Hafta – 9

UML

UML Tarihçesi

- UML: *Unified Modelling Language* (Birleşik Modelleme Dili)
  - 1990'lardan itibaren birçok modelleme metodolojisi geliştirilmiştir.
  - **Booch, Rumbaugh's OMT, Jacobson's, OOSE ← Nesneye Yönelik Tasarım Metodolojileri**
  - Object Management Group (OMG) – www.omg.org
  - UML 1.0 (1997)
  - UML 2.0 (2004)

3

UML Nedir?

- **UML;** Yazılım tasarımı için kullanılan bir modelleme dili
- Tasarlanan sistemin 3 önemli yönünü modeller
  - **Fonksiyonel (işlevsel) model**
    - Kullanıcının bakışı ile sistem (kullanım şekilleri: *use case*)
      - UML Kullanım Şekilleri (*Use Case*) Diyagramları
  - **Nesne modeli**
    - Sistemin yapısı: nesneler, özellikleri, işlemler, yapısal ilişkiler
      - UML Sınıf (*Class*) Diyagramları
  - **Dinamik model**
    - Sistemin iç işleyişi
      - UML Sıra (*Sequence*), Etkinlik (Act*ivity*), Durum (*State*) Diyagramları

4

Nesneler ve Sınıflar

- **Nesne (*object*)**
  - ✓Gerçek dünyada, ayrı ayrı tanımlanabilen her şey bir nesnedir.
  - ✓Modelde, her nesnenin bir kimliği, durumu ve davranışı vardır.
- **Sınıf (*class*)**
  - ✓Gerçek dünyada, benzer karakteristik ve davranışlara sahip nesneler bir sınıf (*class*) ile temsil edilir.
  - ✓Modelde bir sınıf, nesneler tarafından paylaşılan durum ve davranışları temsil eder.

5

Nesne (*Object*)

- Kimlik (*identity*)
  - Nesneyi birtek (*unique*) olarak tanımlar ve onu diğer nesnelerden ayırır
- Durum (*state*)
  - Özellikler (*fields* veya *attributes*) ile belirtilir
  - Özellik =
    - **ad** (*name*) + **tür** (*type*) + **değer** (*value*)
- Davranış (*behavior*)
  - Metotlar (*methods* veya *operations*): nesnenin durum bilgilerine erişebilen ve değiştirebilen işlemler.
  - Metot, metot adı, aldığı parametre türleri, ve döndürdüğü tür ile tanımlanır. Herhangi bir değer döndürmeyen metotlar *void* ile belirtilir.

6

Nesne (*Object*): Örnek

- Öğrenci123456:
  - Durum:
    - ad: “Serdar Doğdu”
    - öğrenciNo: “st123456”
    - yıl: 2005
  - Metotlar:
    - dersEkle()
    - dersSil()
    - danışmanAta()

7

Nesne (*Object*)

- İki nesne:
  - “**eşit**” (*equal*): durumları aynı; özellik değerleri aynı.
  - “**aynı**” (*identical*): aynı nesne
- Metotlar:
  - “erişim metotları” (*accessors*): özellik değerlerini değiştirmeyen metotlar
  - “değişim metotları” (*mutators*): özellik değerlerini değiştiren metotlar

8

Sınıf (*Class*)

- Nesneler (*objects* veya *instances*) **sınıf** (*class*) tanımı kullanarak oluşturulurlar (*instantiation*).
- 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

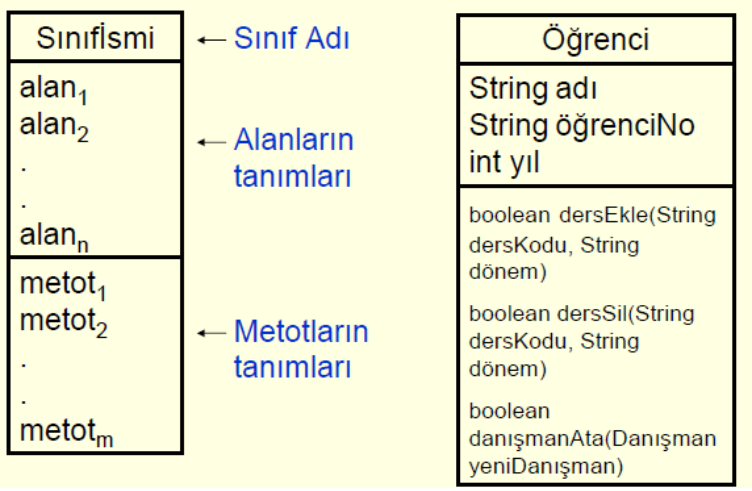
9

Sınıf (*Class*): Örnek

- Öğrenci
  - Alanlar (*fields*)
    - String ad
    - String öğrenciNo
    - int yıl
  - Metotlar (*methods*)
    - boolean dersEkle(String dersKodu, String dönem)
    - boolean dersSil(String dersKodu, String dönem)
    - boolean danışmanAta(Danışman yeniDanışman)

10

UML Sınıf Gösterimi



11

UML Sınıf Tanımları

- Alanlar
  - [Erişim][Tür] Ad [[Sayı]] [=İlkDeğer]
  - Örnek: private String ad = "Serdar"
  - Standart UML gösterimi
    - [Erişim] Ad [[Sayı]] [:Tür] [=İlkDeğer]
    - Örnek: private yıl:int = 2005
- Metotlar
  - [Erişim][Tür] Ad ([*Parametre*, ...])
  - Örnek: public boolean dersAl (String dersKodu)
  - Standart UML gösterimi
    - [Erişim] Ad ([*Parametre*, ...]) [:Tür]
    - Örnek: public danışmanAta(Danışman d):boolean

12

Erişim (*Visibility*) Belirleyiciler

Public: diğer sınıflar erişebilir.

Package: aynı paketteki (*package*) diğer sınıflar tarafından erişilebilir.

Private: yalnızca içinde bulunduğu sınıf tarafından erişilebilir (diğer sınıflar erişemezler).

Protected: aynı paketteki (*package*) diğer sınıflar ve bütün alt sınıflar (*subclasses*) tarafından erişilebilir.

Protected Internal: Sadece tanımlandığı sınıfta ya da o sınıfı miras alan sınıflardan erişilebilir. Ayrıca, tanımlamanın aynı proje içerisinde olma şartı yoktur. *Protected*'dan farkı budur.

13

UML Alan Örnekleri

```
Date doğumGünü      (Java)
doğumGünü:Date      (UML)

public int süre = 100 (Java)
+süre:int = 100       (UML)

private Öğrenci
  öğrenciler[0..MAX_ÖĞR]        (Java)
-öğrenciler[0..MAX_ÖĞR]:Öğrenci(UML)
```

14

UML Metot Örnekleri

```
boolean dersEkle(String dersKodu,
  String dönem)                    (Java)
~dersEkle(String dersKodu, String
  dönem):boolean                   (UML)

public int getSize()               (Java)
+getSize():int                     (UML)
```

15

UML Sınıf Örnekleri

UML (Java syntax):

Point
private int x private int y
public void move(int dx, int dy)

UML (standart):

Point
- x:int - y:int
+move(dx:int, dy:int)

UML (kısaltılmış):

Point veya

Point
x y
move()

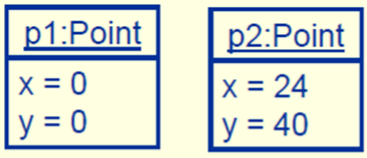
16

UML Nesne Gösterimi

17

UML Nesne Gösterimi

UML



Java

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

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

18

UML Mesaj Geçme

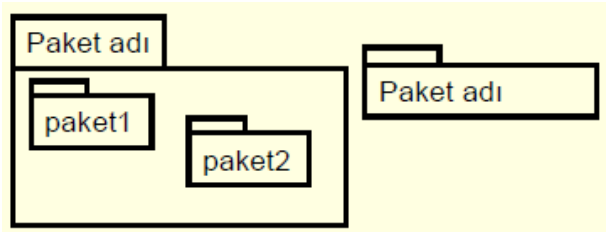
- Nesneler birbirleriyle mesaj geçerek iletişim kurarlar
  - Mesaj geçme (*message passing*) veya
  - Metot çağırma (*method invocation*)
- Mesajı bir nesneye gönderilir (alıcı - *recipient*) ve alıcı nesne çağırılan metodu çalıştırır
- Örnek: `p1.move(10, 20)`
  (p1 nesnesi: x'de 10, y'de 20 pixel kay)
  Alıcı: `p1`, metot: `move()`, parametreler: `(10,20)`

19

UML Paket Gösterimi

- Birbirleriyle **ilişkili sınıflar bir paket (*package*) içine** yerleştirilirler.
- **Paket isimleri küçük harflerle** yazılır.
- Yaygın olarak kullanılacak **paket isimleri internet domain ismini tersten** yazarak kullanırlar.

✓ tr.edu.pau



20

İlişkilerin ve Yapıların Modellenmesi

UML Sınıf Diyagramı

- ***Inheritance*** (Kalıtım)
- ***Association*** (İlişki)
- ***Aggregation*** ve ***Composition*** (Birleştirme ve Kompozisyon)
- ***Dependency*** (Bağımlılık)

21

Kalıtım

- ***Multiple Inheritance*** (çoklu kalıtım): Bir sınıf birden fazla üst sınıftan kalıtım alabilir (*inherit from multiple superclasses*).
- ***Single Inheritance*** (tekli kalıtım): **Çoğu nesnesel programlama dili tekli kalıtıma izin verir.** Java gibi.
- Java'da kısıtlı olarak çoklu kalıta izin vardır; ancak bu arayüzlerden (*interface*) olabilir.

22

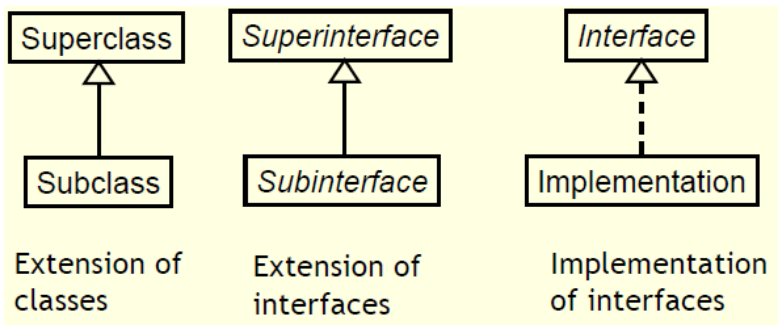
Kalıtım

- Sınıf ve arayüzler arasındaki ilişki;
- **3 tür:**
  - *Extension*: üst sınıf (*superclass*) ve alt sınıf (*subclass*) arasında
  - Arayüzler arası genişletme
  - *Implementation*: bir sınıf bir arayüzü gerçekleştiriyor.
- **UML'ce:**
  - Uzmanlık (genişletme) 
  - Gerçekleştirme (uygulama) 

23

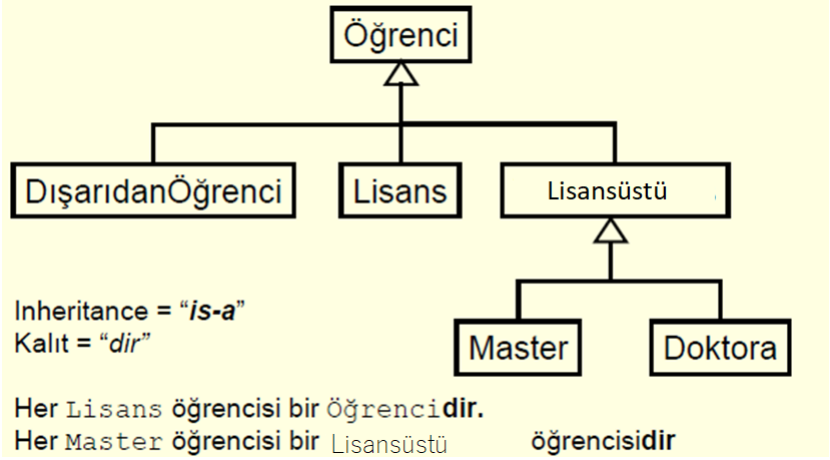
Kalıtım

UML Notasyonu



24

Kalıtım – Örnek



25

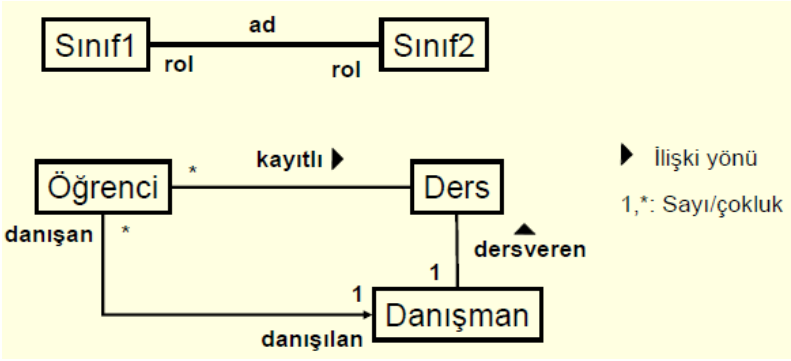
İlişkinin Çokluğu / Sayısı

- ***a .. ü*** **:** alt değerden üst değere kadar.
- ***i*** **:** tek bir değer
- *** :** *0 .. n*
- **Örnekler:**
  - ✓ 0 .. * 0 veya daha fazla
  - ✓ 1 .. * 1 veya daha fazla
  - ✓ 2 .. 5 2'den 5'e kadar
  - ✓ 2, 5, 7 2, 5, veya 7
  - ✓ 1, 3, 5 .. * 1, 3, 5, veya daha fazla

26

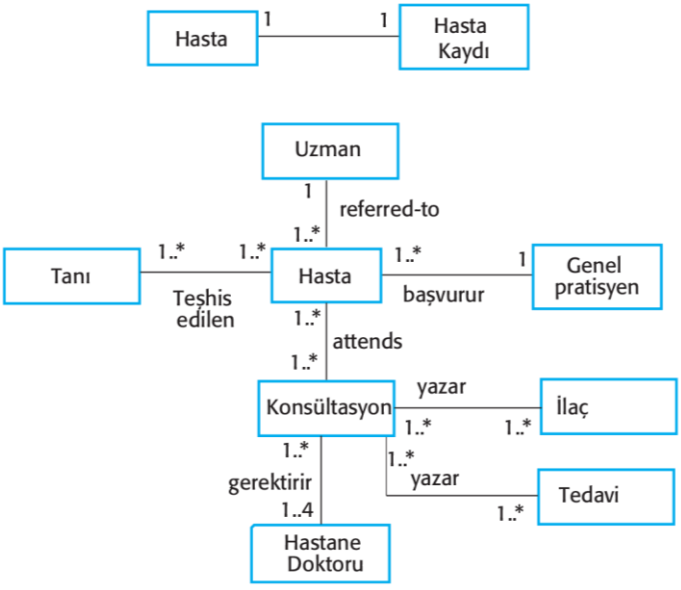
İlişki – Örnek

- Sınıflar arası ikili (*binary*) ilişkiler



İlişki – Örnek

UML Sınıfları ve İlişkiler

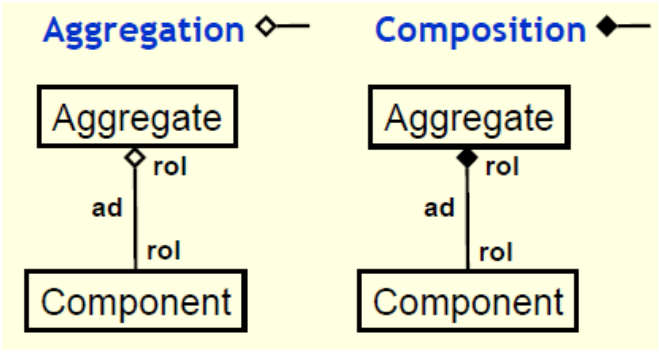


Birleştirme ve Kompozisyon

- İlişkinin özel bir halidir.
- Parça-Bütün (*part-whole*) ilişkisini temsil eder.
- Parça veya bütünün hayat süresi konusunda bir yaptırımı yoktur.
- **Birleştirmenin daha güçlü bir hali komposizyondur (*composition*).**
- Kompozisyonda, parçalar bütün olmadan olmazlar (ortadan kalkarlar).
- İlişkideki çokluk, isimlendirmeler, navigasyon (ilişkilerin yönü) (birleştirme ve kompozisyonda) aynen uygulanır.

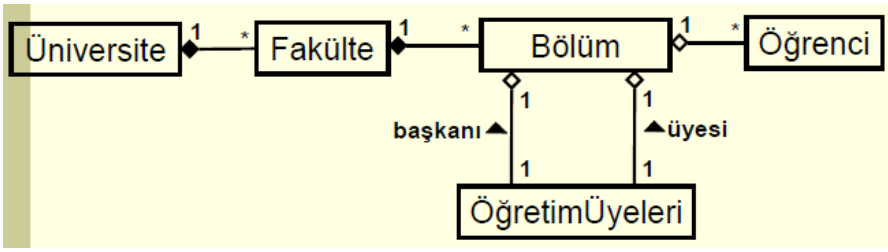
29

Birleştirme ve Kompozisyon



30

Birleştirme ve Kompozisyon – Örnek

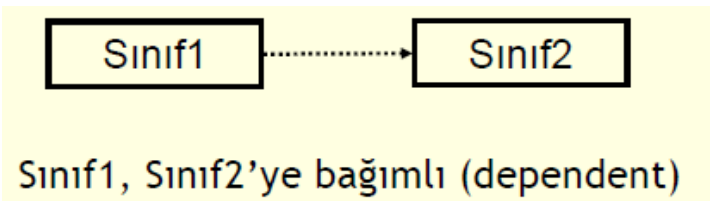


- Fakülte olmadan bölümler **olamaz** (kompozisyon)
- Üniversite olmazsa fakülteler de **olamaz** (kompozisyon)
- Bölüm olmadan öğretim üyeleri ve öğrenciler(?) **olabilir** (birleştirme)

31

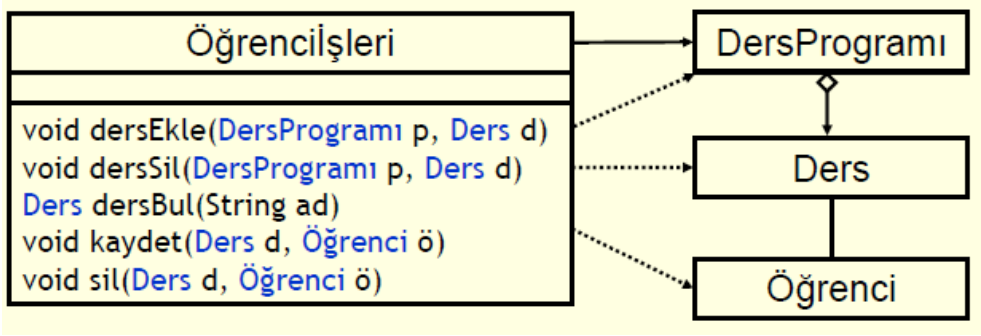
Bağımlılık

- Bir sınıfın diğer bir sınıfa bağlı (*dependent*) olması
- **Örneğin:** bir sınıfın diğer bir sınıfı "kullanması" (*use*) (sınıfın diğer sınıfın metotlarını çağırması gibi, ya da o sınıftan bir nesneyi döndürmesi gibi.



32

Bağımlılık – Örnek



33

Dinamik Davranışların Modellenmesi

- "*Class diagrams*" : statik yapı
- Dinamik davranışlar?

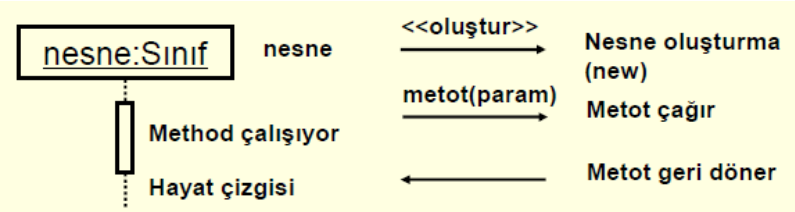
  Nesneler arası aktiviteler, bir nesne üzerinde olayların ve hareketlerin sıralanması
- "*Sequence*" diyagramları
- "*State*" (durum) diyagramları

  İç-içe durum (*nested state*) diyagramları

34

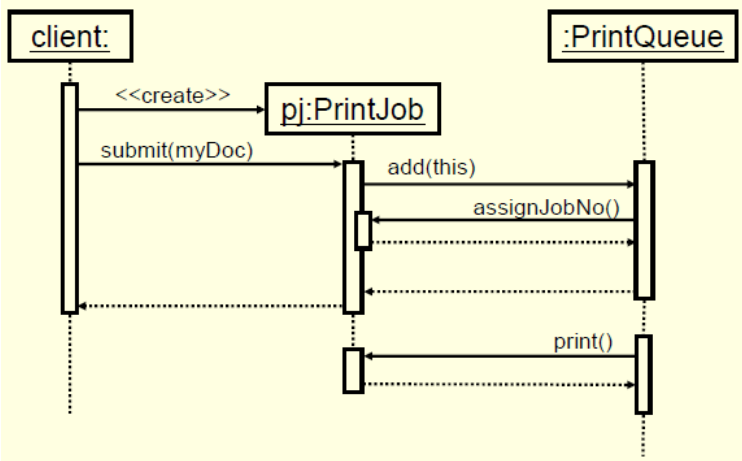
Sıra Diyagramı

- Nesneler arası metot çağırma işlemlerinin zaman çizgisinde sıralı gösterimi
- Zaman çizgisi y-ekseni üzerinde yukarıdan aşağı gösterilir.
- İlgili nesneler x-ekseni üzerinde en üstte soldan sağa sıralanır.



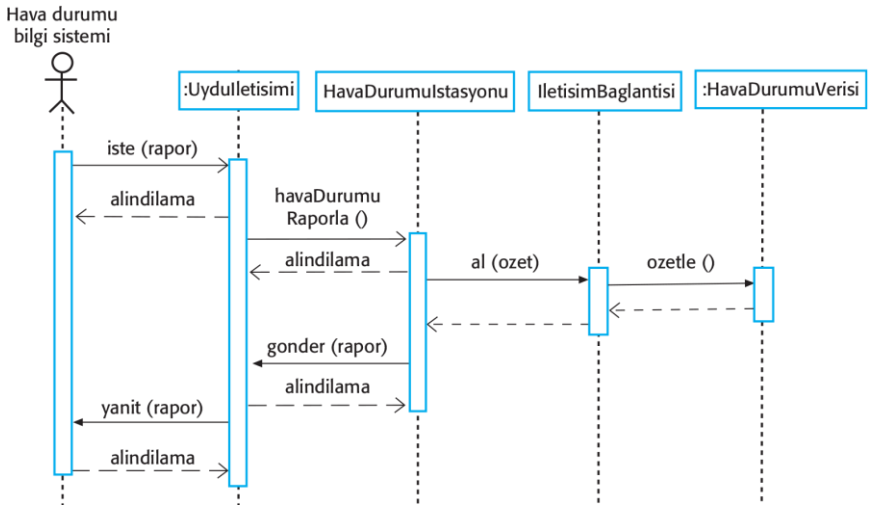
35

Sıra Diyagramı – Örnek



36

Sıra Diyagramı – Örnek



37

Durum Diyagramı

- Sonlu durum makinesinin (*Finite State Machine*) genelleştirilmiş halidir.
- Durumlar (*states*) ve bunlar arasındaki geçişlerin (*transitions*) gösterilmesi
- Geçiş (*transition*): nesnenin bir durumdan diğerine geçmesi. Bu bir olayla tetiklenebilir (*triggered*) ve hiçbir sebep olmadan olabilir (*triggerless*).

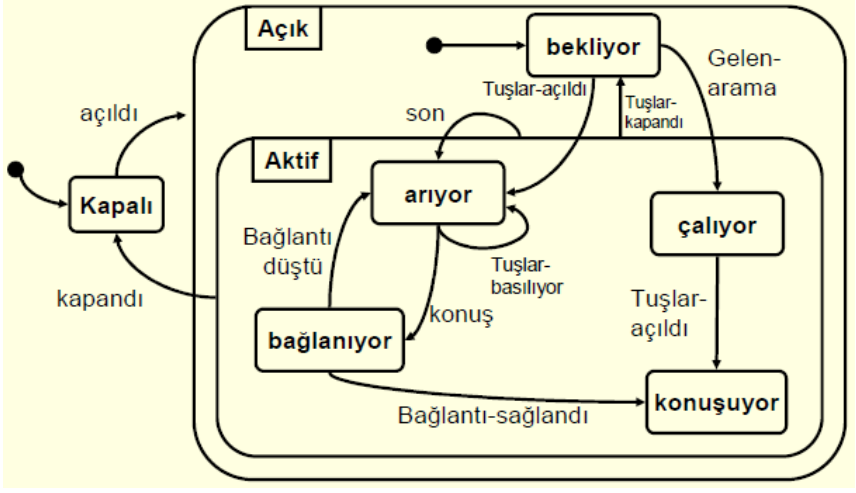
  [Olay-listesi][[Kontrol]][/Aksiyon]

38

Durum Diyagramı

39

Durum Diyagramı – Telefon Örneği



40

Gereksinimlerin Modellenmesi

- Sistem gereksinimleri UML *Use Case* (kullanım şekilleri) diyagramları ile belirtilir.
- Yazılım geliştirme için gerekli değildir, fakat gereksinimler ve nesnesel modeller arasında en önemli bağlantıdır
- ***Use Case:*** Kullanım şekli
  - Bir sistem fonksiyonunun dışarıdan gözlemlenen davranışıdır.
  - Sistemle, sistem dışı aktörler (kullanıcı veya diğer sistemler gibi) arasında etkileşimler.
  - **Sistem "ne" yapıyor ile ilgili,** "nasıl" yapıyor ile ilgili değildir.

41

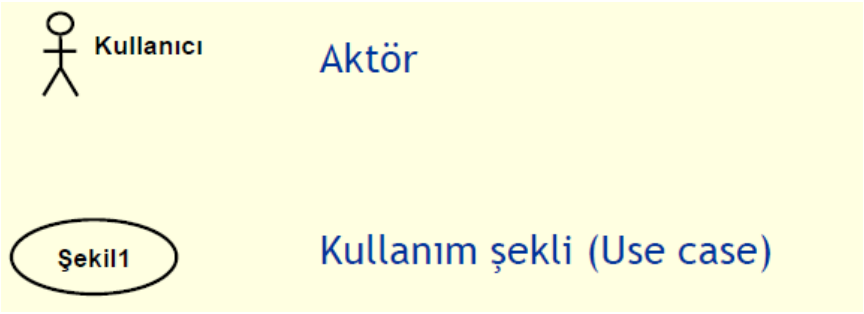
Gereksinimlerin Modellenmesi

- **Use Case:** Kullanım şekli
  - Bir adı var
  - Birkaç senaryodan oluşabilir

    Bunlardan birisi ana senaryo, diğerleri alternatif senaryolar olabilir.

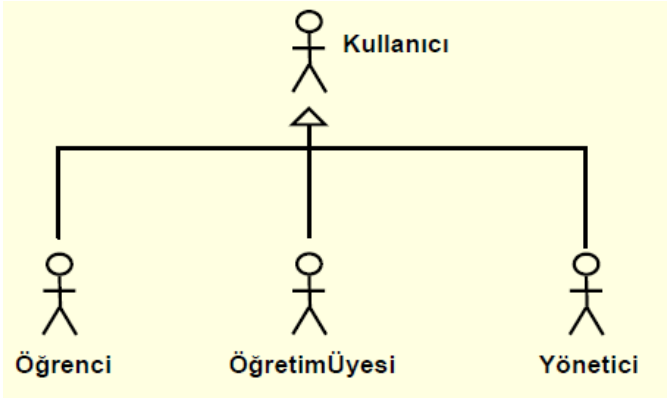
42

Use Case – Grafik Gösterimi



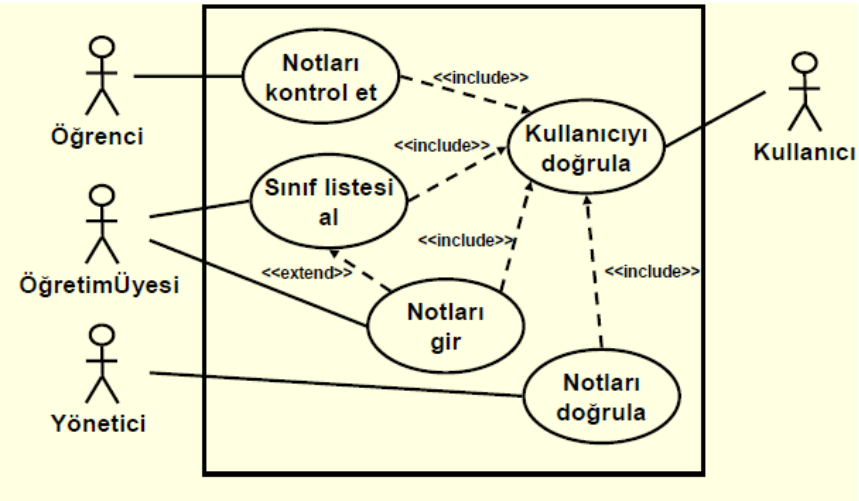
43

Aktörler Arası İlişki



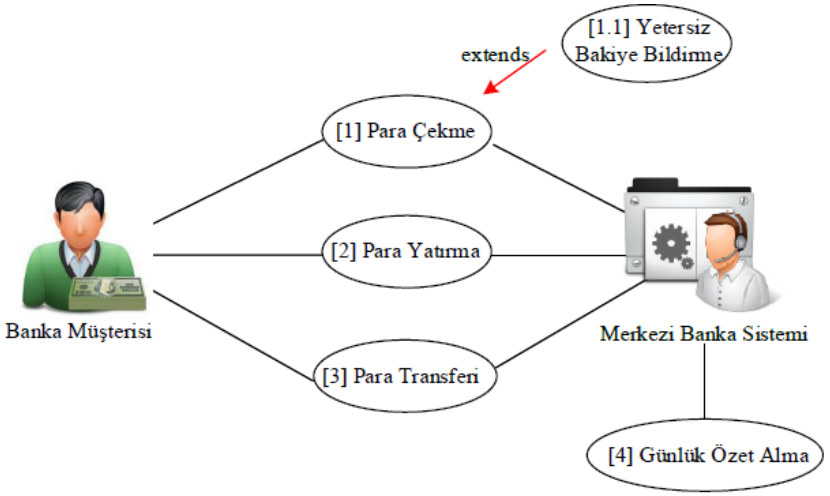
44

Kullanım Şekilleri – Örnek



45

Kullanım Şekilleri – Örnek



46

eKitapçı – Örnek

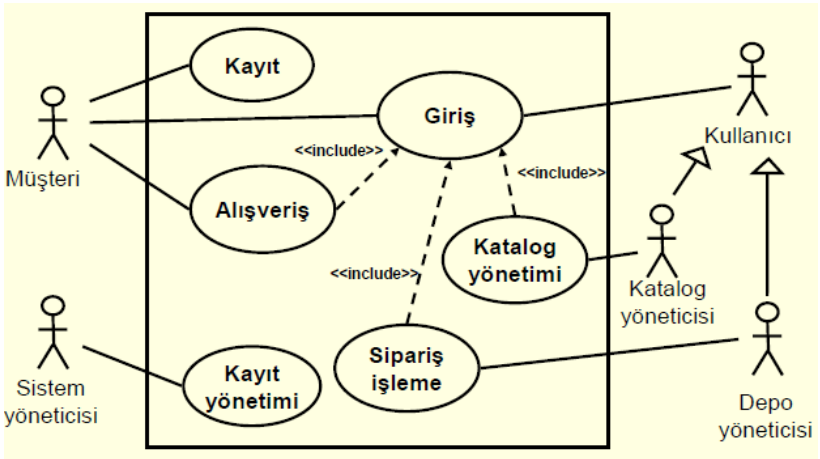
Kavramsallaştırma

- Sitemin temel gereksinimleri:
  - Müşterilerin kitapları, müzik CD'lerini, ve bilgisayar yazılımlarını İnternet üzerinden taramalarını ve satın almalarını sağlamak
- Sistemin temel fonksiyonları:
  - Müşterilere satın almalarında yardımcı olacak bilgileri vermek,
  - Müşterilerin kayıtlarını, siparişlerini, ve adres bilgilerini almak,
  - Sistem yönetimi: kayıt girme, silme, ve değiştirme; müşteri bilgilerinin güncellenmesi

47

eKitapçı – Örnek

Kullanım Şekilleri Diyagramı



48

eKitapçı – Örnek

Kullanım Şekilleri Diyagramı

- Önkoşul (precondition): Müşteri kayıtlı
- Ana senaryo:
  - Aktör: Müşteri

Giriş eylemi	Sistem eylemi ve cevabı
Giriş isteği	Hoşgeldin mesajı, kullanıcı adı ve şifresi isteği
Kullanıcı adı ve şifre girilir	Kullanıcı doğrulanır ve girişe izin verilir
Tekrarla:	
Arama ve listeleme	Satılan şeylerin listelenmesi
Almak üzere seçme	Seçilen şeyin sepete eklenmesi
Alışveriş tamamlandı	Sepettekileri, ödeme/gönderme adresleri göster
Siparişi doğrula, ve ödeme yap	Ödeme metodunu doğrula Ödeme tamamlanır Sipariş işlenir, e-fiş verilir; depoya sipariş emri
Çıkış	

- Alternatif senaryo:
  - Müşteri alışverişi tamamlamadan, alışveriş sepetini saklar, çıkar
- Aykırı senaryo (exceptional):
  - Müşteri girişi başarısız olur; giriş yinelenir
- Aykırı senaryo (exceptional):
  - Ödeme işlemi başarısız olur; müşterinin başka bir ödeme yöntemi girmesi..

49

eKitapçı – Örnek

Nesne Modelleri

- Sınıfların belirlenmesi
  - Use case'lerde geçen sınıfların belirlenmesi
  - Ne sınıftır, hangi özellikler sınıfta olmalıdır?
    - Sınıflar, nesneleri ifade etmelidir, eylemleri değil
      - Fiziki nesneler (araç, gereç, ürün, vs.)
      - Kişiler (öğrenci, öğretim üyesi, müşteri, ve bunların rolleri, gibi)
      - Organizasyonlar (üniversite, şirket, bölüm, vb.)
      - Yer (bina, oda, koltuk, vb.)
      - Olaylar (farenin tıklanması, servis isteği, alım siparişi, vb.)
      - Kavramlar (çokboyutlu uzaylar, işlemler, hava raporu haritaları, vb.)
    - Eylemler, sınıfların metotları olarak modellenmeli
    - Basit kural:
      - Sınıf -> İSİMLER (NOUN)
      - Metotlar -> EYLEM (VERB)

50

eKitapçı – Örnek

Sınıfların Belirlenmesi

- Müşterilerin kitapları, müzik CD'lerini, ve bilgisayar yazılımlarını İnternet üzerinden taramalarını ve satın almalarını sağlamak
- Sistemin temel fonksiyonları:
  - Müşterilere satın almalarında yardımcı olacak bilgileri vermek,
  - Müşterilerin kayıtlarını, siparişlerini, ve adres bilgilerini almak,
  - Sistem yönetimi: kayıt girme, silme, ve değiştirme; müşteri bilgilerinin güncellenmesi

51

eKitapçı – Örnek

Sınıflar

- EKitapçı: Tüm sistem
- Müşteri: eKitapçı müşterileri
- Kitap: eKitapçıda satılan kitaplar
- MuzikCD
- Yazılım

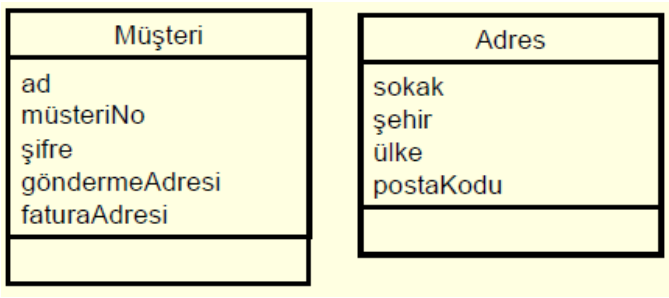
Ayrıca:

- Sepet: Müşterinin almak istediklerini tutan geçici liste
- Sipariş: Müşterinin siparişi
- Adres: Müşteri adresi

52

eKitapçı – Örnek

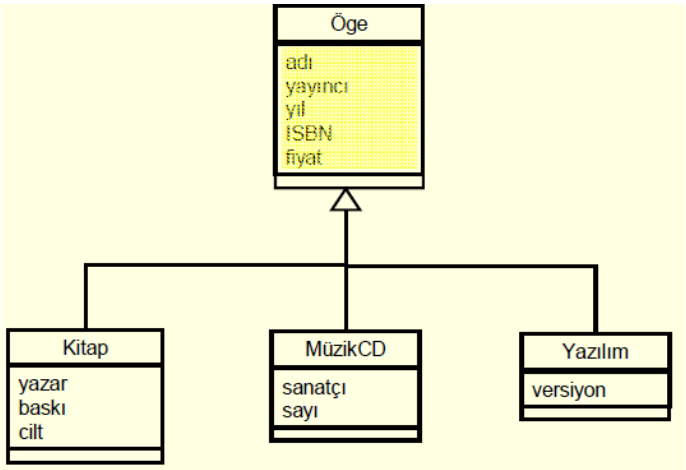
Sınıflar



53

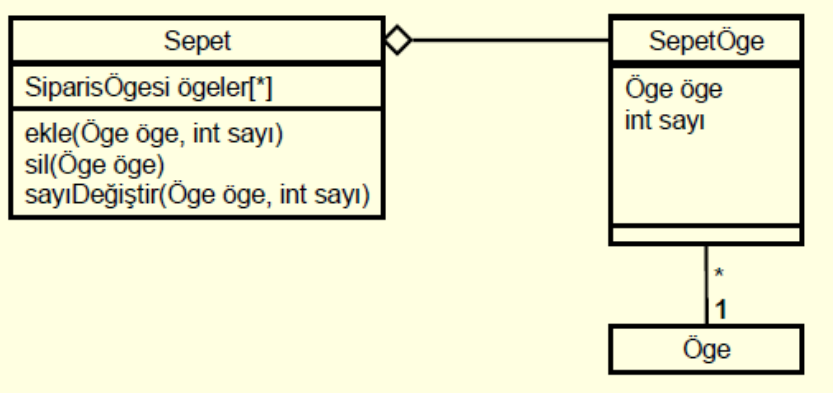
eKitapçı – Örnek

Sınıflar



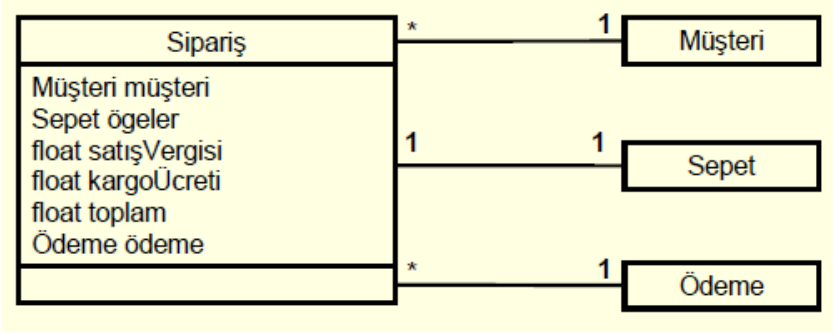
54

eKitapçı – Sınıf Diyagramı



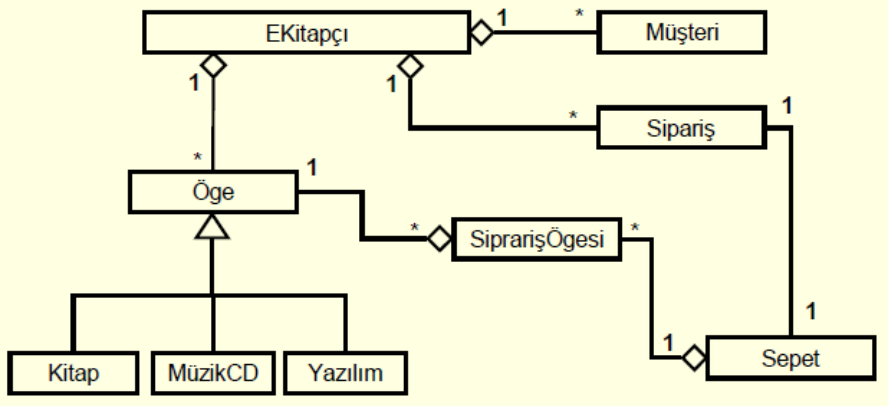
55

eKitapçı – Sınıf Diyagramı



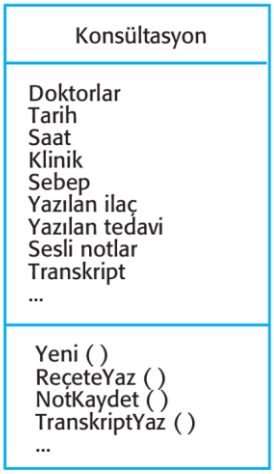
56

eKitapçı – Sınıf Diyagramı



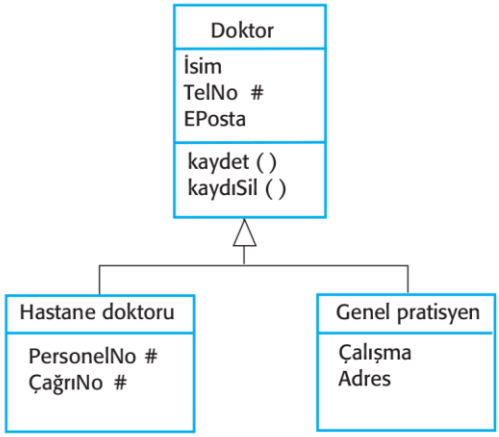
57

Konsültasyon - Sınıf Diyagramı



58

Sınıf Hiyerarşisi



59

Çalıştırılabilir UML

Executable UML

Model güdümlü mühendisliğin arkasındaki asıl fikir modellerden koda tam otomatik dönüşümün mümkün olabilmesidir. Buna ulaşabilmek için şekilsel modelleri, anlamları açık biçimde tanımlanmış ve çalıştırılabilir koda derlenebilecek biçimde kurabilmek gereklidir. Ayrıca şekilsel modellere bu modellerde tanımlanan işlemlerin nasıl gerçekleştirildiği hakkında bilgiler eklemek için bir yönteme ihtiyaç vardır. Bu, UML 2'nin çalıştırılabilir UML veya xUML adı verilen bir alt kümesini kullanarak mümkündür (Mellor ve Balcer 2002).

60

Not

- Bu sunumda;
  - **Erdoğan Doğdu** (*TOBB Ekonomi ve Teknoloji Üniversitesi Bilgisayar Mühendisliğ Bölümü, Ankara)* UML ile Nesnesel Modelleme ders notlarından
  - **Yasemin Topaloğlu** (*Ege Üniversitesi Bilgisayar Mühendisliği Bölümü, İzmir*) Yazılım Mühendisliği kitabı sunumlarından

faydalanılmıştır.

61