






HAFTA 5 – NESNEYE YÖNELİK TASARIM

İÇERİK

Nesneye yönelik tasarım

-  Nesne ve sınıf kavramları
-  Tasarım için kullanılan başlıca UML elemanları
-  Üst düzey tasarım adımları

Örnek çözümleme

-  Kütüphane destek sistemi (KDS)



“NESNE” VE “SINIF” KAVRAMLARI

“NESNE” NEDİR?

👤 İyi tanımlı bir kapsamı ve kimliği olan, belirli bir durum ve davranışı içeren, soyut veya somut varlıktır.

👤 “A discrete entity with a well-defined boundary and identity that encapsulates state and behavior”

👤 Nesne gerçek dünyadaki somut bir varlığı temsil edebilir.

◆ Televizyon, motor, vb.

👤 Nesne tamamen kavramsal bir varlığı temsil edebilir.

◆ Banka hesabı, vb.

“NESNE”: ÖRNEK

Müşteri Hesabı

miktar

paraCek
paraYatir
miktarSorgula

 Her nesne aşağıdakilere sahiptir:

-  Kişilik (“identity”)
-  Özellik (“attribute”)
-  Durum (“state”)
-  Davranış (“behavior”)
-  Sorumluluk (“responsibility”)

ÖZELLİK

👤 Her nesnenin kendine ait bir dizi özelliği vardır.

👤 Özellikler nesneye ait verileri taşır.

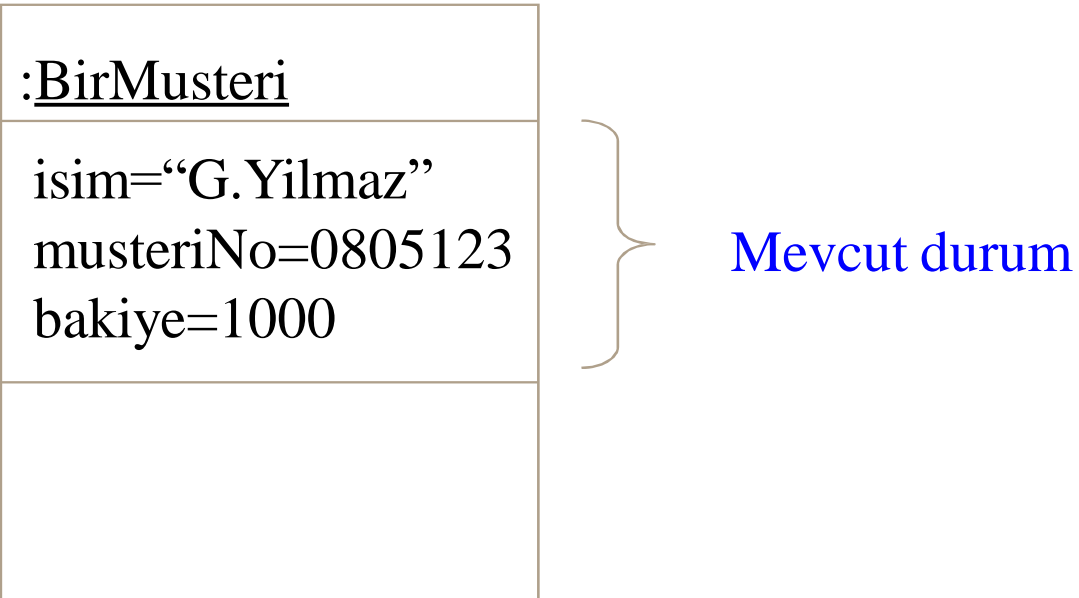
<u>:Arabam</u>
model marka renk

<u>:BirMusteri</u>
isim musteriNo bakiye

<u>:BirPencere</u>
boyut pozisyon renk

DURUM (I)

🏠 Nesnenin tüm özellikleri ve bu özelliklerin o anki değerleri, nesnenin durumunu oluşturur.



DAVRANIŞ

👤 Her nesnenin iş yapabilmeyi sağlayan tanımlı bir davranış şekli vardır.

👤 Nasıl davranır, nasıl tepki verir ? (“act”/“react”)

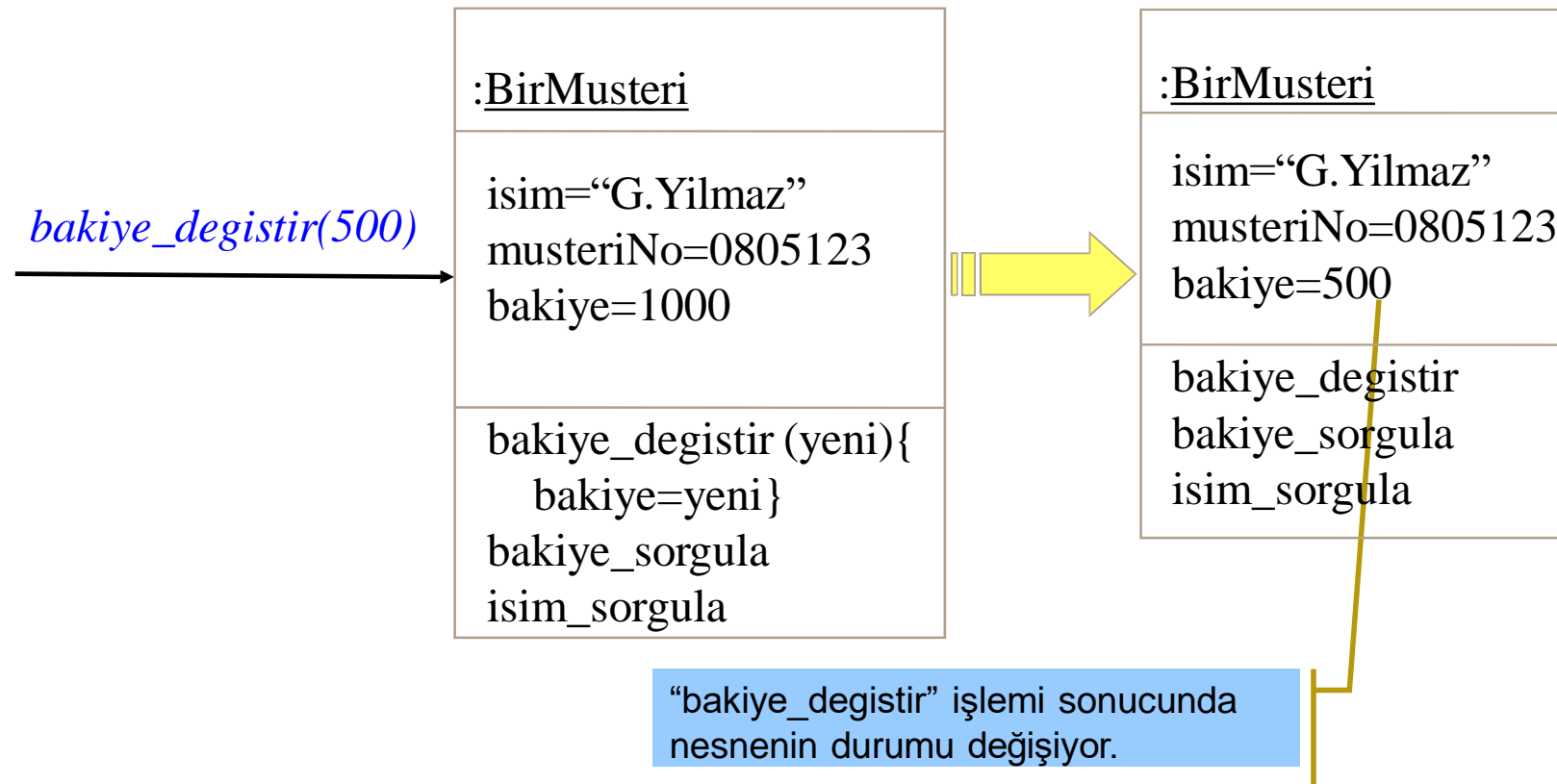
- ◆ Nesnenin operasyonları
- ◆ Görülebilir aktivite
- ◆ Diğer nesneler tarafından kullanılan arayüz (“public interface-operations”)

👤 Davranışın gerçekleştirilmesi bilgisayar kodu ile yapılır.

- ◆ Kodlanan nesne yordamları, davranışı gerçekleştirir.
- ◆ Gizli gerçekleştirme (“encapsulated implementation”)

DURUM (2)

İşlemler sonucunda nesnenin durumu değişir.



SORUMLULUK

👤 Nesnenin sorumluluğu tüm sistemin işlevselliğine nasıl katkıda bulunacağını tanımlar.

👤 Nesnenin sistemde oynayacağı rol

👤 Neden böyle bir nesneye gereksinim var ?

👤 Özellikler ve davranışlar, birlikte nesnenin sorumluluklarını yerine getirmesini sağlar.

İLİŞKİ (“RELATIONSHIP”)

- 👤 Nesneler arasındaki fiziksel veya kavramsal bağlantı
- 👤 En yaygın ilişki: Bir nesne diğer nesnenin sunduğu servislerden yararlanır.
 - 👤 Mesaj iletimi
 - 👤 Müşteri/tedarikçi ilişkisi
- 👤 Sistem işlevselliğini sağlamak amacıyla, nesneler birbirleri ile ilişkiler aracılığı ile işbirliği yaparlar.
 - 👤 Nesneye yönelik programlama modeli

“SINIF” NEDİR?

👤 Yapısal ve/veya davranışsal olarak aynı özelliklere sahip nesneler SNF altında gruplanır.

- 👤 Her nesne bir sınıfın örneğidir (“instance”).
- 👤 Sınıfları tanımlar ve nesneleri sınıf tanımından örnekleriz (“instantiation”).
- 👤 Her nesne ait olduğu sınıfı bilir.

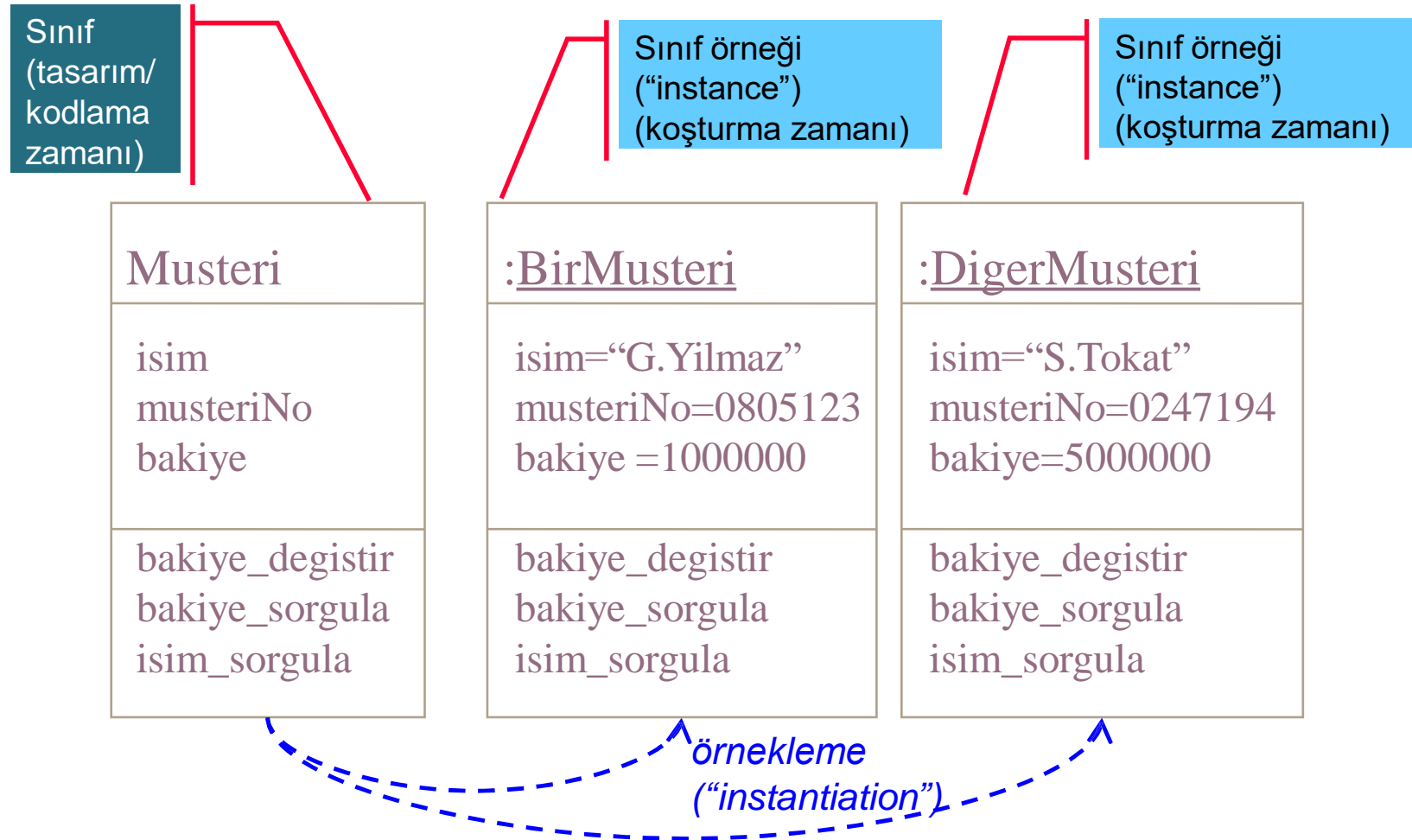
👤 Sınıf, nesneler için şablon tanımdır.

- 👤 Özellikler ve yordamlar sınıf için yalnızca bir kez tanımlanır.

👤 Sınıfların birbirleri arasındaki ilişkileri sistemin sınıf yapısını (“class structure”) oluşturur.

- 👤 İki sınıf arasında ilişki varsa karşılık gelen nesneler arasında da vardır.

“SINIF”: ÖRNEK



“SINIF” VE “NESNE”

👤 Her sınıfın sıfır veya daha fazla örneği vardır.

👤 Sınıf statik, nesne dinamiktir.

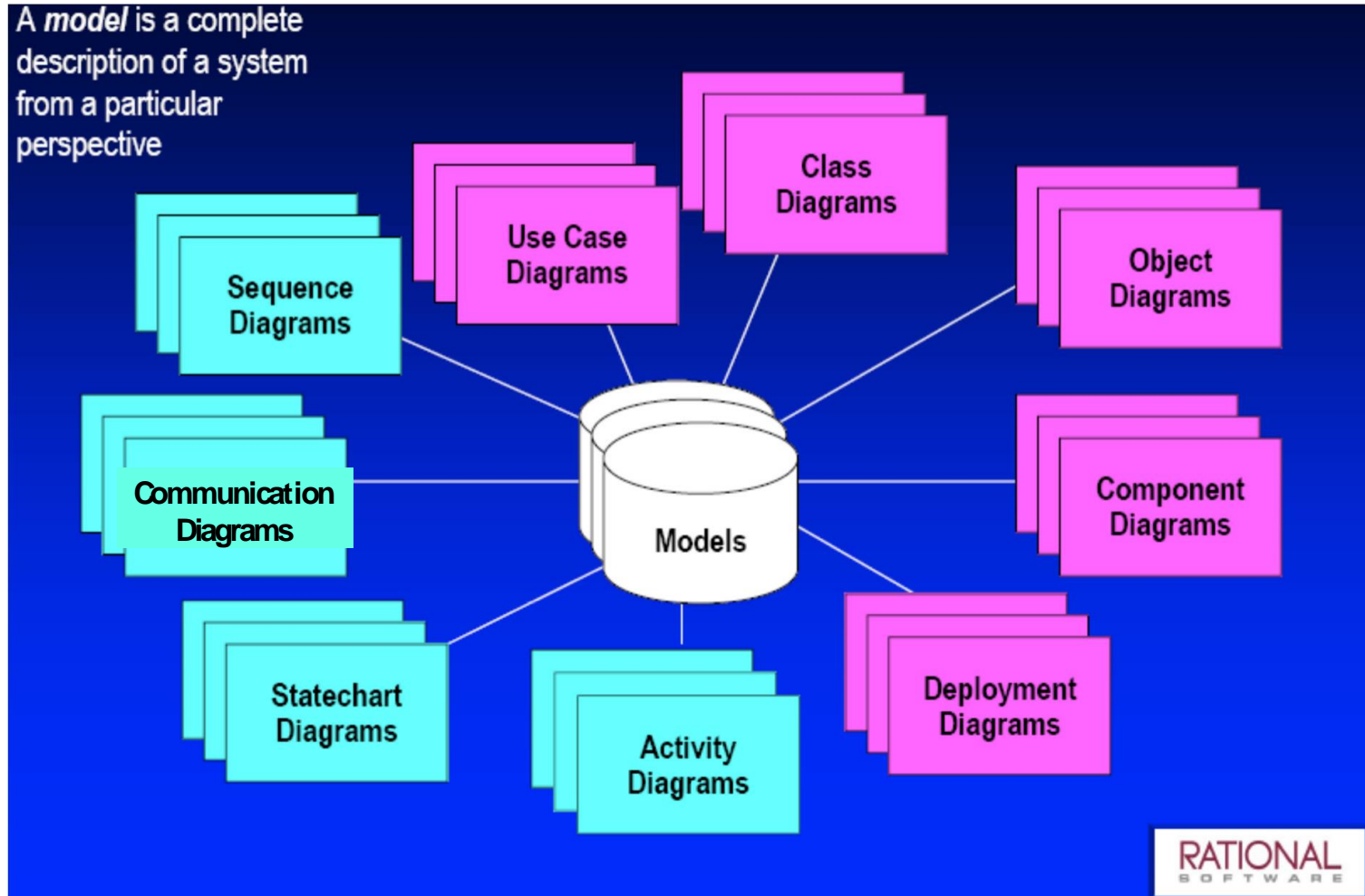
- 👤 Sınıfın varlığı, semantiği ve ilişkileri program koşturulmadan önce sabit olarak belirlenmiştir.
- 👤 Nesneler program koşturulduğunda sınıf tanımından dinamik olarak yaratılırlar (“construction”).
- 👤 Nesneler sorumluluklarını tamamladıklarında ortadan kaldırılırlar (“destruction”).

👤 Nesnenin sınıfı sabittir ve nesne bir kez yaratıldıktan sonra değiştirilemez.



NESNEYE YÖNELİK TASARIM İÇİN KULLANILAN BAŞLICA UML ELEMANLARI

UML DİYAGRAMLARI





TASARIM İÇİN KULLANILAN BAŞLICA UML ELEMANLARI (I)



Sınıf diyagramı (“class diagram”)

-  Sistemi oluşturan sınıflar ve bunlar arasındaki ilişkiler

Ardıl-işlem diyagramı (“sequence diagram”)



-  Nesneler arası etkileşim (davranış)
-  Nesneler arası kontrolün akışı (zamana göre sıralı)

İletişim diyagramı (“communication diagram”)

-  Nesneler arası etkileşim (davranış)
-  Nesneler arası kontrolün akışı (mesaj sıralı)

TASARIM İÇİN KULLANILAN BAŞLICA UML ELEMANLARI (2)



Durum diyagramı (“statechart diagram”)

-  Dinamik davranış, olay (“event”) esaslı
-  Bir nesnenin iç davranışı (nesne yaşam döngüsü)

Etkinlik diyagramı (“activity diagram”):

-  Operasyon akışları

Bileşen diyagramı (“component diagram”):

-  Uygulamanın fiziksel yapısı
-  *Gerçekleştirme aşamasında*

Yayılma diyagramı (“deployment diagram”):

-  Donanım topolojisi
-  *Tasarım ve gerçekleştirme aşamalarında*



ÜST DÜZEY TASARIM

ÜST DÜZEY TASARIM

👤 Amaç: Sistemi oluşturacak temel varlıkları ve ilişkilerini tanımlamak

👤 **Sistemin sınıf yapısı**

- ◆ Uygulama alanına özgü sınıflar (“domain classes”)
- ◆ Sınıflar arası ilişkiler

👤 **Davranış modeli**

- ◆ Sınıfların işlevsel gereksinimleri karşılamak için birbirleri ile nasıl işbirliği yapacakları

👤 *Literatürde “OO Analysis” olarak da geçiyor*

“USE CASE” VE TASARIM MODELLERİ

USE-CASE MODELİ :

- 👤 Doğal dil
- 👤 Dış bakış açısı (“black box”)
- 👤 “Use case” ile yapılandırılır
- 👤 Müşteriler esaslı
- 👤 İşlevselliğin “ne” olduğunu tanımlar

TASARIM MODELİ :

- 👤 Yazılım modelleme dili
- 👤 İç bakış açısı
- 👤 Sınıflar ile yapılandırılır
- 👤 Geliştiriciler esaslı
- 👤 İşlevselliğin sistemde “nasıl” gerçekleştirileceğini tanımlar

ÜST DÜZEY TASARIM ADIMLARI

1. Sınıfları belirle

- ☞ Amaç: “Use case” tanımlarından analiz sınıflarını (“domain classes”) sorumluluklarını belirlemek
 - ◆ Odak noktası sınıfları belirlemektir (detaylı tanımların sonra yapılması önerilir)

2. Sınıf yapısını belirle

- ☞ Amaç: Belirlenen sınıfların “use case”lerde tanımlanan iş adımlarını gerçekleştirmek için nasıl etkileşimde bulunacağını saptamak
 - ◆ “Use case”ler gözden geçirilerek belirlenen sınıfların arasındaki ilişkiler tanımlanır

3. Davranışı modelle

- ☞ Amaç: “Use case”lerin içerdiği işlevselliğin, belirlenen sınıfların davranışları aracılığıyla nasıl gerçekleştirileceğini tanımlamak
 - ◆ Nesnelerin belirlenen işi yerine getirmek için nasıl mesajlaşacakları belirlenir
 - ◆ Nesneler arası kontrol akışı dinamik olarak gösterilir

ADIM-I. SINIFLARI BELİRLE (I)

- 👤 Use case modelinde yer alan her use case tanımı gözden geçirilir
 - ☞ Analiz sınıfı (“domain class”) özelliklerini taşıyan varlıklar belirlenir
 - ☞ Belirlenen sınıflara sorumluluk atanır
 - ◆ Use case tanımlarındaki işler esas alınır
 - ☞ Önceki use case’lerde belirlenmiş sınıfları kullanmaya çalışırız
 - ◆ Analiz sınıfları büyük olasılıkla birçok use case’de karşımıza çıkacaktır

ADIM-I.SINIFLARI BELİRLE (2)

👤 Use case tanımlarının üzerinden geçerken:

- 👤 Kullanılan “isimler” aday sınıf olarak ele alınır
- 👤 Aday sınıflar belirli kurallar ve gereksinimler doğrultusunda elenir
- 👤 Elemeler sonucunda oluşan “isim” kümesi sistemin sınıfları olarak ele alınır
- 👤 Elemelerden geçemeyen “isim”ler belirlenen sınıfların özellikleri olabilirler
- 👤 İşi tanımlayan “fiil”ler sınıfların sorumluluklarını veya sınıflar arası ilişkileri tanımlar

ADIM-I. SINIFLARI BELİRLE (3)

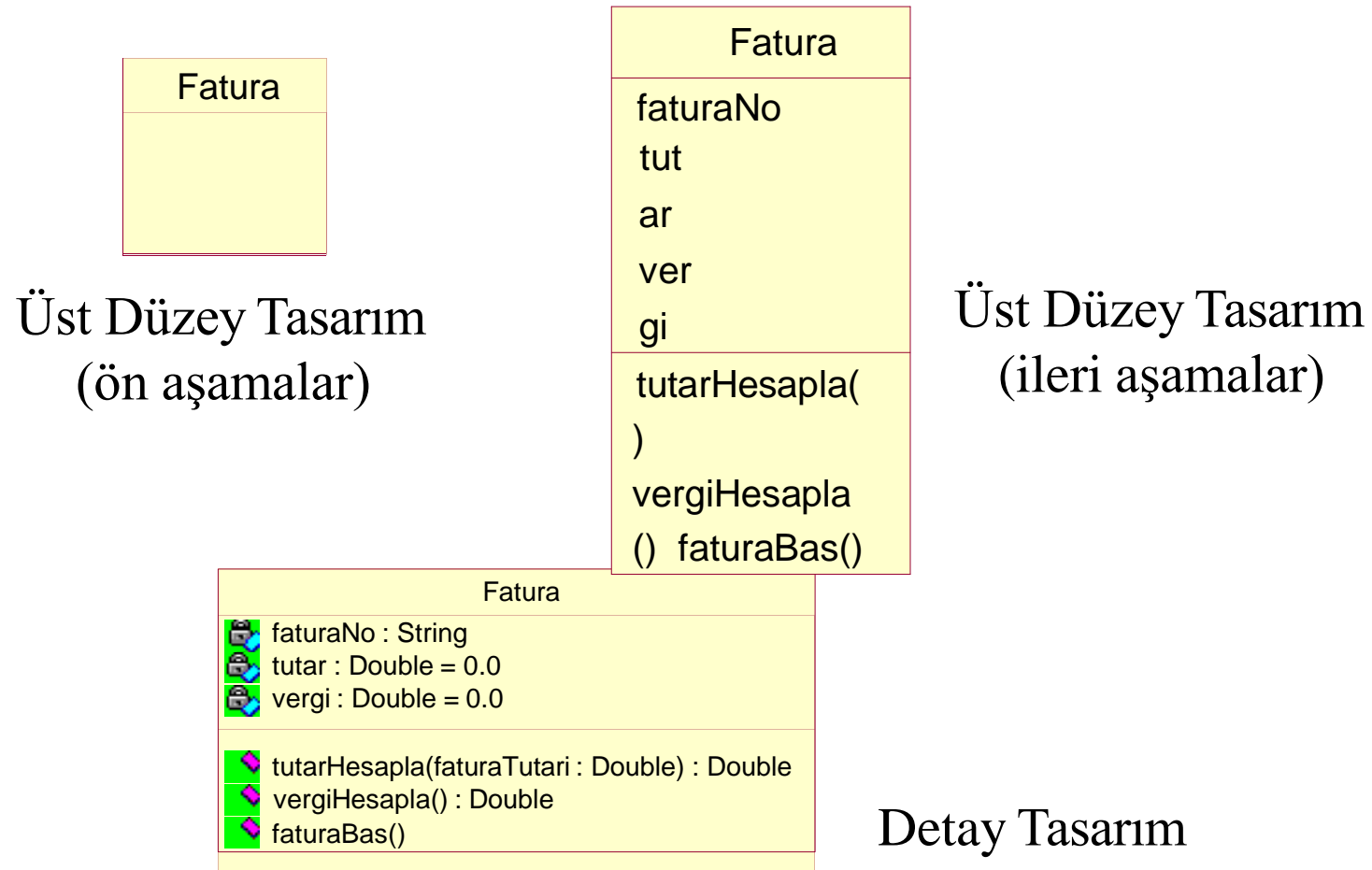
👤 Belirlenen her sınıf için, sınıfın özellikleri tanımlanır:

- ☞ Özellikler “use case” tanımlarında doğrudan yer alabilir
- ☞ Özellikler sınıfın sorumluluklarından çıkarılabilir
- ☞ Bu aşamada özellikler ile ilgili detaylı bir tanımlamaya gerek yoktur
 - ◆ Veri yapılarına girilmeden sadece isimlendirilir

👤 Belirlenen her sınıf için sınıfın davranışı, sorumluluklar esas alınarak operasyon halinde tanımlanır

- ☞ “Use case”de tanımlanan işlevsellik, sınıflar arasında paylaştırılır
- ☞ Sorumluluk, sınıfın farklı “use case”lerde aldığı rollerin birleşimidir
 - ◆ Her rol bir operasyon olabilir
- ☞ Bu aşamada “class methods”, “signatures” detaylarına girilmez

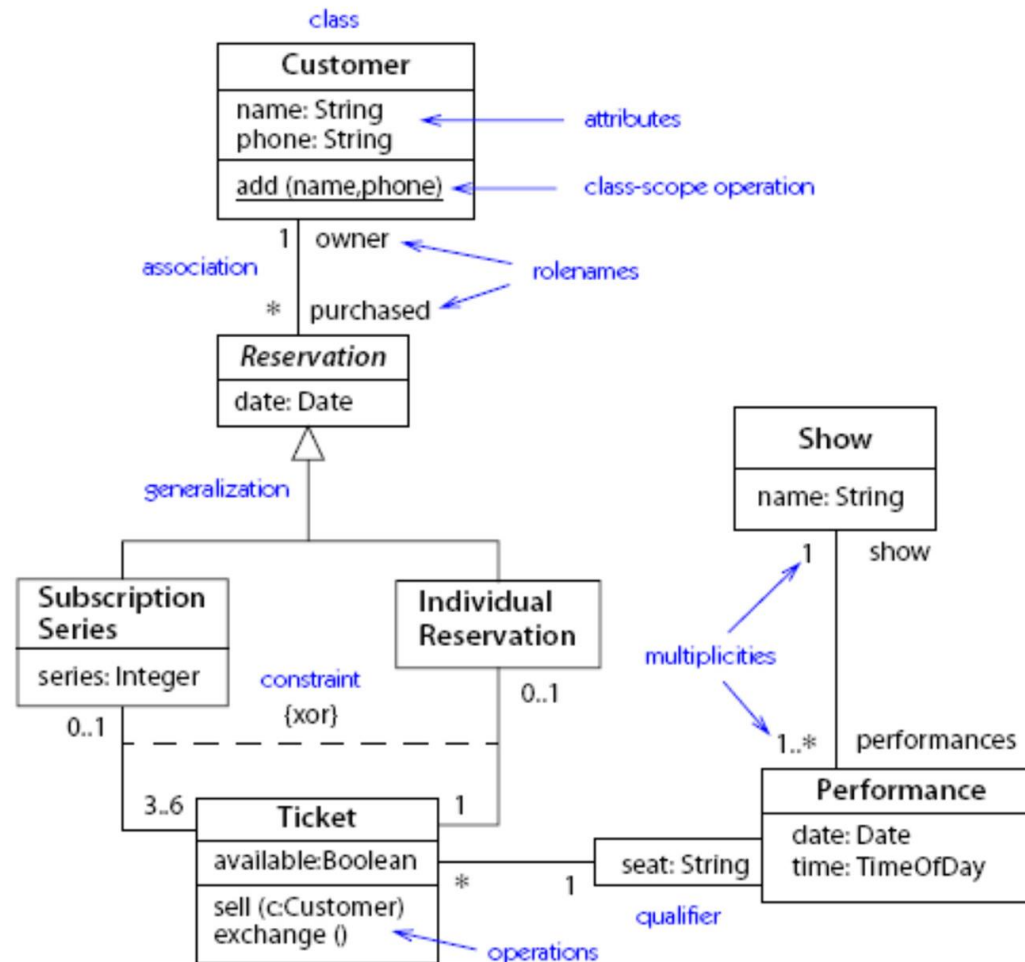
TASARIM BOYUNCA SINIF TANIMININ GELİŞİMİ: ÖRNEK



ADIM-2. SINIF YAPISINI BELİRLE

- 👤 Belirlenen ilişkiler davranış modellemesine esas oluşturur
 - 👤 “Fiil”ler ilişkilerin göstergesidir
- 👤 Sınıflar arası ilişkiler kavramsal olarak tanımlanır
 - 👤 İlişkinin türlerine ve detay özelliklerine girilmez
- 👤 Sınıflar arası ilişkiler en az sayıda tutulmaya çalışılır
- 👤 Tasarım süreci devam ettikçe basit ilişkiler özel formlara dönüşebilir
 - 👤 “association”, “aggregation”, “composition”, “generalization”, vb.

“STATIC VIEW” “CLASS DIAGRAM”: ÖRNEK



SINIF DİYAGRAMI

👤 Sınıf diyagramı “statik” bakış açısı sunar

🗺️ Yol haritası gibi

- ◆ Nesneler şehirleri, ilişkiler şehirler arasındaki yolları gösterir
- ◆ Hedefe ulaşmak için hangi yolun takip edilmesi gerektiğini söylemez

👤 Sınıf diyagramında hangi nesnelerin işbirliği yaptığı belli, ancak nesnelerin nasıl işbirliği yapacakları belli değildir

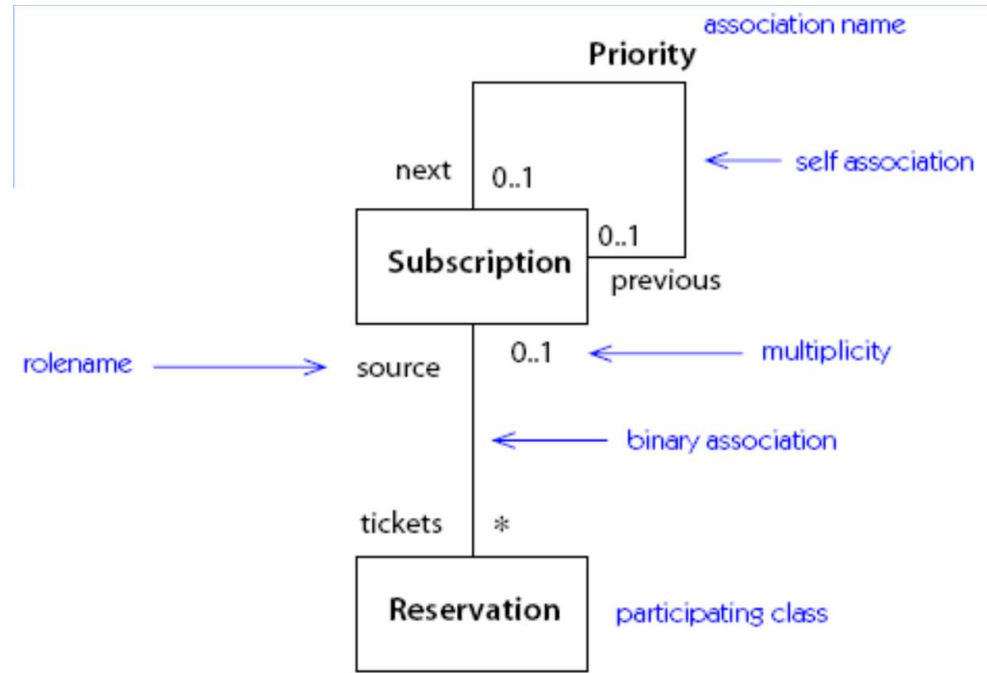
🗺️ Nasıl sorusunun cevabı ardıl-işlem (“sequence”) veya iletişim (“communication”) diyagramlarında tanımlanır

👤 Sınıf diyagramları geliştirme boyunca kullanılır

👤 Sınıf diyagramı, sistem için tanımlanan tüm sınıfları içermeyebilir

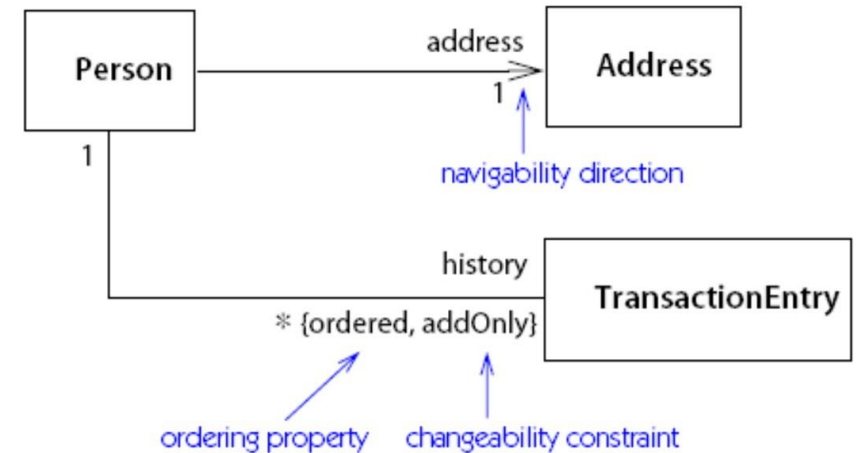
- 🗺️ Sistem modeli farklı sınıf diyagramlarından oluşabilir
- 🗺️ Her sınıf diyagramının bir görevi vardır
 - ◆ “Collaboration”, “generalization”, vb.

“ASSOCIATION” (I)

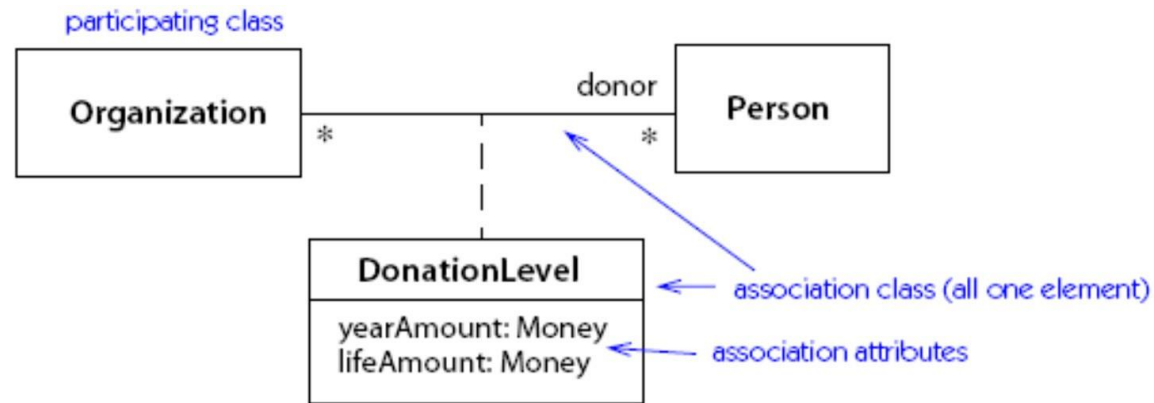


“association”

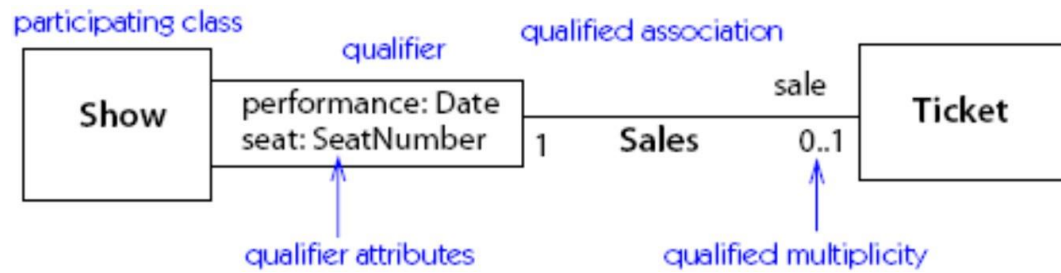
“design properties of an association”



“ASSOCIATION” (2)



“association class”

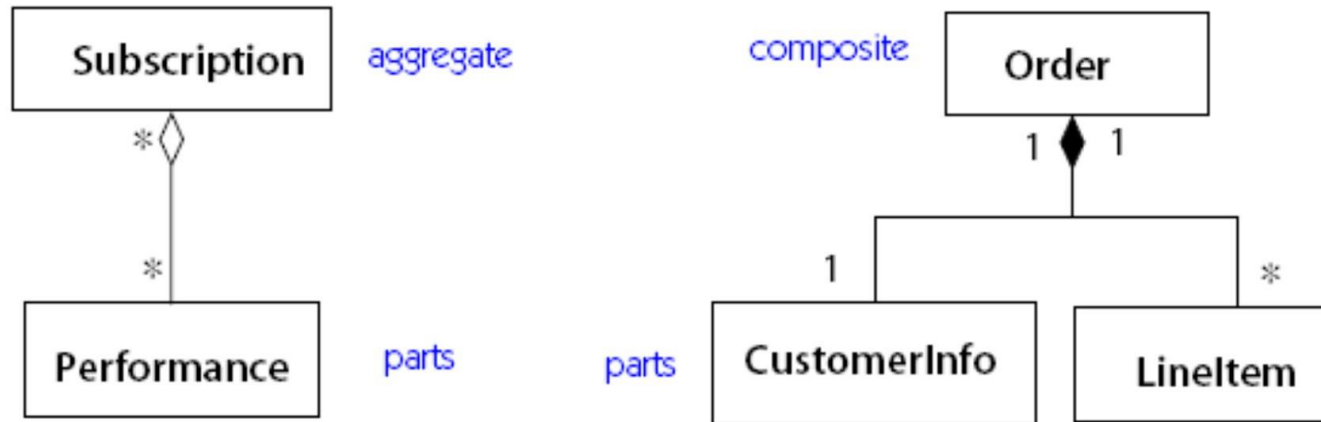


“qualified association”

“A *qualifier* is a value that selects a unique object from the set of related objects across an association. (e.g., lookup tables and arrays)”

“AGGREGATION” VE “COMPOSITION”

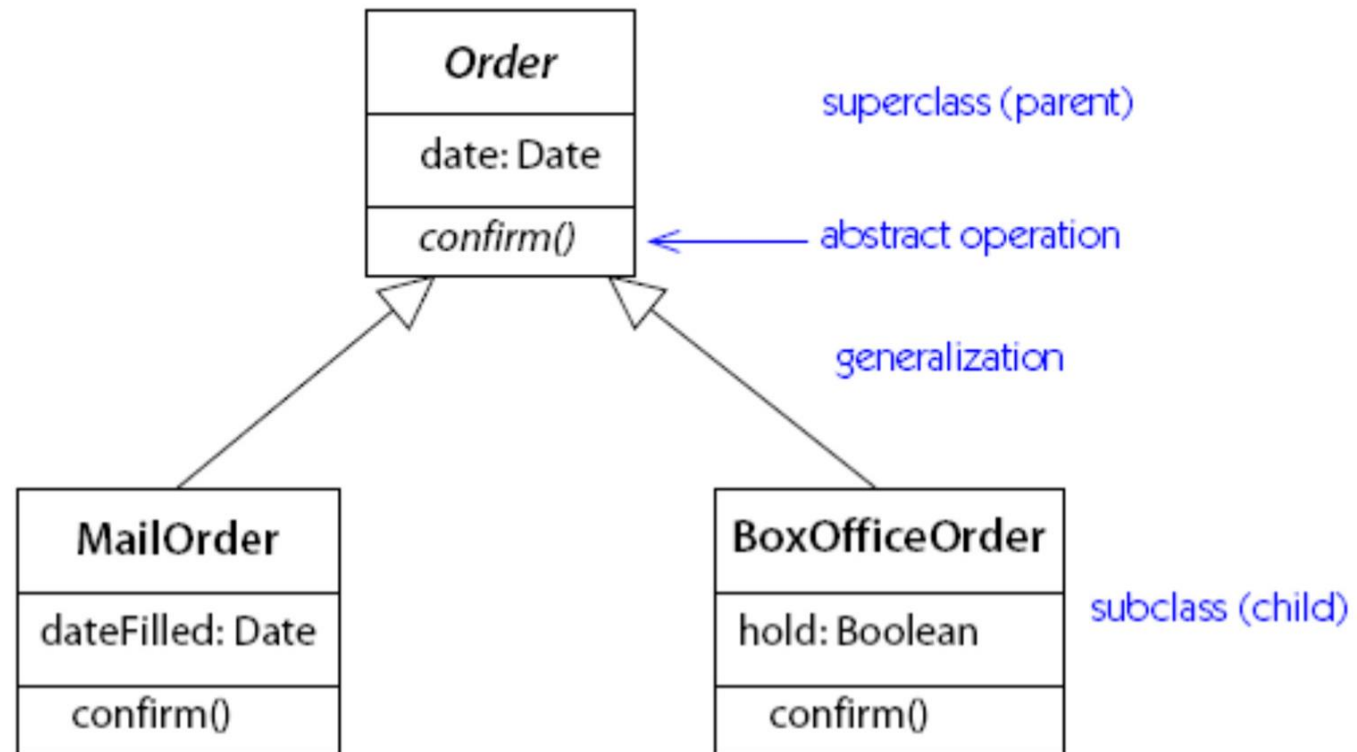
🏠 Bütün-parça ilişkisi



“An *aggregation* is an association that represents a part-whole relationship.”

“A *composition* is a stronger form of association in which the composite has sole responsibility for managing its parts—such as their allocation and deallocation.”

“GENERALIZATION”



ADIM-3. DAVRANIŞI MODELLE

👤 Davranışlar iki şekilde tanımlanabilir:

- ☞ Ardıl-işlem (“sequence”) diyagramları ile
 - ◆ Nesneler arası iletişimin zamana göre sıralaması
- ☞ İletişim (“communication”) diyagramları ile
 - ◆ Nesnelerin etkileşimde aldığı roller ve ilişkiler
 - ◆ Nesneler arası iletişimin etkileşime göre sıralaması

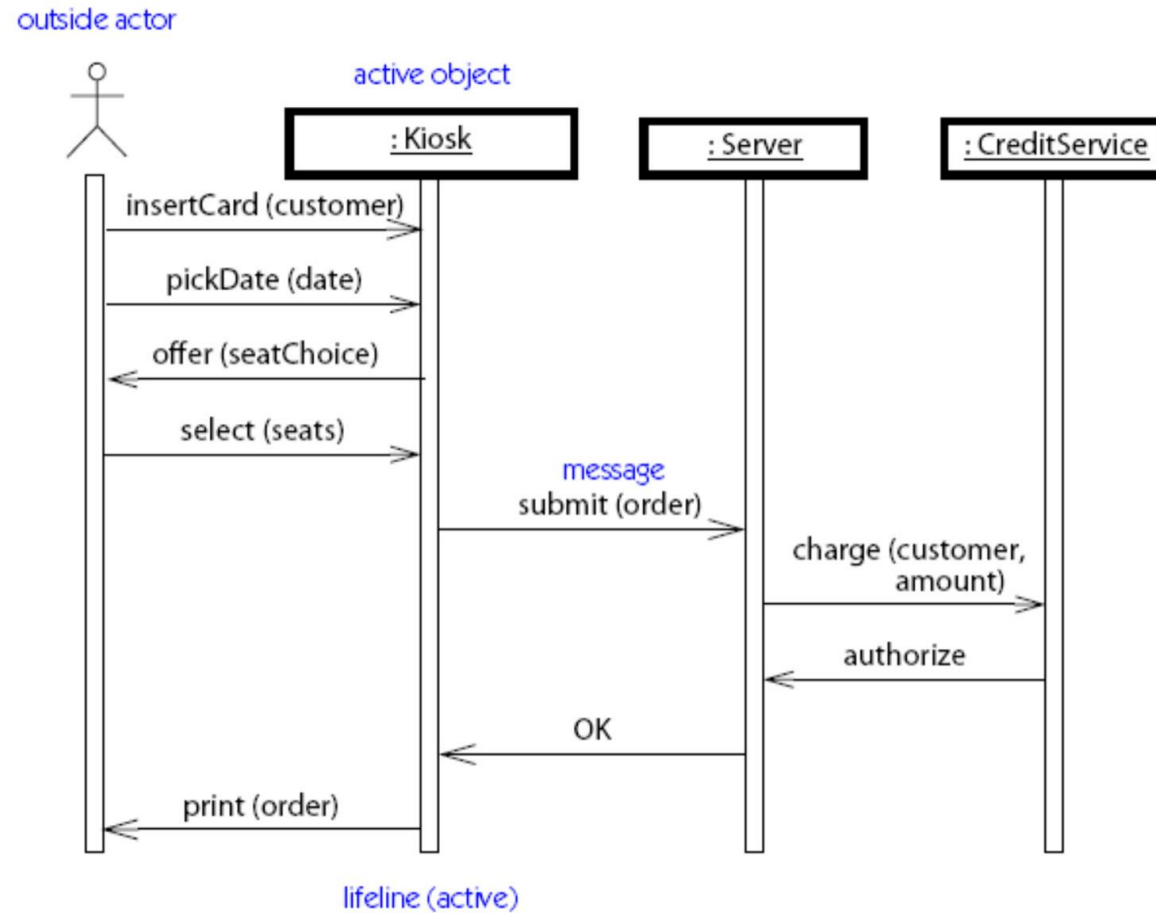
👤 Önerilen yöntem:

- ☞ Bir “use case” veya senaryo seçilir
- ☞ Hangi nesnelerin rol alacağı belirlenir
- ☞ Bir nesneden diğerine gönderilecek mesajlar zamana veya etkileşime göre sıralanarak gösterilir

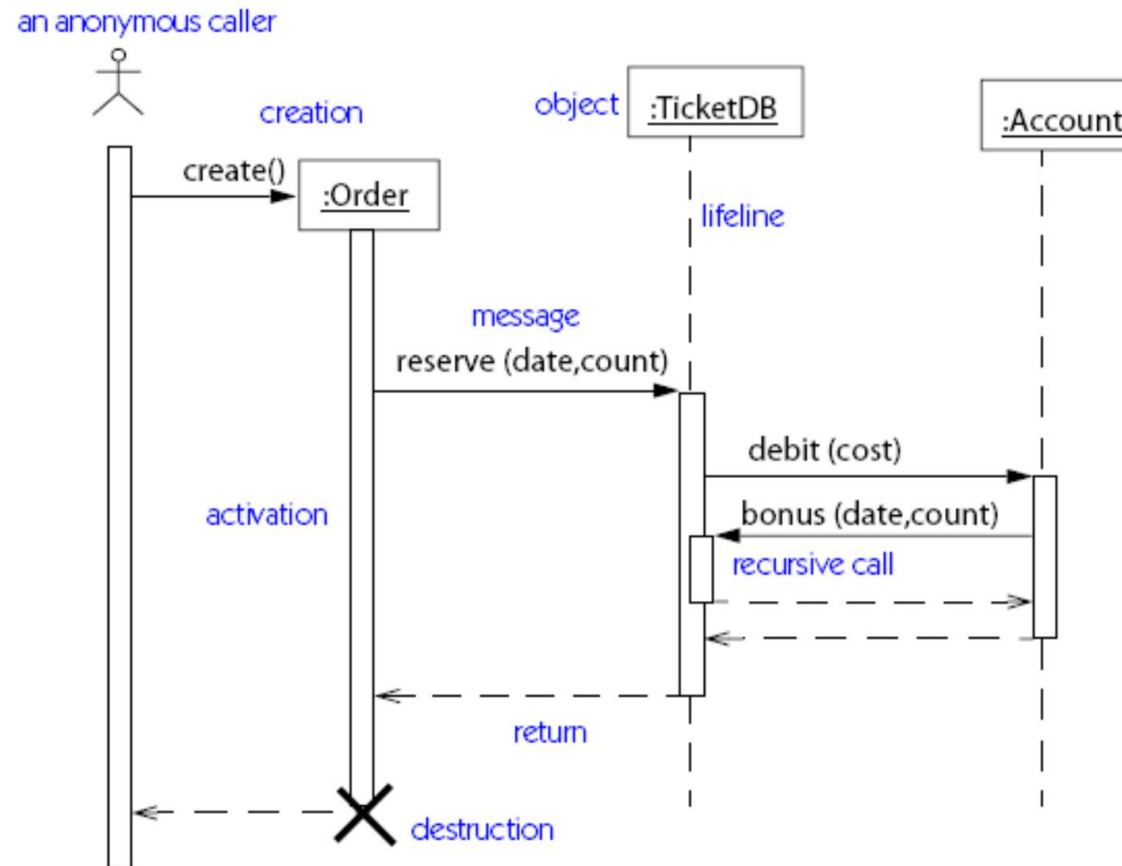
ARDIL-İŞLEM (“SEQUENCE”) DİYAGRAMI

- 👤 Nesneler arasındaki ilişkileri zaman sırasına göre düzenlenmiş olarak göstermeye yarar
- 👤 Bir etkileşimde yer alan nesneleri ve çağrılan mesajların sırasını gösterir
- 👤 İletişim diyagramı ile benzer bilgileri farklı biçimde gösterir
- 👤 İletişim diyagramında bulunan nesne ilişkileri ardıl-işlem diyagramında açıkça görülmez

“INTERACTION VIEW” “SEQUENCE DIAGRAM”: ÖRNEK



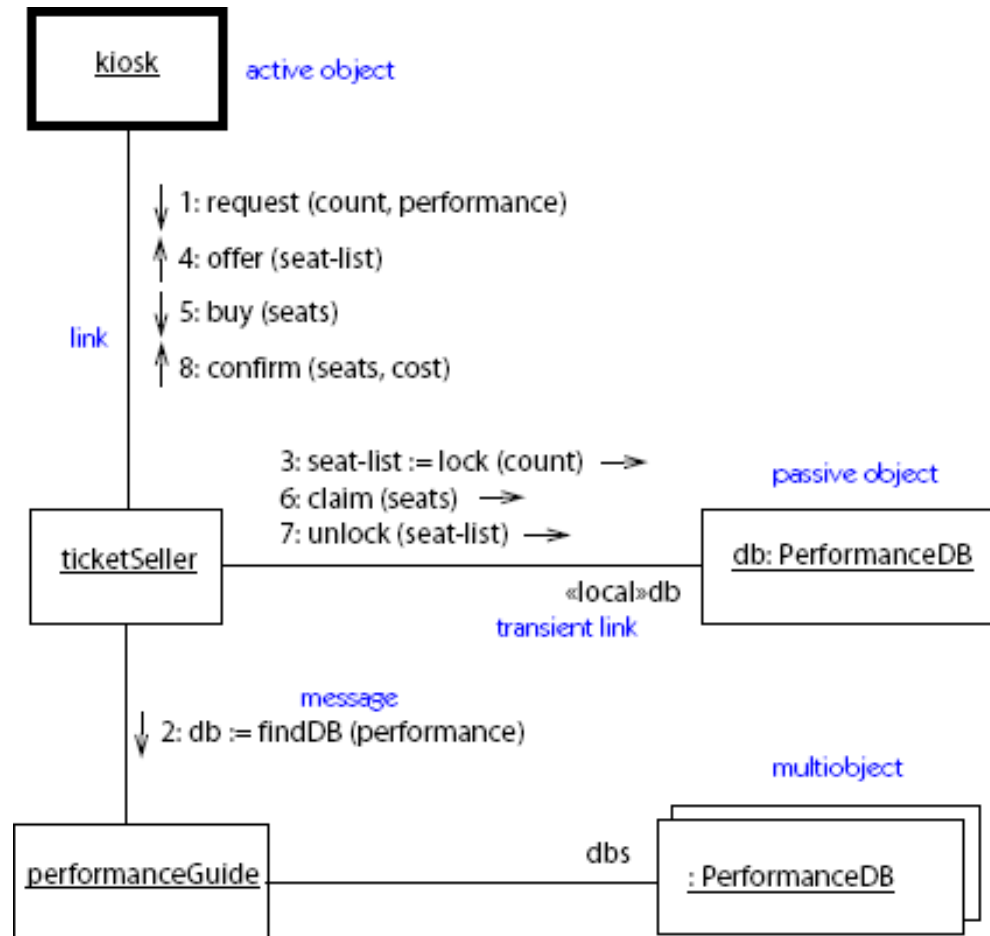
ACTIVATION”



İLETİŞİM (“COMMUNICATION”) DİYAGRAMI

- 👤 İletişim ve ardıl-işlem diyagramları nesneleri ve aralarındaki mesajlaşmayı dinamik olarak modellemeye yarar
- 👤 İletişim diyagramı, ardıl-işlem diyagramından farklı olarak, sınıflar arasındaki ilişkileri doğrulamada yardımcı olur
 - 👤 Fazla ya da eksik ilişkileri bulabilirsiniz

“INTERACTION VIEW” “COMMUNICATION DIAGRAM”: ÖRNEK



SINIF ÇALIŞMASI: KÜTÜPHANE DESTEK SİSTEMİ

- 🏠 Kütüphane işlemlerini desteklemek amacıyla bir yazılım sistemi oluşturulacaktır.
- 🏠 Sistem; kayıtlı müşterilere, yine kayıtlı kitap ve dergileri ödünç verecektir.
- 🏠 Kütüphane, yeni başlıklı kitap ve dergilerin satın almasını yapacaktır. Popüler başlıklar, birden çok kopya satın alınacaktır. Eski kitap ve dergiler, zaman aşımına uğradıklarında veya çok yıprandıklarında yok edilecektir.
- 🏠 Kütüphanede, müşterilerle iletişimi sağlayacak ve yaptığı işler sistem tarafından desteklenecek bir kütüphane görevlisi bulunacaktır.
- 🏠 Müşteri, kütüphanede o anda bulunmayan bir kitap veya dergiyi rezerve edebilecektir. Kitap veya dergi kütüphaneye geri döndürüldüğünde, rezervasyonu yapan müşteri haberdar edilecektir. Rezervasyon, müşteri kitap veya dergiyi ödünç aldığı anda veya müşterinin özel isteği üzerine iptal edilecektir.
- 🏠 Sistem; kitap ve dergi başlıklarının, kitap ve dergi kopyalarının, müşterilerin, ödünç işlemlerinin ve rezervasyonların kaydedilmesine, güncellenmesine ve silinmesine olanak sağlayacaktır.
- 🏠 Sistem tüm popüler bilgisayar ortamlarında (UNIX, Windows, OS/2, vb.) çalışacak ve modern bir kullanıcı ara yüzüne sahip olacaktır.
- 🏠 Sistem yeni işlevler eklemek suretiyle genişletilebilir olacaktır.

REFEREANSLAR

- Software Engineering (10th. Ed.); Ian Sommerville; 2015.
- Guide to Software Engineering Body of Knowledge (v3); 2014.
- Hacettepe Üniversitesi BBS-651, A. Tarhan, 2019.