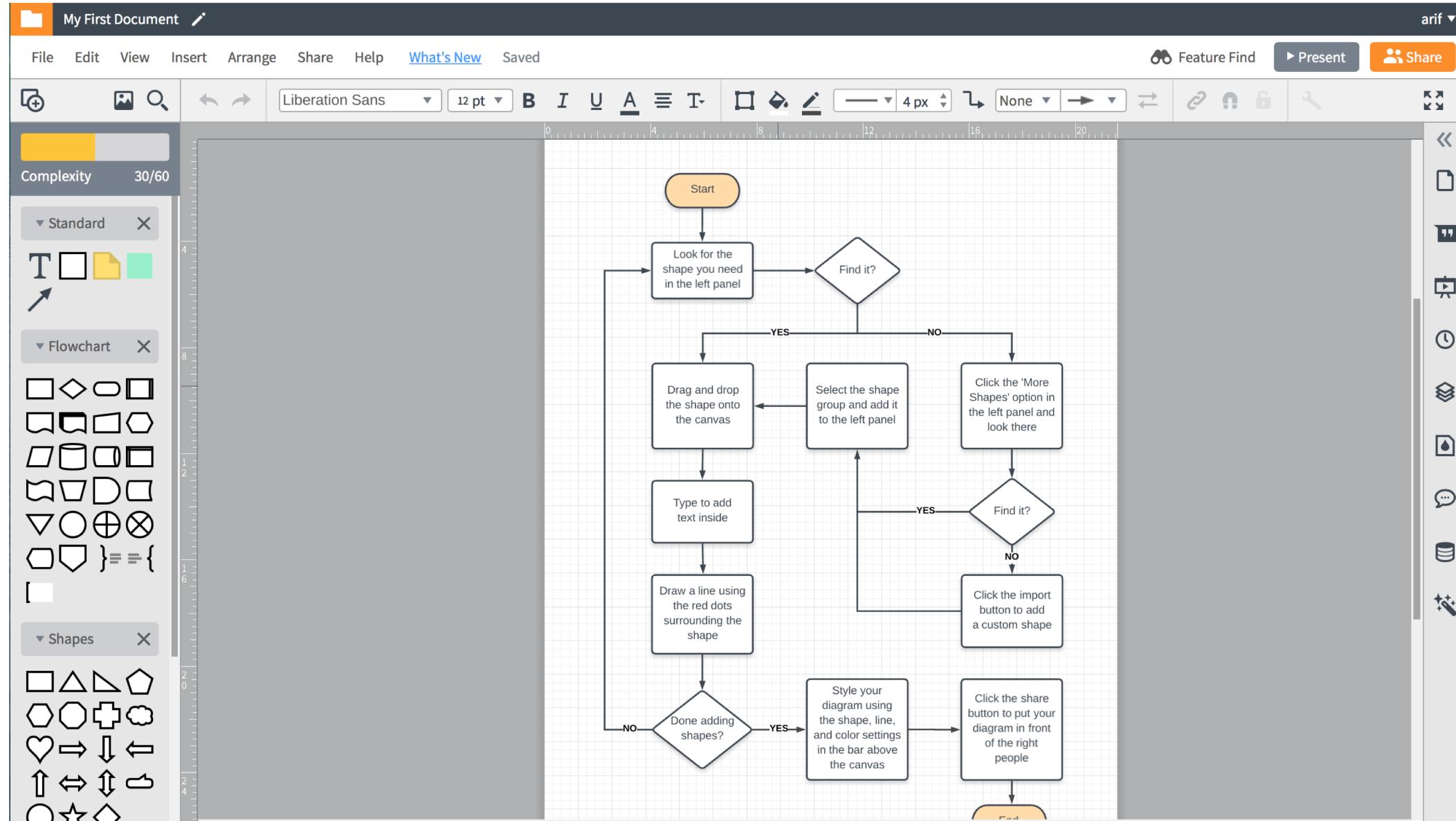


# UML: Unified Modeling Language (Sınıf Diagramı)



[https://d2slcw3kip6qmk.cloudfront.net/marketing/pages/chart/what-is-a-class-diagram-in-UML/UML\\_class\\_diagram\\_example3-800x786.png](https://d2slcw3kip6qmk.cloudfront.net/marketing/pages/chart/what-is-a-class-diagram-in-UML/UML_class_diagram_example3-800x786.png)

# UML: Kaynaklar



<https://www.lucidchart.com/pages/uml-deployment-diagram>

# UML: Kaynaklar

<https://argouml.en.softonic.com/>



## Welcome to ArgoUML

ArgoUML is the leading open source UML modeling tool and includes support for all standard UML 1.4 diagrams. It runs on any Java platform and is available in ten languages. See the [feature list](#) for more details.

ArgoUML 0.26 and 0.26.2 were downloaded over 80,000 times and are in use all over the world.

ArgoUML is distributed under the [Eclipse Public License \(EPL\) 1.0](#).

## UMLet 14.3

Free UML Tool for Fast UML Diagrams



Like



Follow

527 followers



<https://www.umlet.com/>

<http://www.umlet.com/umletino/umletino.html>

The header features the BOUML logo (a stylized 'B' and 'UML' with a yellow dot), two version links ('version' with UK and France flags), and a navigation menu with links: Home, Features, Screenshots, Documentation, Download, Install, Contributions, FAQ, Benchmark, Donations, and Author & links.

## BOUML

### Overview

BOUML is a free **UML 2** tool box including a modeler allowing you to specify and generate code in **C++, Java, Idl, Php, Python** and **MySQL**.

Since the release 7.0 **BOUML** is again a free software.

BOUML runs under **Windows, Linux** and **MacOS X**.

<https://www.bouml.fr/>

# UML: Kullanım Alanları

---

## İş modelleme (Business Modeling):

- Tasarımı yapılacak olan yazılımın çözümü gerçekleştirecek iş modelinin etrafında kavranmasını sağlamaktır.
- İş alanının özel ihtiyaçlarını tanımlamak için kullanılır.

## Sistem Modelleme (System Modeling)

- Geliştirilecek olan yazılımın gereksinimlerinin belirlenmesi sürecinde kullanılır.

# UML: Kullanım Alanları

---

## Kuramsal Veri Modelleme (Conceptual Data Modeling)

- ER modelinin oluşturulması aşamasında kullanılır.

## Fiziksel Veritabanı Tasarımı (Physical Database Modeling)

- Fiziksel olarak veritabanını oluşturmak için kullanılır.

# UML: Kullanım Alanları

---

## Donanım Sistem Modelleme (Hardware System Modeling)

- Uygulamada kullanılan donanımı tanımlamak için kullanılır.

# UML: Kullanım Alanları

---

Nesne tabanlı programlamada UML aktif bir biçimde kullanılmaktadır:

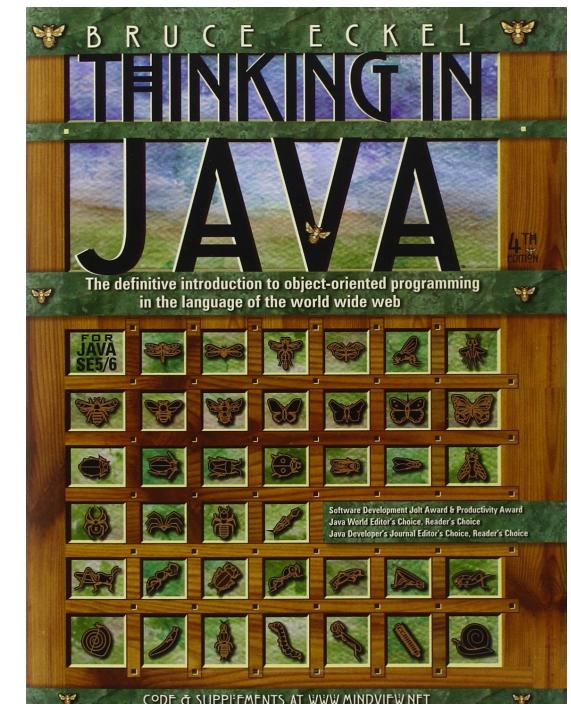
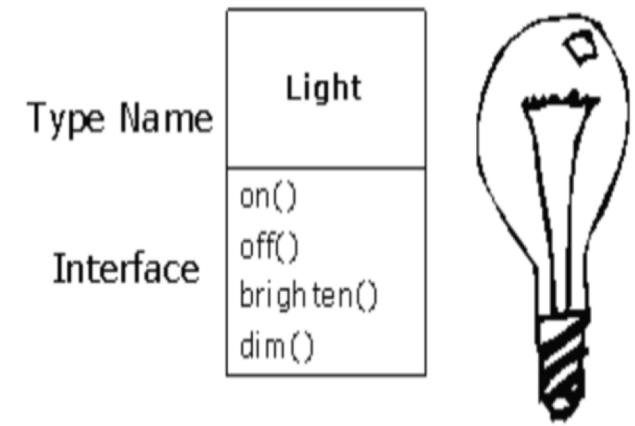
**Yazılımı oluşturan nesneleri**

**ve nesneler arasındaki ilişkileri tanımlamayı sağlar**

# UML: Nesne Tabanlı Programlama Kavramları

## Nesne (Object)

- Durum (state), davranış (behavior) ve kimlik (identity) bilgileri olan varlıklara nesne denir.
- Nesnelerin bir araya gelip birbirlerine gönderilen mesajlar yardımıyla etkilişim halinde oldukları yapıya program denir.
- Her bir nesnenin kendisine ait hafızada (memory) yer ayrılır.
- Her bir nesnenin tipi vardır.
- Aynı tipde olan nesneler aynı mesajları alabilirler



# UML: Nesne Tabanlı Programlama Kavramları

---

## Sınıf (Class)

- Fonksiyonlar ve veriler yardımıyla oluşturulur
- İçerisinde alan (field) , method, yapıcılar (constructors) ve diğer özellikleri barındırır
- programların daha anlaşılır olmasını sağlarlar.

# UML: Nesne Tabanlı Programlama Kavramları

---

## Soyutlama (Abstraction)

Bir sistemin arka planında olan **komplex yapısının gizlenip basit bir ara yüze** kullanıcının kullanımına sunulmasıdır.

- Java: String Class
- Yüksek seviyeli diller (JAVA, C++, ...) assembly dili için bir soyutlamadır.

# UML: Nesne Tabanlı Programlama Kavramları

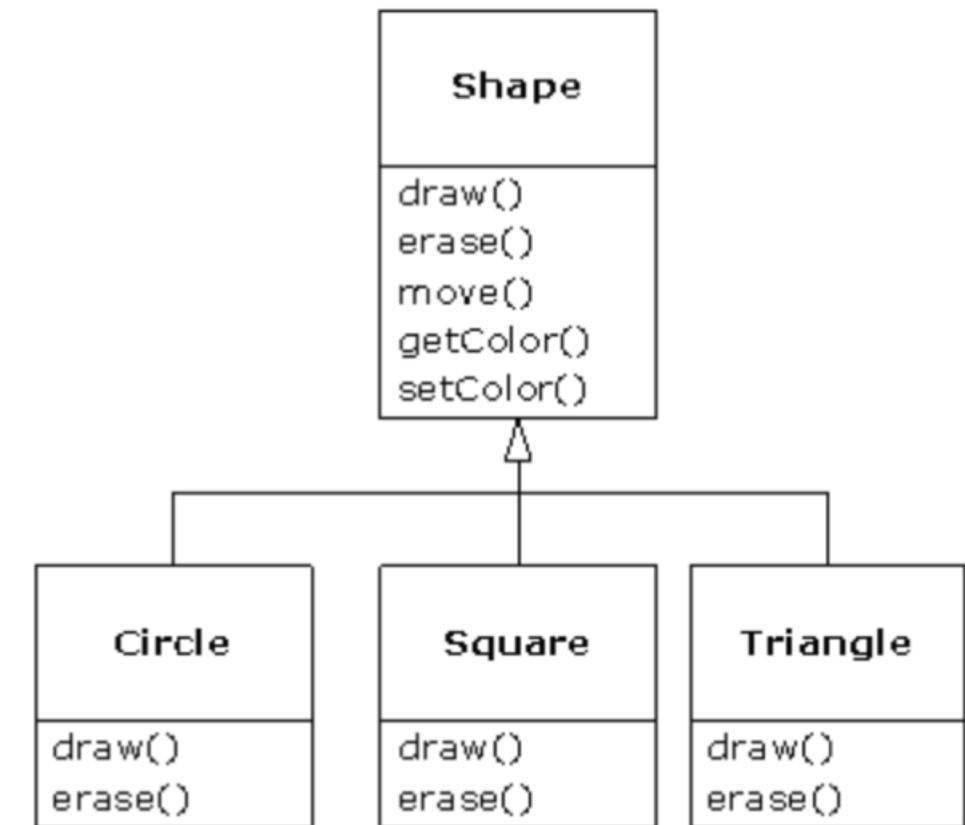
Kapsulleme (Encapsulation, information hiding): Bir birleriyle alakalı kodların bir araya toplanıp ve kodlama detaylarının dışarıdan direkt erişiminin engellenmesiyle gerçekleştirilecektir.

```
class Account{  
     private int account_number;  
    private int account_balance;  
  
    public void show Data(){  
        // code to show data  
    }  
  
    public void deposit(int a){  
        // code to deposit  
    }  
}
```

# UML: Nesne tabanlı programlama kavramları

## Kalıtım (inheritance)

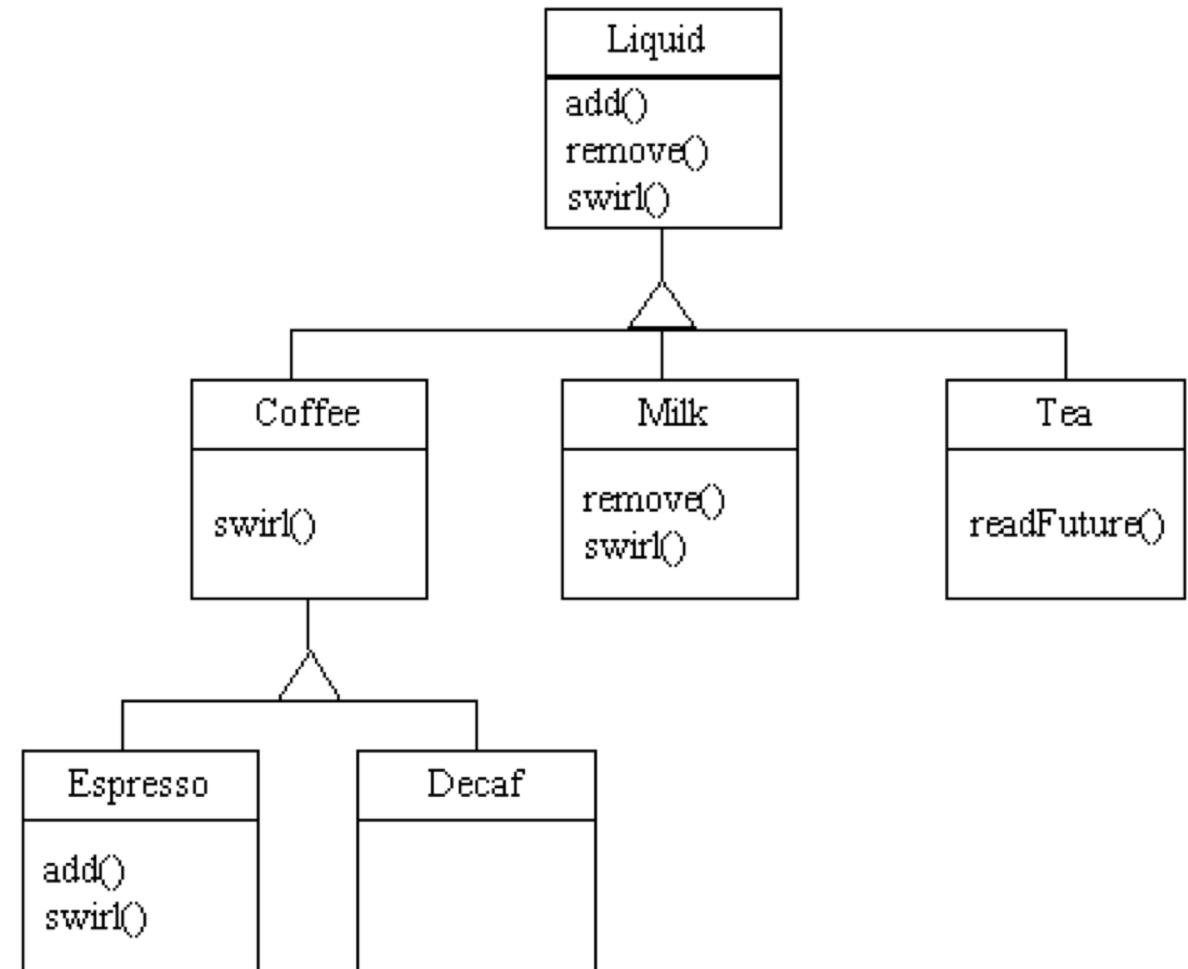
- Üst sınıfın özelliklerini katılsal olarak devralmaktadır
- Kod tekrarını önler ve alt sınıflara yeni özellikler eklenmesine imkan sağlar



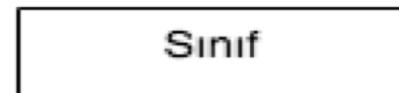
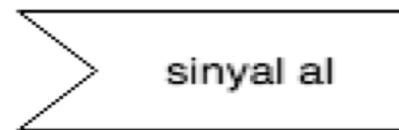
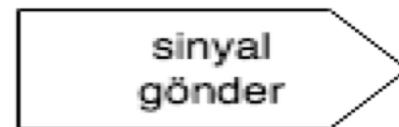
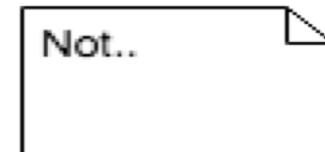
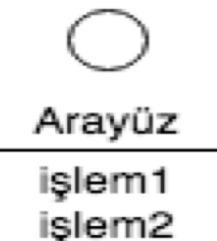
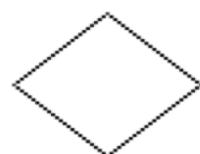
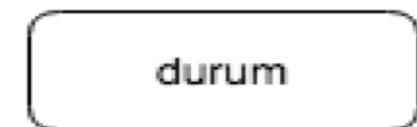
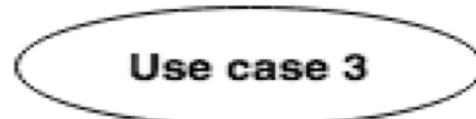
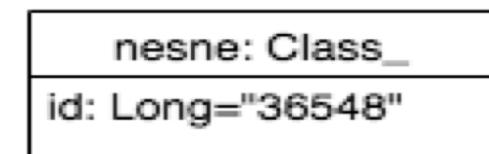
# UML: Nesne tabanlı programlama kavramları

**Polymorphism** ( çokbiçimlilik,  
çokçeşitlilik)

Bir sınıfın alt sınıflarının birden fazla ve çeşitli davranışlar göstermesidir.



# UML Sembollerı



bağıntı(association)

one-to-many

1    \*

many-to-many

\*    \*

yönlü bağlantı

kümelleme(icerir)

kompozisyon(oluşur)

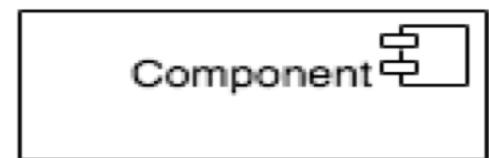
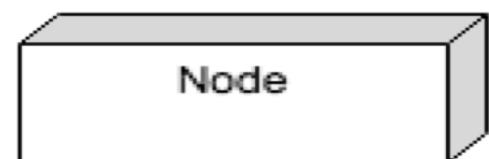
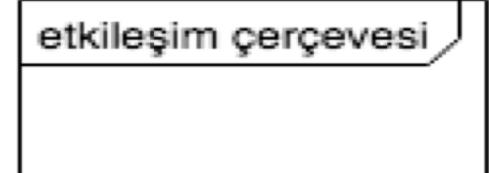
kalıtım(inheritance)

< bağımlılık(dependency) - - -

sağlar

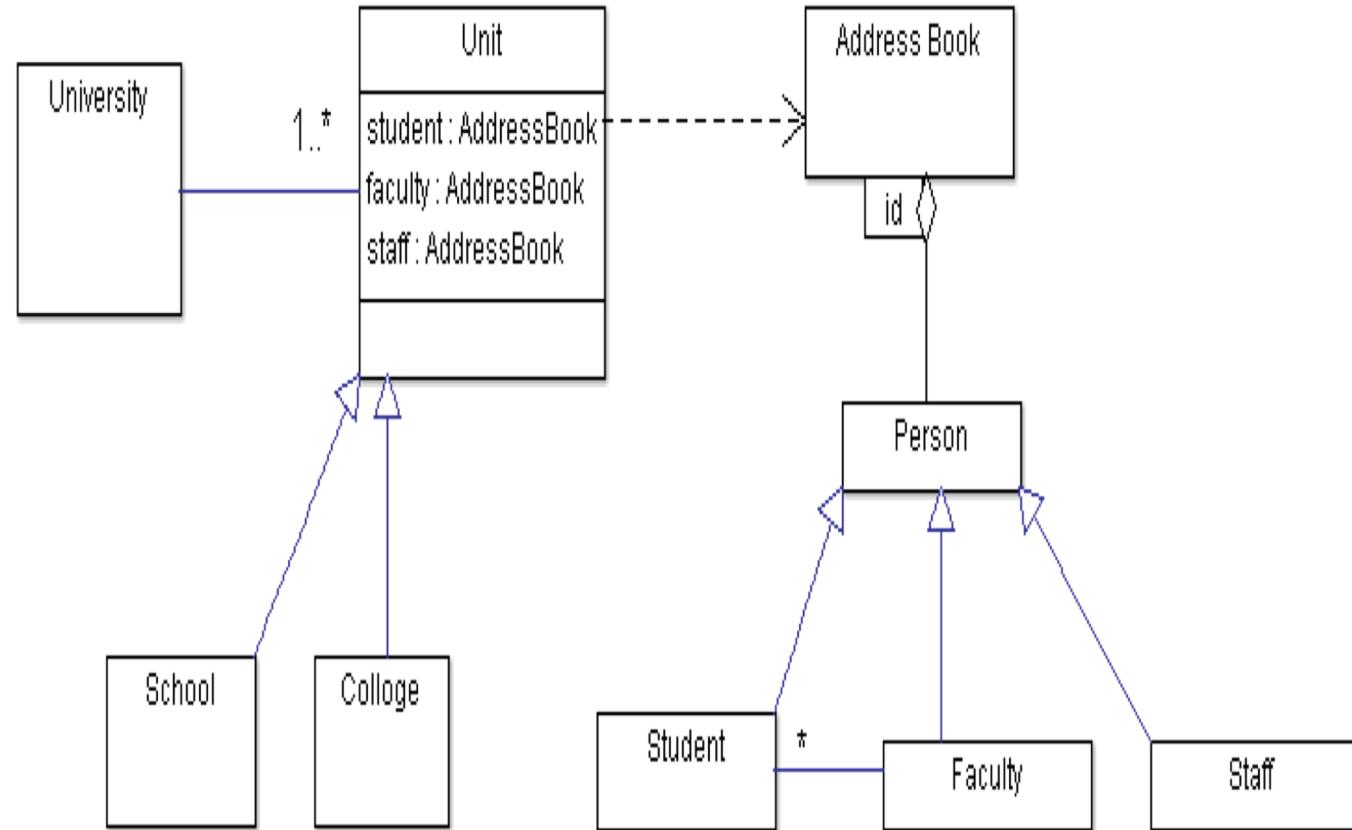
gerekir

:örnek



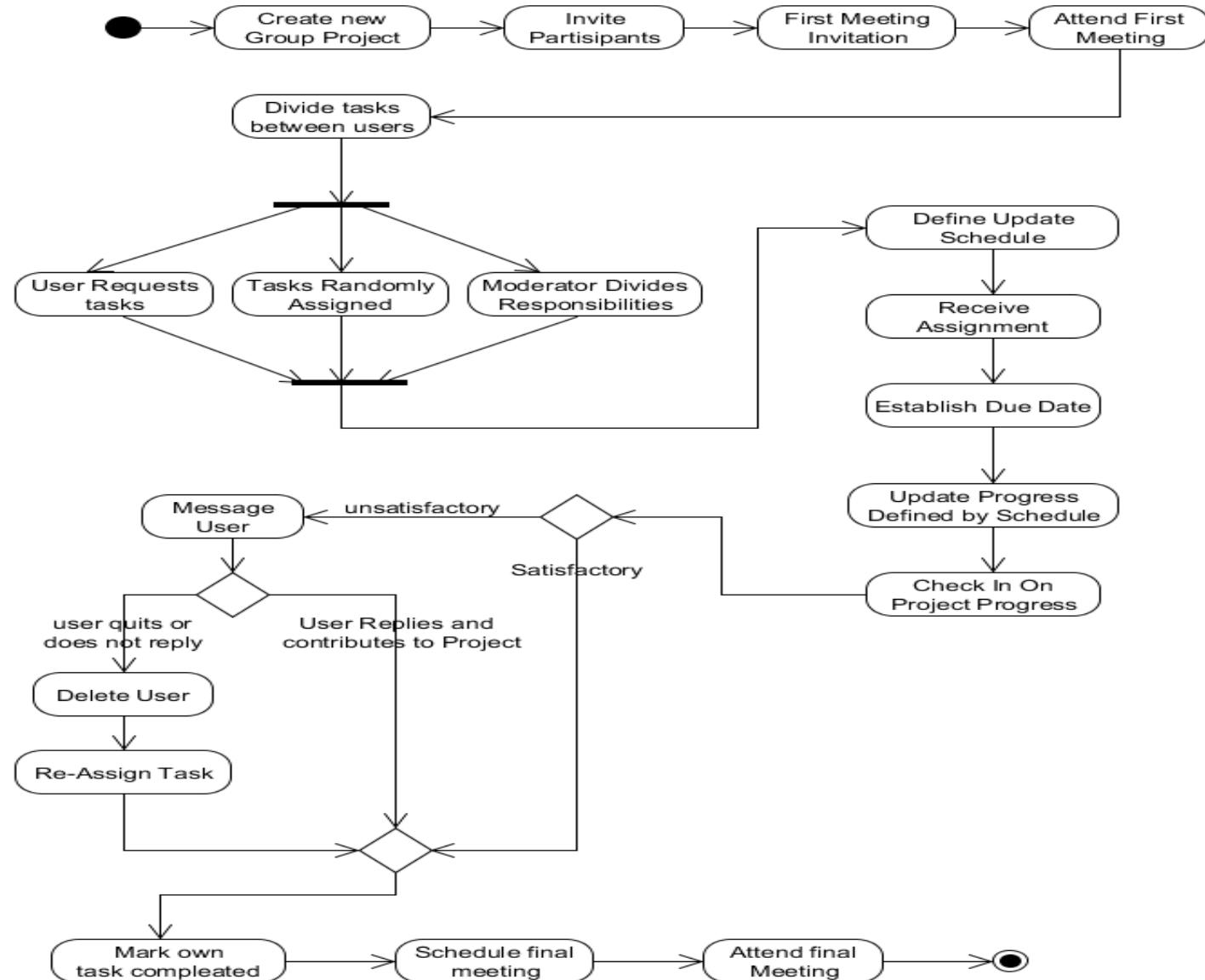
# UML: Yapısal Diagramlar

- Sınıf (class)
- paket (package),
- Nesne (object),
- bileşen (component)
- bileşik yapı (composit structure)
- uygulama (deployment).



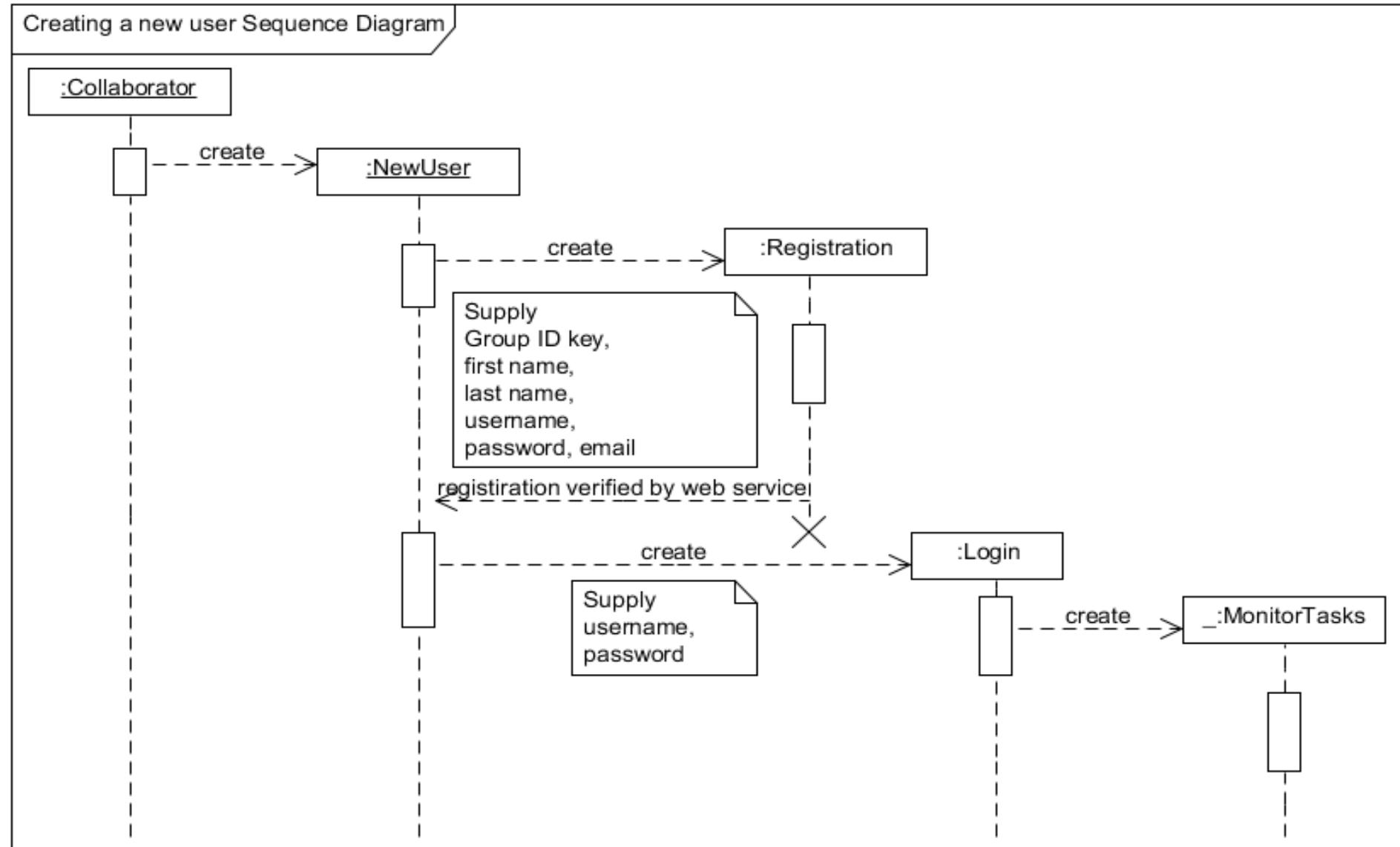
# UML: Davranışsal Diagramlar

## Aktivite (Activity) diagramı

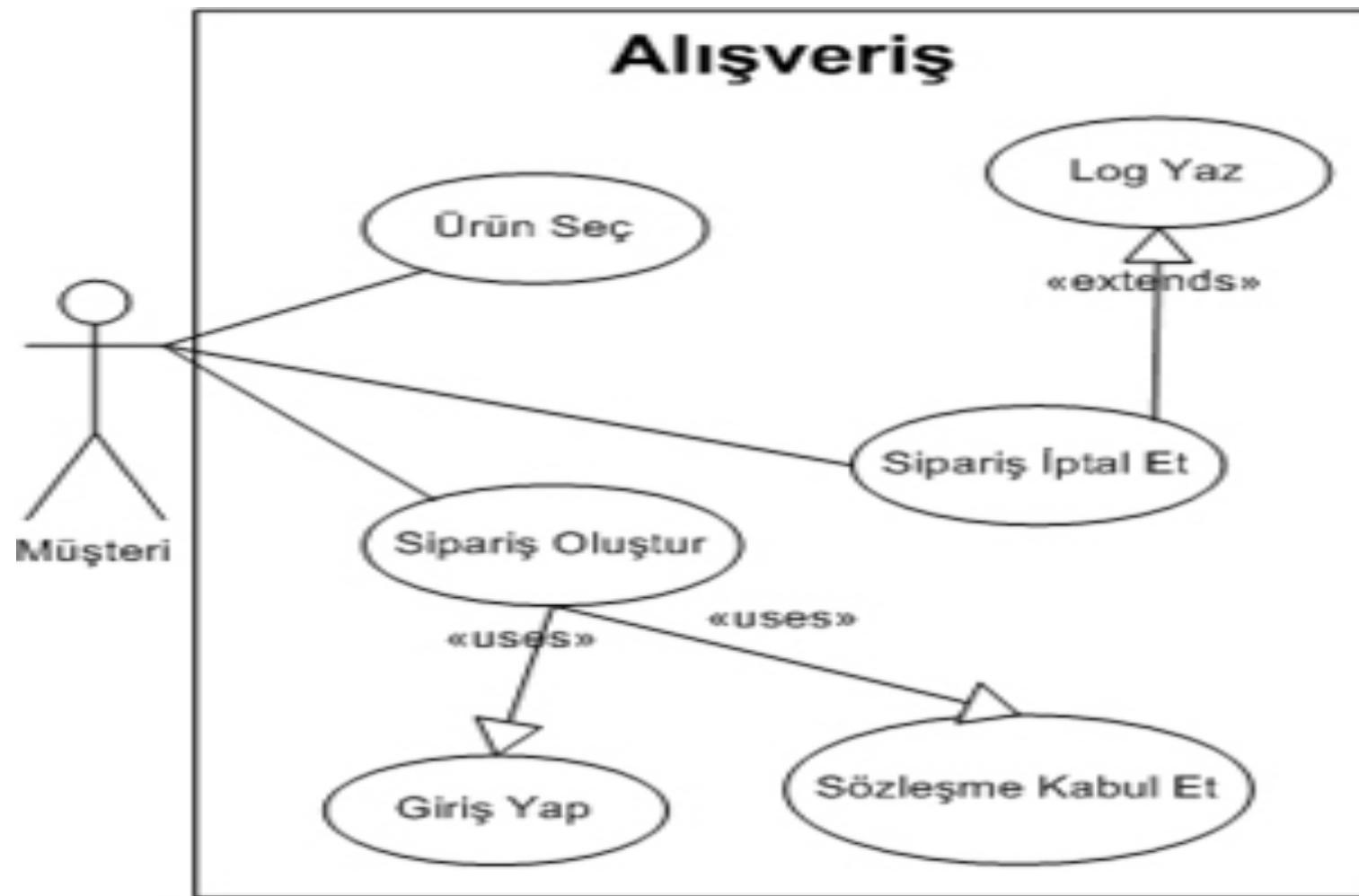


# UML: Davranışsal Diagramlar

Sıralı (sequence)  
diagram



# UML: Davranışsal Diagramlar



**Kullanım senaryosu**  
(Use case) diagram