TÜMLEŞİK MODELLEME DİLİ

UML

(Unified Modeling Language)

UML NEDİR?

* Yazılım ve donanımların bir arada düşünülmesi gereken,
* Zor ve karmaşık programların,
* Özellikle birden fazla yazılımcı tarafından kodlanacağı durumlarda,
* Standart sağlamak amacıyla endüstriyel olarak geliştirilmiş grafiksel bir dildir.
* ~~Programlama dili~~ Diyagram çizme ve ilişkisel modelleme dili

UML NEDİR?

* UML 'in doğuşu son yıllarda yazılım endüstrisindeki en büyük gelişmelerden biri olarak kabul edilebilir.
* UML 1997 yılında yazılımın, diyagram şeklinde ifade edilmesi için bir standartlar komitesi tarafından oluşturuldu.
* Diğer dallardaki mühendislerin standart bir diyagram çizme aracı -Programcıların UML

UML NEDİR?

* UML yazılım sisteminin önemli bileşenlerini tanımlamayı, tasarlamayı ve dokümantasyonunu sağlar
* Yazılım geliştirme sürecindeki tüm katılımcıların (kullanıcı, iş çözümleyici, sistem çözümleyici, tasarımcı, programcı,...) gözüyle modellenmesine olanak sağlar,
* UML gösterimi nesneye dayalı yazılım mühendisliğine dayanır.

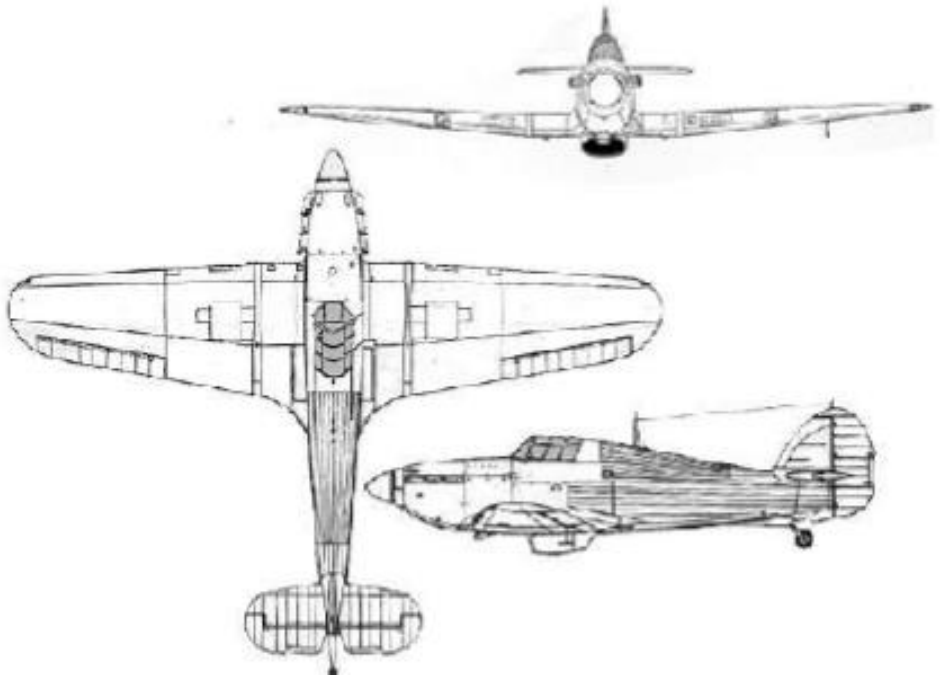
UML Faydası

* Yazılımın geniş bir analizi ve tasarımı yapılmış olacağından kodlama işlemi daha kolay ve doğru olur
* Hataların en aza inmesine yardımcı olur
* Geliştirme ekibi arasındaki iletişimi kolaylaştırır
* Tekrar kullanılabilir kod sayısını artırır
* Tüm tasarım kararları kod yazmadan verilir
* Yazılım geliştirme sürecinin tamamını kapsar
* “resmin tamamını” görmeyi sağlar

Grafiksel Gösterimler

Grafiksel Gösterimler

* Bu grafiksel gösterimler aynı şeye farklı şekillerde bakabilmeyi sağlar.



4+1 bakış

* Bir yazılım sistemi oluşturulurken sadece tek boyutta analiz ve modelleme yapılmaz.
* Bu amaçla; yazılım geliştirme sürecinin farklı aşamalarında farklı UML diyagramlarını kullanmak gerekmektedir.
* 4+1 bakış, bu diyagramları sınıflandırmak ve yazılım yaşam çevrimindeki kullanım yerlerini ortaya koymak için kullanılan bir kavramdır.

4+1 bakış

1. • Kullanıcı Bakışı (User View)
2. • Yapısal Bakış (Structural View)
3. • Davranış Bakışı (Behavioral View)
4. • Gerçekleme Bakışı (Implementation View)
5. • Ortam Bakışı (Environment View)

4+1 bakış

1. Kullanıcı Bakışı (User View)
2. Yapısal Bakış (Structural View)
3. Davranış Bakışı (Behavioral View)
4. Gerçekleme Bakışı (Implementation View)
5. Ortam Bakışı (Environment View)

4+1 bakış

1-) Kullanıcı Bakışı (User View)

* Müşteri gereksinimlerini ortaya koymak ve müşteriye, sistemi tanıtmak amacı ile kullanılan bakış açısıdır.
* Bazı kaynaklarda; ***kullanım senaryosu (use case)*** bakışı olarak da açıklanmaktadır.
* Bu amaçla "kullanım senaryosu" (use case) diyagramları kullanılmaktadır.

4+1 bakış

1. • Kullanıcı Bakışı (User View)
2. • Yapısal Bakış (Structural View)
3. • Davranış Bakışı (Behavioral View)
4. • Gerçekleme Bakışı (Implementation View)
5. • Ortam Bakışı (Environment View)

4+1 bakış

2-) Yapısal Bakış (Structural View)

* Sistemin nelerden meydana geldiğini gösteren bakış açısıdır.
* Bu amaçla; "sınıf" (class) diyagramları ve "nesne" (object) diyagramları kullanılmaktadır.

4+1 bakış

1. • Kullanıcı Bakışı (User View)
2. • Yapısal Bakış (Structural View)
3. • Davranış Bakışı (Behavioral View)
4. • Gerçekleme Bakışı (Implementation View)
5. • Ortam Bakışı (Environment View)

4+1 bakış

3-) Davranış Bakışı (Behavioral View)

* Sistemin dinamik yapısını ortaya koyan bakış açısıdır.
* Bu amaçla; ***«ardışık" (sequence), "işbirliği" (collaboration), "durum« (state) ve «etkinlik" (activity)*** diyagramları kullanılmaktadır.

4+1 bakış

1. • Kullanıcı Bakışı (User View)
2. • Yapısal Bakış (Structural View)
3. • Davranış Bakışı (Behavioral View)
4. • Gerçekleme Bakışı (Implementation View)
5. • Ortam Bakışı (Environment View)

4+1 bakış

4-) Gerçekleme Bakışı (Implementation View)

* Sistemin alt modüllerini ortaya koyan bakış açısıdır.
* Bu amaçla; "bileşen" (component) diyagramları kullanılmaktadır.

4+1 bakış

1. • Kullanıcı Bakışı (User View)
2. • Yapısal Bakış (Structural View)
3. • Davranış Bakışı (Behavioral View)
4. • Gerçekleme Bakışı (Implementation View)
5. • Ortam Bakışı (Environment View)

4+1 bakış

5-) Ortam Bakışı (Environment View)

* Donanımın, fiziksel mimarisinin ortaya konduğu bakış açısıdır.
* Bu amaçla; "dağıtım" (deployment) diyagramları kullanılmaktadır.

Sınıf Diyagramları

* Sınıf Diyagramları UML 'in en sık kullanılan diyagram türüdür.
* Sınıflar nesne tabanlı programlama mantığından yola çıkarak tasarlanmıştır.
* Sınıf diyagramları bir sistem içerisindeki nesne tiplerini ve birbirleri ile olan ilişkileri tanımlamak için kullanılırlar.

Sınıf Diyagramları

* Sınıfların
  * bir adı
  * nitelikleri ve
  * İşlevleri vardır
  * Bunlara ek olarak
    * “notes”
    * “Constraints”

Sınıf Diyagramları

* Örnek:

NESNE Örneği

* Kimlik:
  Öğrenci123
* Durum:

ad: “Ali Yılmaz”

öğrenciNo: “0401…..”

yıl: 2007

* Metotlar:

dersEkle()

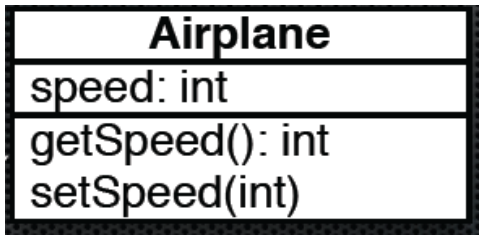
dersSil()

danışmanAta()

Sınıf (*class*) aşağıdakileri tanımlar:

* Alanlar (*fields*):Nesne özelliklerini tanımlayan değişkenler, adları ve türleri ile.
* Metotlar (*methods*):Metot adları,döndürdüğü tür, parametreleri, ve metotu gerçekleştiren program kodu

UML Diyagramı




```java
public class Airplane {

    private int speed;

    public Airplane(int speed) {
        this.speed = speed;
    }

    public int getSpeed() {
        return speed;
    }

    public void setSpeed(int speed) {
        this.speed = speed;
    }

}
```

UML Sınıf Tanımlamaları

* **Alanlar:**

Kod→private long maas

UML→private maas:long

* **Metotlar:**

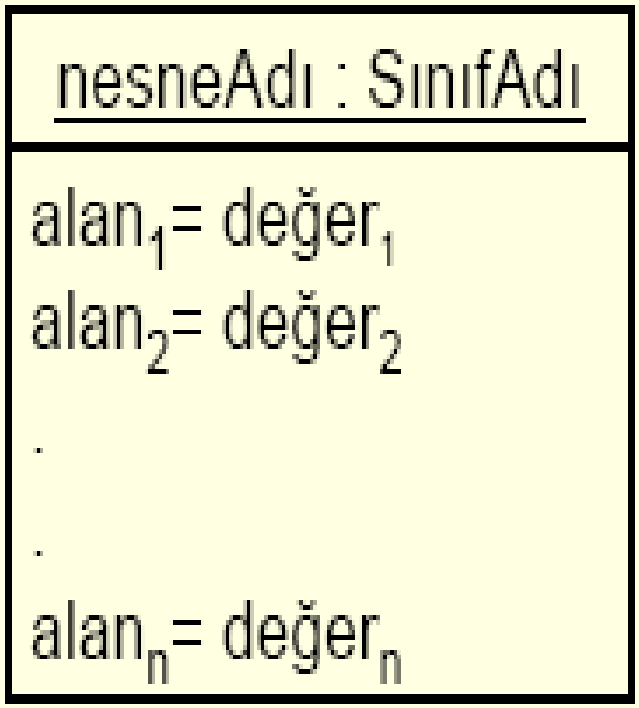
Kod→public doublemaas Hesapla()

UML→public maasHesapla():double

ERİŞİM

* **Public:**diğer sınıflar erişebilir. UML'de +sembolü ile gösterilir.
* **Protected:**aynı paketteki (*package*) diğer sınıflar ve bütün alt sınıflar (*subclasses*) tarafında erişilebilir. UML'de#sembolü ile gösterilir.
* **Package:**aynı paketteki (*package*) diğer sınıflar tarafında erişilebilir. UML'de ~sembolü ile gösterilir.
* **Private:**yalnızca içinde bulunduğu sınıf tarafından erişilebilir (diğer sınıflar erişemezler). UML'de - sembolü ile gösterilir.

UML'de Nesne Gösterimi

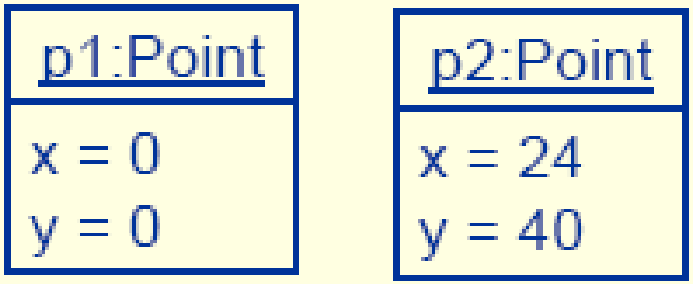


← Nesne ve Sınıf Adı; altı çizili

← Alanlar ve aldıkları değerler

UML'de Nesne Gösterimi

UML



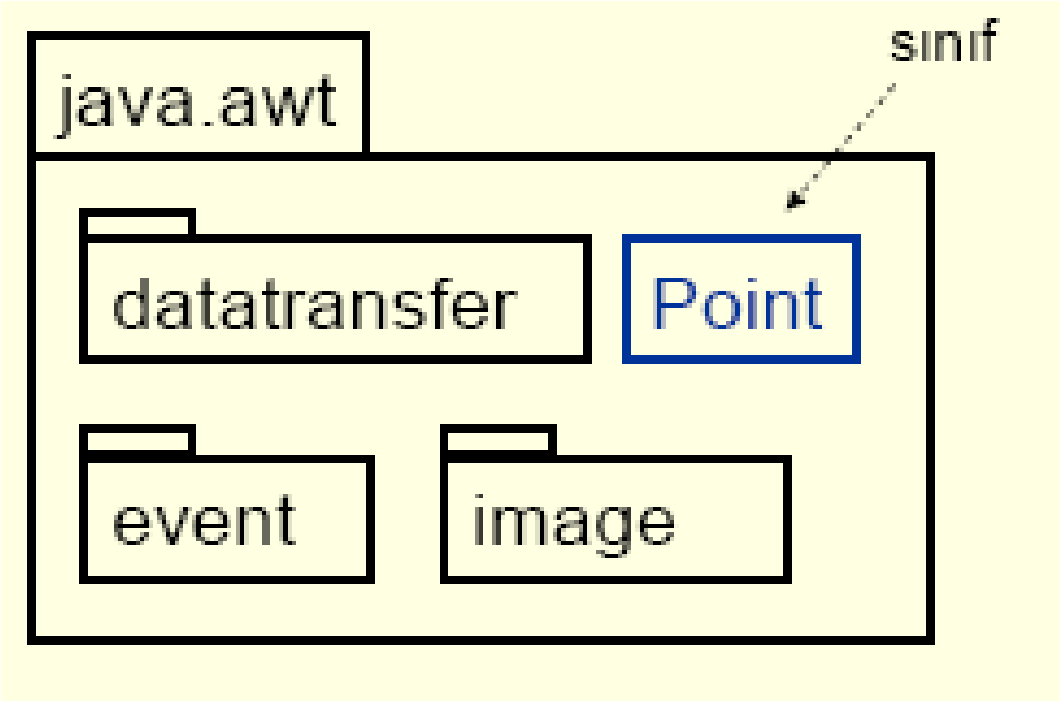
Java

```
Point p1 = new Point();
p1.x = 0;
p1.y = 0;

Point p2 = new Point();
p2.x = 24;
p2.y = 40;
```

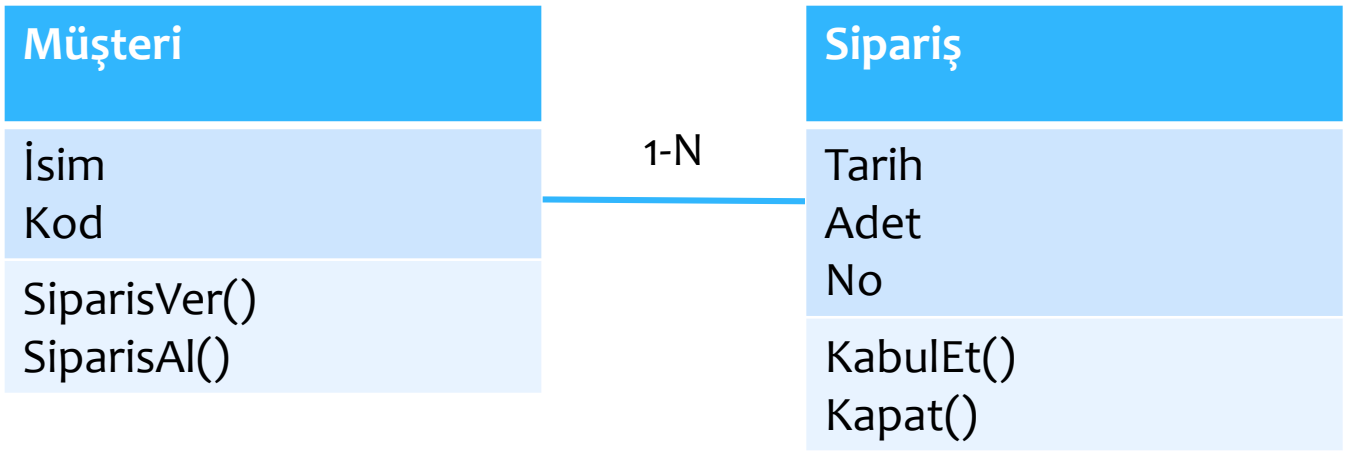
UML'de Paket Gösterimi

* Birbirleriyle ilişkili sınıflar bir paket (package) içine yerleştirilirler.
* Paket isimler küçük harflerle yazılır



Sınıf Diyagramları

* Sınıflar arasındaki ilişkiyi göstermek için iki sınıf arasına düz bir çizgi çekilir.
* İlişkiyi gösteren çizginin üzerine ilişkinin türü yazılır.

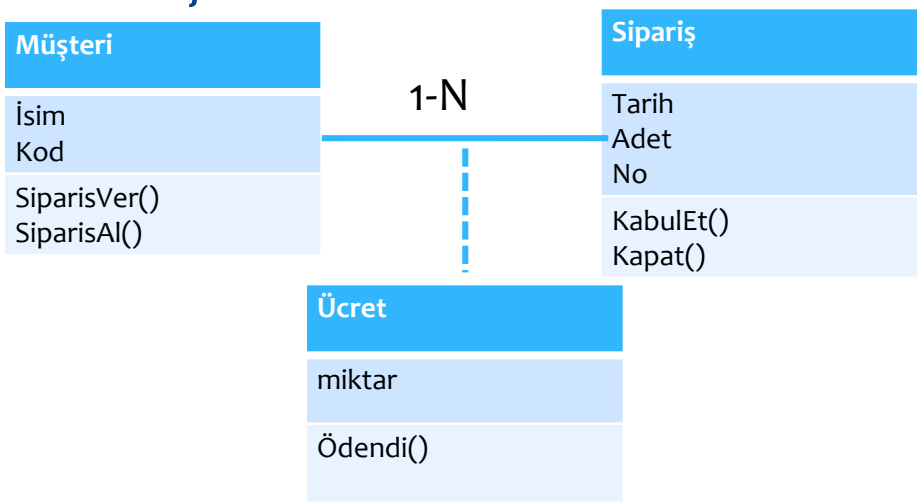


Sınıf Diyagramları

* Bazı durumlarda sınıflar arasındaki ilişki, bir çizgiyle belirtebilecekşekilde basit olmayabilir.
* Bu durumda ilişki sınıfları kullanılır.
* İlişki sınıfları bildigimiz sınıflarla aynıdır.
* Sınıflar arasındaki ilişki eğer bir sınıf türüyle belirleniyorsa UML ile gösterilmesi gerekir.

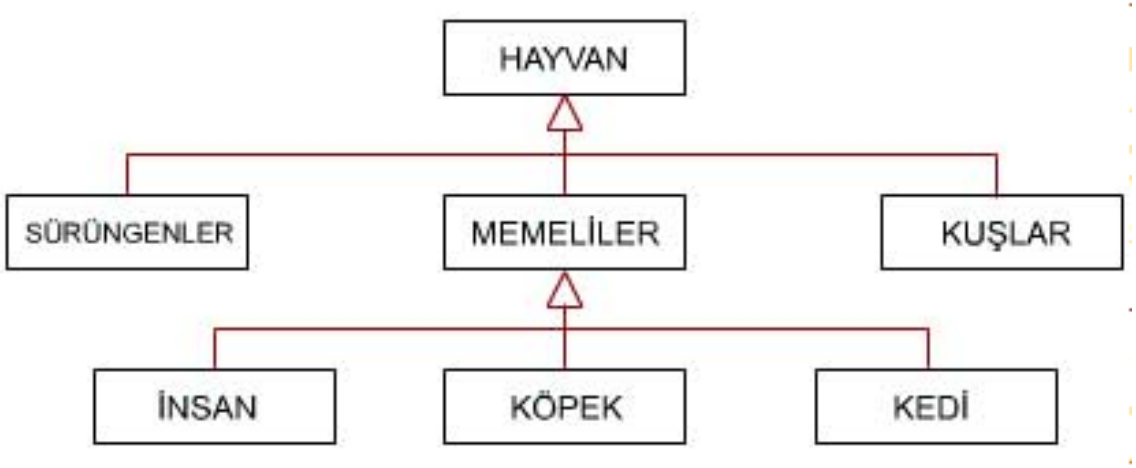
Sınıf Diyagramları

* Müşteri ile Sipariş sınıfı arasında ilişki vardır.Fakat müşteri satın alırken Ücret ödemek zorundadır
* Bu ilişkiyi göstermek için Ücret sınıfı ilişki ile kesikli çizgi ile birleştirilir.

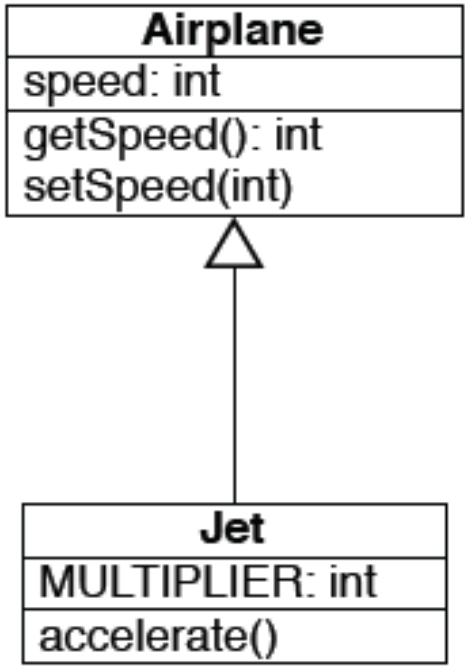


KALITIM (INHERITANCE)

* Eğer eşyalar arasında genellemeler yapabiliyorsak genellemeyi yaptığımız eşyalarda ortak özelliklerin olduğunu biliriz.
* Mesela, "Hayvan" diye bir sınıfımız olsun.
* Memeliler, Sürüngenler, Kuşlar da diğer sınıflarımız olsun.
* Memeliler, Sürüngenler ve Kuşlar sınıfının farklı özellikleri olduğu gibi hepsinin Hayvan olmasından dolayı birtakım ortak özellikleri vardır.



KALITIM ÖRNEK



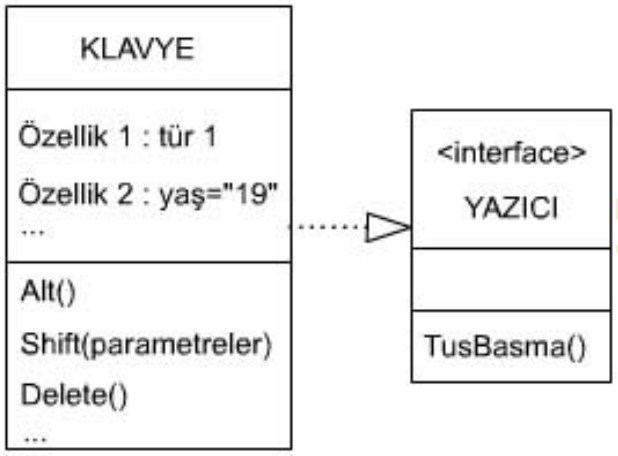

```
1  public class Jet extends Airplane {
2
3      private static final int MULTIPLIER = 2;
4
5      public Jet(int id, int speed) {
6          super(id, speed);
7      }
8
9      public void setSpeed(int speed) {
10         super.setSpeed(speed * MULTIPLIER);
11     }
12
13     public void accelerate() {
14         super.setSpeed(getSpeed() * 2);
15     }
16
17 }
18
```

İÇERME (AGGREGATIONS)

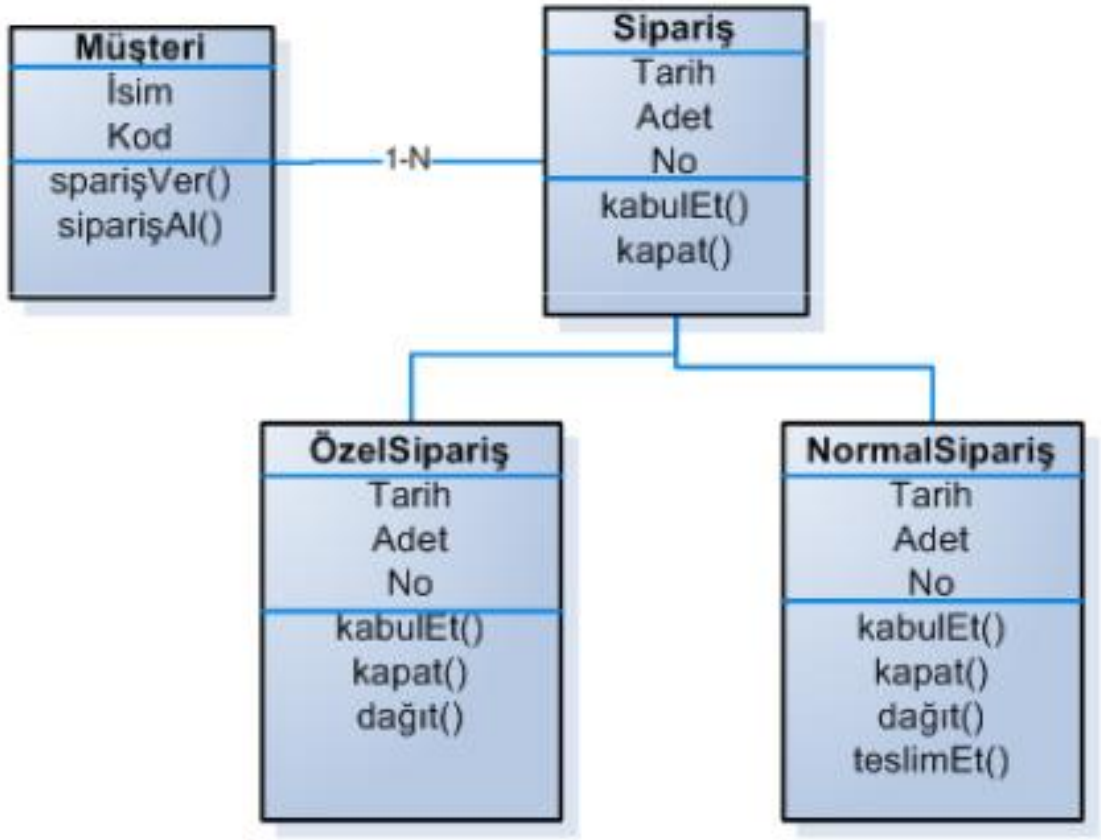
* Bazı sınıflar birden fazla parçadan oluşur.Bu tür özel ilişkiye "Aggregation" denir.Mesela ,bir TV 'yi ele alalım.Bir televizyon çeşitli parçalardan oluşmuştur.Ekran,Uzaktan Kumanda,Devreler vs.. Bütün bu parçaları birer sınıf ile temsil edersek TV bir bütün olarak oluşturulduğunda parçalarını istediğimiz gibi ekleyebiliriz. Aggregation ilişkisini 'bütün parça' yukarıda olacak şekilde ve 'bütün parça'nın ucuna içi boş elmas yerleştirecek şekilde gösteririz.

ARAYÜZ(INTERFACE)

* Bazı durumlarda bir sınıf sadece belirli işlemleri yapmak için kullanılır. Herhangi bir sınıfla ilişkisi olmayan ve standart bazı işlemleri yerine getiren sınıfa benzer yapılara arayüz(interface) denir.
* Arayüzlerin özellikleri yoktur. Yalnızca bir takım işleri yerine getirmek için başka sınıflar tarafından kullanılırlar.
* Mesela, bir "TuşaBasma" arayüzü yaparak ister onu "KUMANDA" sınıfında istersek de aşağıdaki şekilde görüldüğü gibi "KLAVYE" sınıfında kullanabiliriz.



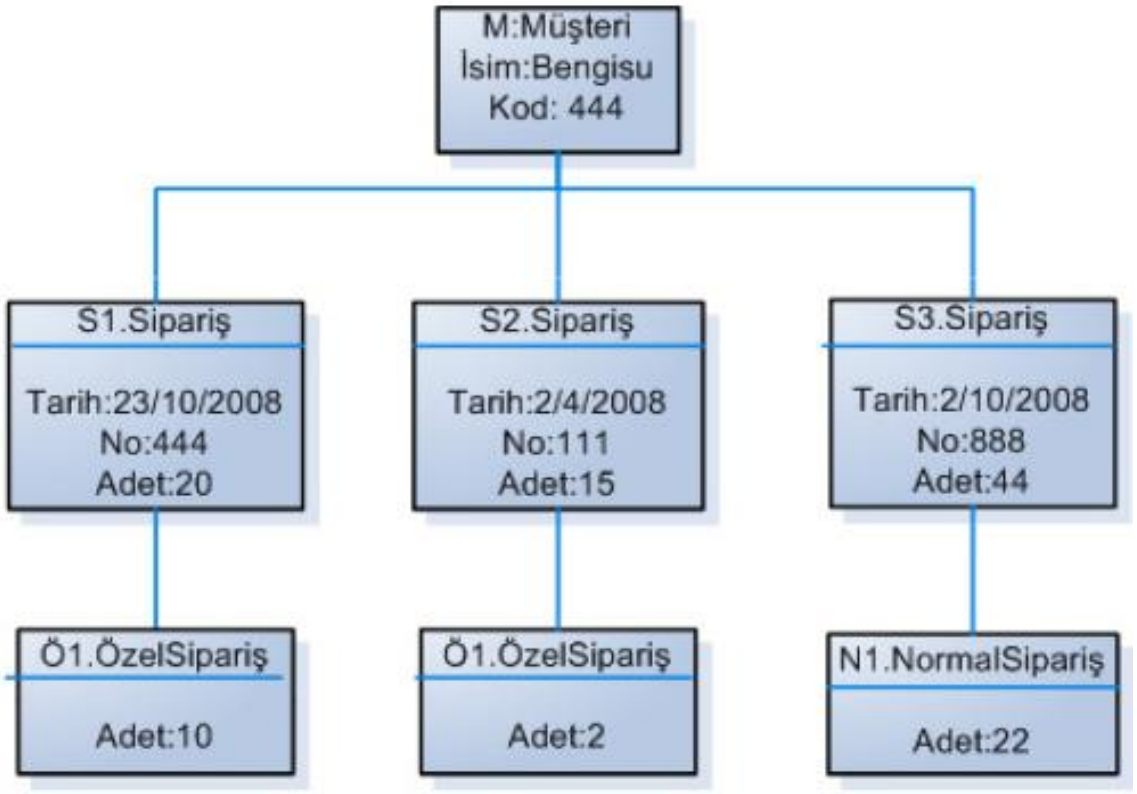
Örnek Sınıf Diyagramı



Nesne Diyagramları

* Nesne Diyagramları nesneler ve aralarındaki bağıntıları gösterirler.
* Nesneler, sınıfların somut örnekleri olduğundan sınıf özelliklerinin değer almış halidirler.

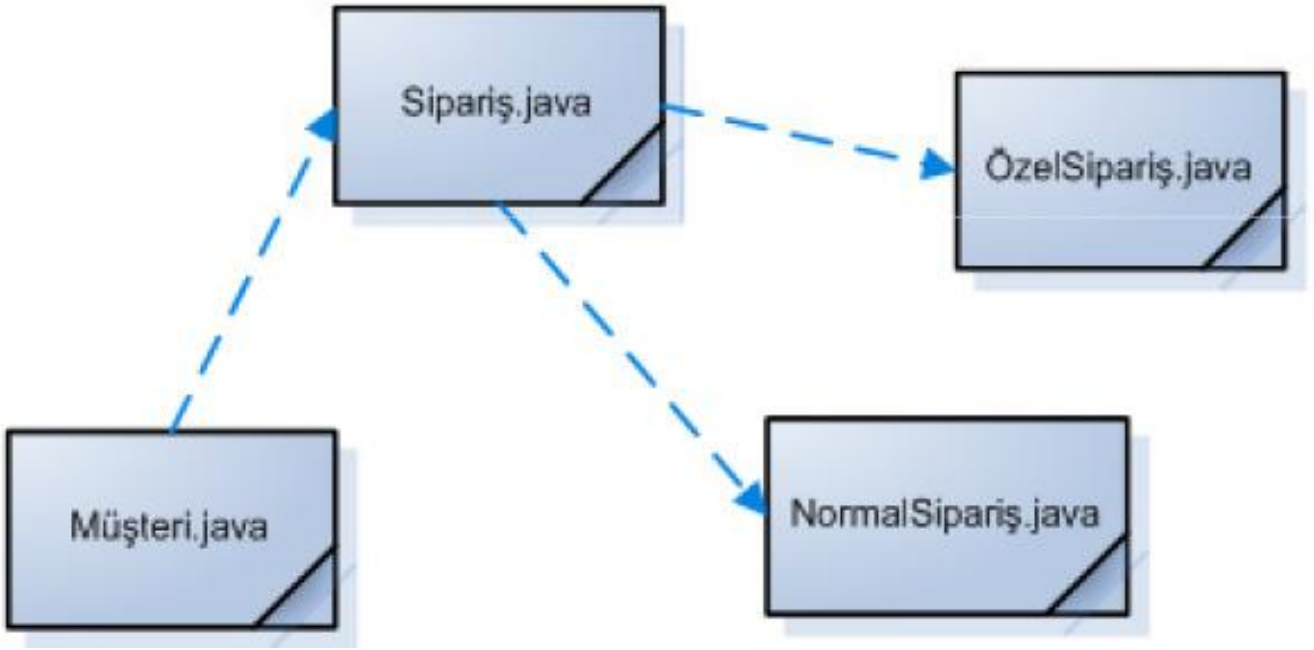
Örnek Nesne Diyagramı



Bileşen Diyagramları

* Bileşen Diyagramları yazılım sistemine daha yüksek bir seviyeden bileşenler seviyesinden bakabilmeyi sağlar.
* Bunlar sistemdeki sınıflar, çalıştırılabilen program parçaları kütüphane bileşenleri veya veritabanı tabloları olabilir.
* Bu gösterim bileşenler, arayüzler ve bağımlılık ilişkilerinden oluşur ve sistemin fiziksel gösterimini sağlar.

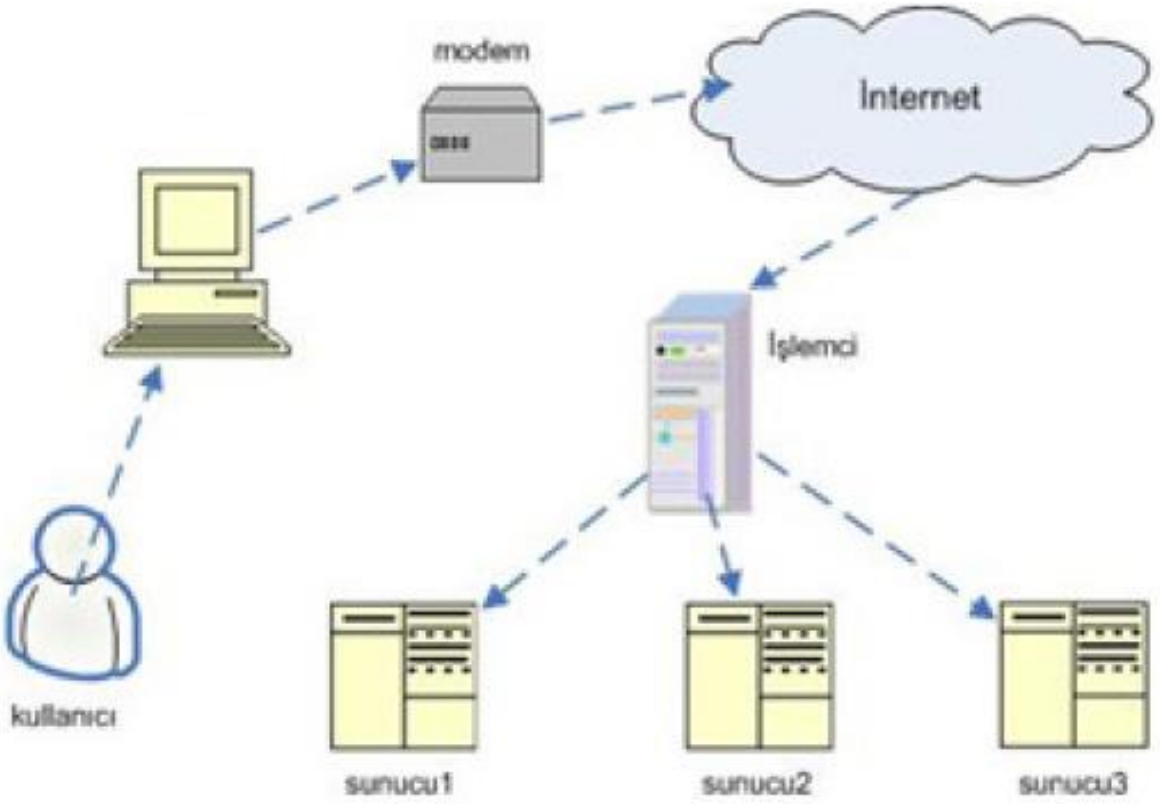
Örnek Bileşen Diyagramı



Dağılım Diyagramları

* Sistemin çalışma platformundaki durumlarını gösterirler.
* Sistemdeki yürütülebilen parçalar, kütüphaneler, tablolar ve dosyalar gibi bileşenlerin dağılımını belirler.
* Bu diyagramlar, sistem donanım mimarisinin gösterilmesi, gerekli donanım bileşenlerinin tanımlanması amacıyla kullanılabilirler.

Örnek Dağılım Diyagramı



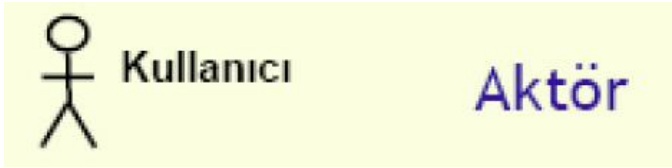
Kullanım Senaryosu Diyagramları

* Kullanım senaryosu diyagramları sistemin işlevsel gereksinimlerinin ortaya çıkarılması için kullanılır.
* Sistemin kullanıcısının bakış açısıyla modellenmesi amacıyla kullanılır.
* Kullanım senaryosu diyagramları sistemin kabaca ne yaptığı ile ilgilenir, kesinlikle nasıl ve neden yapıldığını incelemez.

Kullanım Senaryosu Diyagramları

* Kullanım senaryosu
  * aktörler (sistem dışı aktörler –kullanıcılar veya diğer sistemler)
  * senaryolar (kullanıcı tarafından başlatılan çeşitli olaylar dizisi)

Kullanım Senaryosu Diyagramları

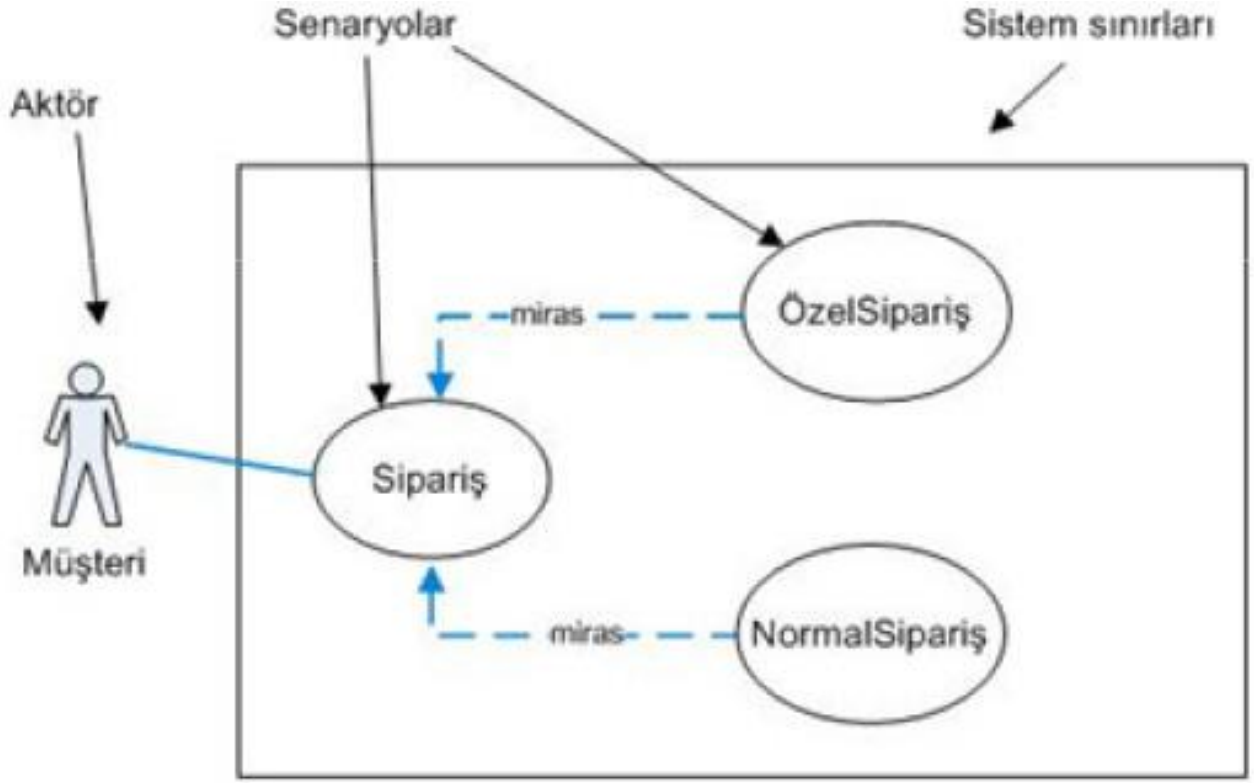


* Aktör genellikle “insan” olarak düşünülmekle birlikte başka “sistem” ve “donanım” da olabilir.
* Aktör sistemi “uyarır” ,işlevleri haricen “tetikler”(aktif) yada sistemden “uyarıcı alır”(pasif).
* Aktör sistemin parçası değildir, “harici” dir.

Kullanım Senaryosu Diyagramları

* Aktörler belirlenir
* Her aktörün “ne” yapmak istediği belirlenir
* Her aktörün “ne” si için “ana senaryo özeti” çıkarılır
* Tüm sistemin ana senaryo özetleri incelenir, ayıklanır, birleştirilir
* Her senaryo için ana işlem adımları belirtilir.

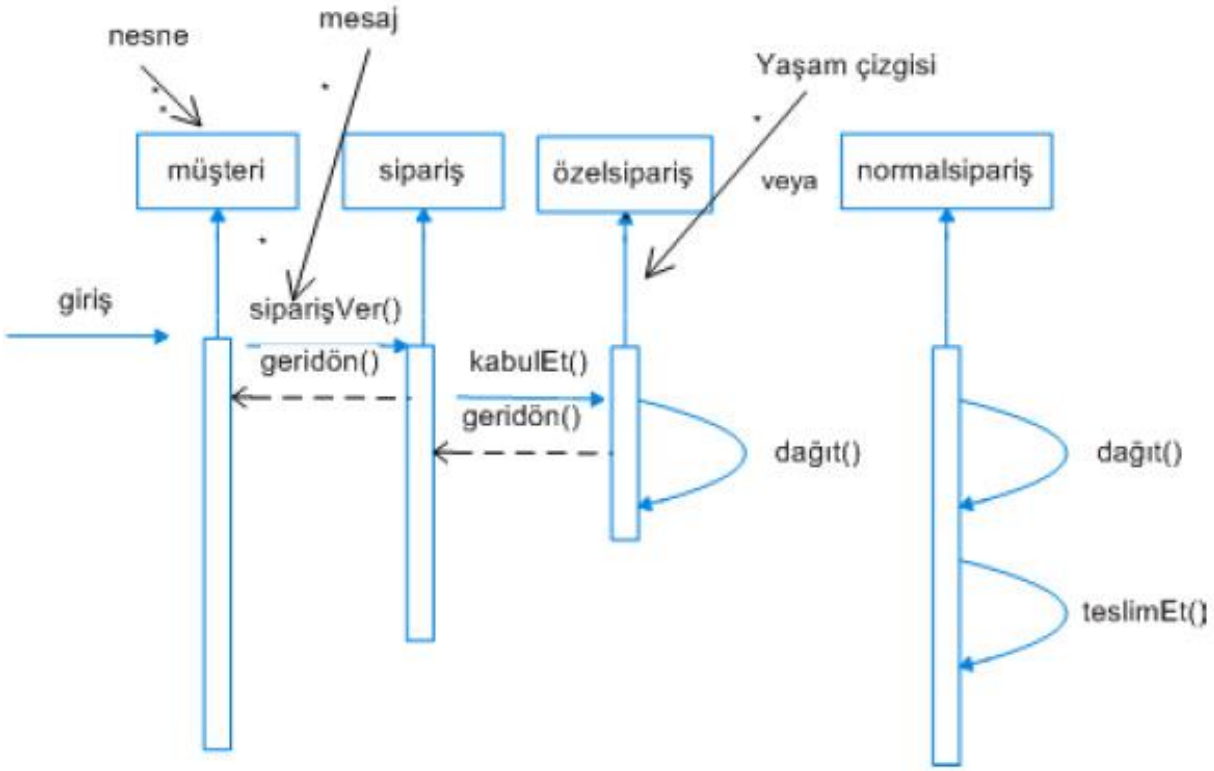
Örnek Kullanım Senaryosu Diyagramı



Ardışık Diyagramlar

* Sistem içindeki nesnelerin zaman içindeki ardışık aksiyonlarını gösterirler
* Sistemin dinamik bir resmini çizer
* Aktörün hayat süresi boyunca gerçekleştirdiği işlemler gösterilir.
  * Aksiyonlar->dikdörtgen
  * Etkileşimler->mesajlar

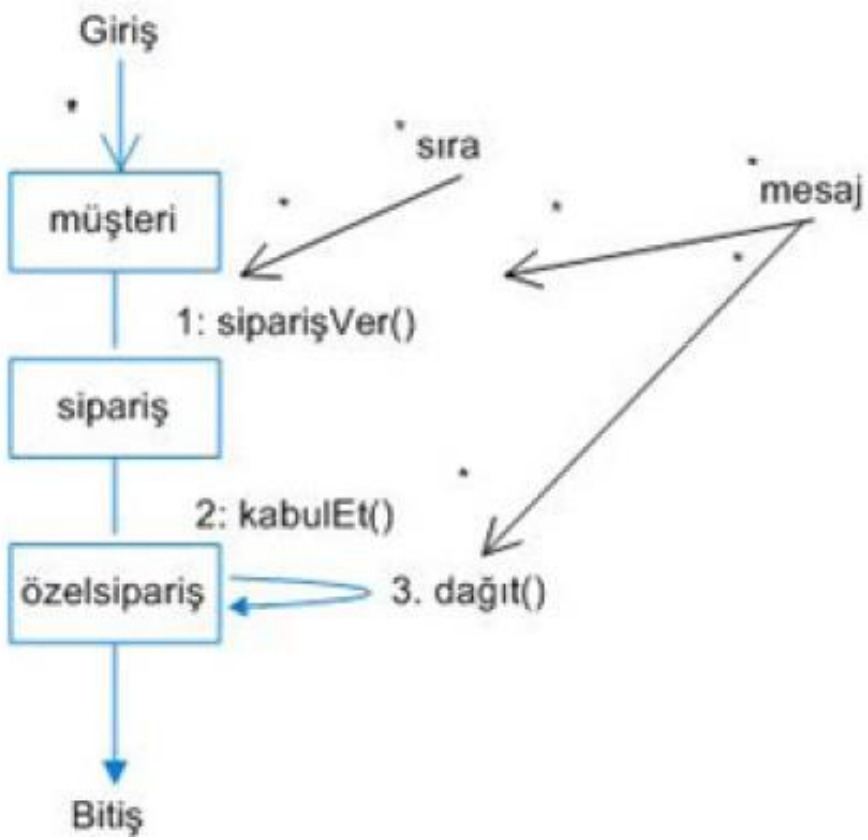
Ardışık Diyagramlar



İşbirliği Diyagramları

* Parçaların bütünü nasıl oluşturduğunu ve sistemin dinamik davranışını göstermek amacıyla kullanılır.
* Nesneler arasındaki bağıntıları sıralı mesajlar şeklinde gösterirler
* Ardışık diyagramlardan zaman kavramının olmayışı ile farklılık gösterirler.

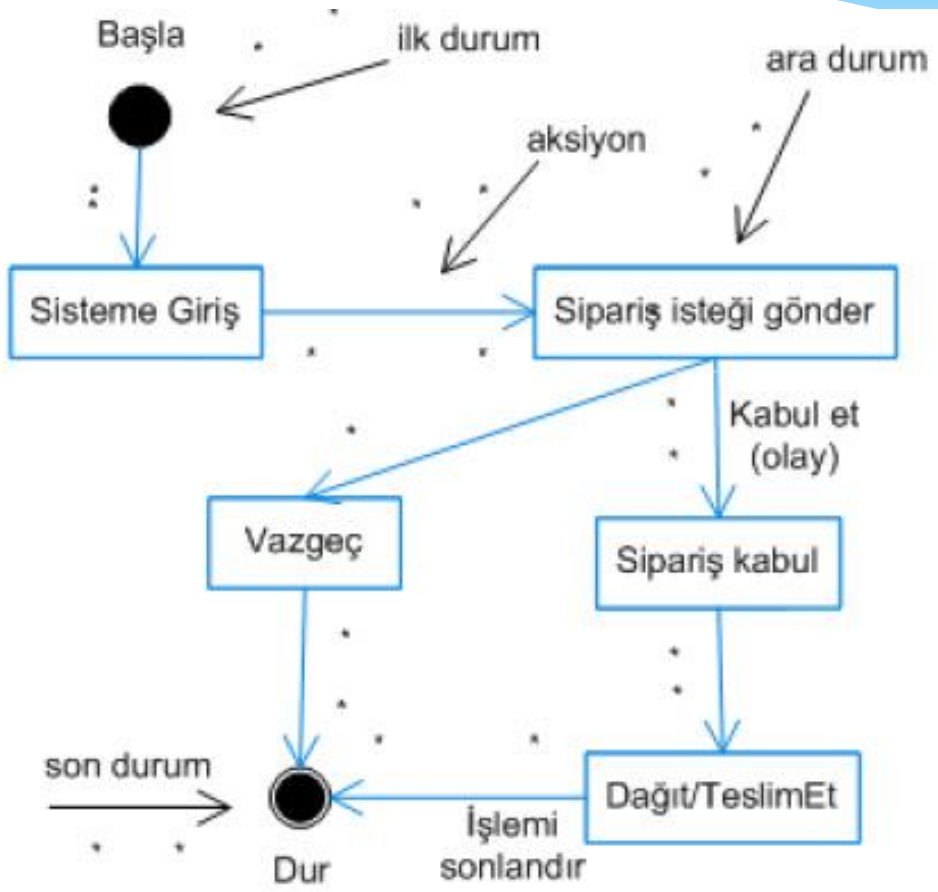
İşbirliği Diyagramları



Durum Diyagramları

* Sistemdeki nesnelerin anlık durumlarını göstermek amacıyla kullanılırlar.
* Sistemin küçük alt sistemlere veya nesnelere ilişkin dinamik davranışlarının ortaya çıkartılması amacıyla yararlanılır
* Sistemin durumları ve bunların birbirlerini tetikleme ilişkileri belirtilir

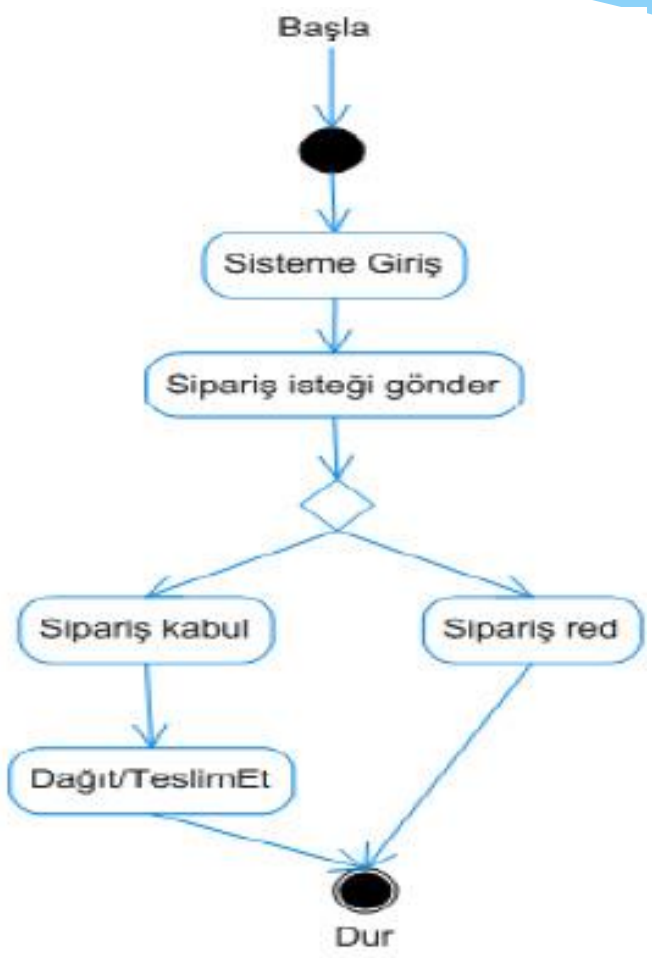
Durum Diyagramları



Etkinlik Diyagramları

* Sistemdeki nesnelerin faaliyetlerini göstermek amacıyla kullanılırlar
* Durum diyagramlarının bir alt kümesi olarak değerlendirilen etkinlik diyagramları, iş akışlarının grafiksel gösteriminde kullanılırlar.

Etkinlik Diyagramları



Kaynakça

* Algan, S. (2002). http://www.csharpnedir.com/articles/read/?id=6
* Kurt, B. Ders notları: http://web.itu.edu.tr/bkurt/Courses/oose/ooswe_release_1_0_6.pdf
* Bozkır, S. http://www.slideshare.net/aselmanb/uml-ile-modelleme
* Tokdemir, G. ve Çağıltay, N. E. (2010). *Veritabanı Sistemleri Dersi*. Seçkin yayıncılık, Ankara.