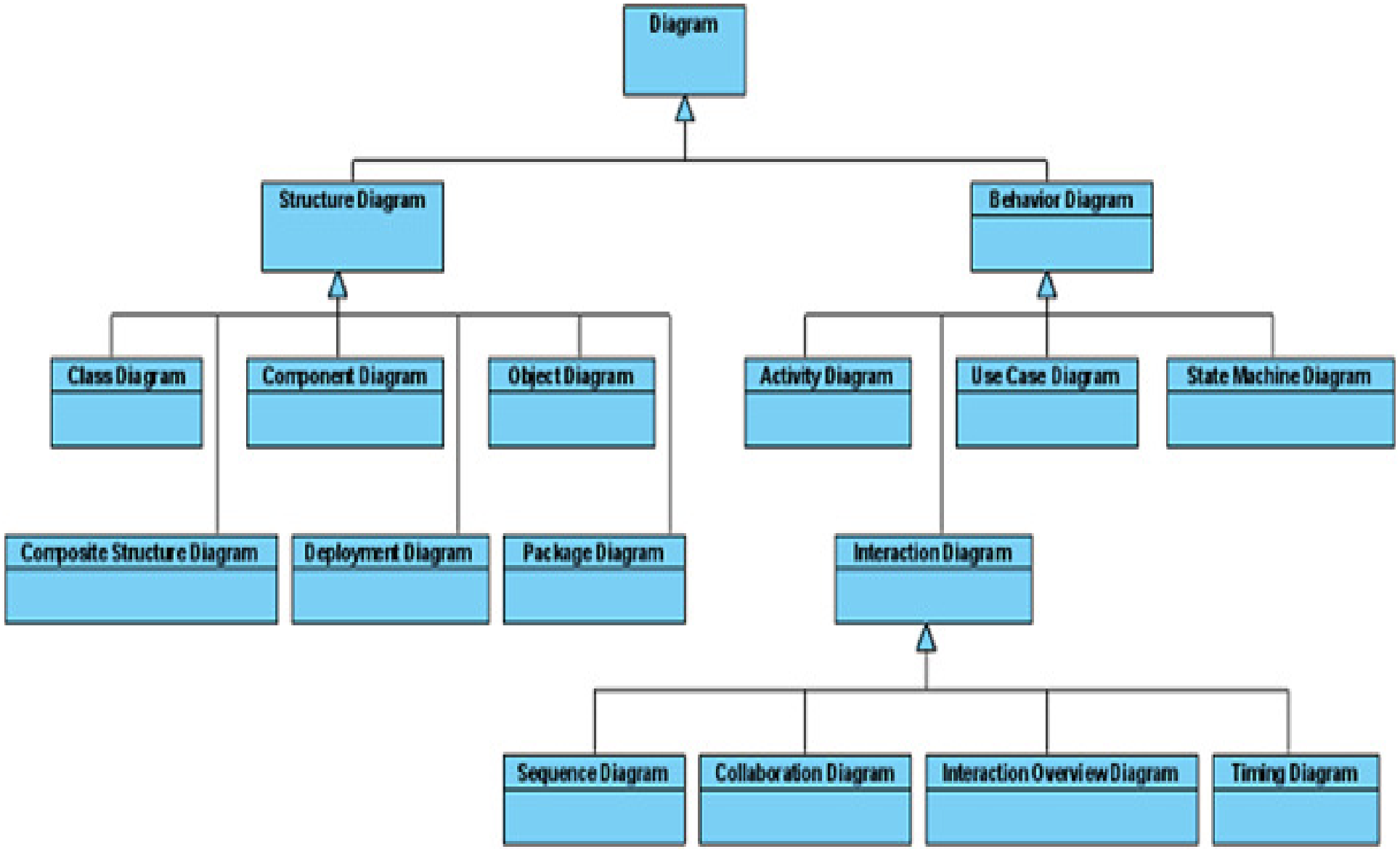
UML Diyagramlarının Sınıflandırması



Yapısal (Structural) Diyagramlar

- *Class Diagrams,*
- Object Diagrams,
- Package Diagrams,
- Component Diagrams
- Deployment Diagrams

Sınıf Diyagramı (Class Diagram)

- Sınıf diyagramları UML diyagramları içerisinde en yaygın kullanılan diyagramlardır ve geliştirilecek sistemin oluşturulmasında çok önemlidir.
- Sınıf diyagramları class, interface, association ve collaboration bildirimlerinden oluşur.
- ClasSınıf diyagramları bir sistemin nesneye yönelik tasarımının (object oriented view of a system) statik olarak tasarlanmasıdır.
  - Ürünün geliştirilmesi aşamasında tasarımı oluşturur.

Nesne Diyagramı (Object Diagram)

- Nesne diyagramları (Object diagrams) sınıf diyagramının bir örneğidir (instance of class diagram).
  - Bu diyagramlar problemin implementasyonunda gerçek dünya senaryolarını örnekler.
- Nesne diyagramları da aynı sınıf diyagramında olduğu gibi bir dizi nesne ve bu nesneler arasındaki ilişkilerden oluşur.

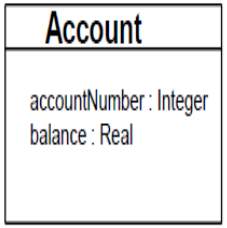
Nesne & Sınıf Tasarım Kavramları

- Nesneler gerçek dünyadaki şeyleri (things) simgeler.
  - Gerçek dünyanın anlaşılabilirliğini sağlar.
  - Problemin bilgisayar çözümünün temelini oluşturur.
- Bir nesne Object (object instance) tek bir «şeydir.»

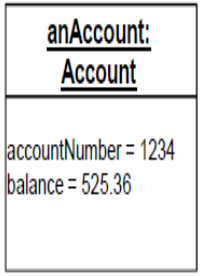
  birHesap, birCalışan gibi
- Bir nesne sınıfı (Object Class) aynı özelliklere sahip nesnelerin bir koleksiyonudur (collection of objects).
- Öznitelik (Attribute) o nesnenin sahip olduğu veri değeridir. (Data value)

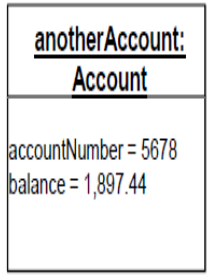
  «account number», «balance» gibi

Class with attributes

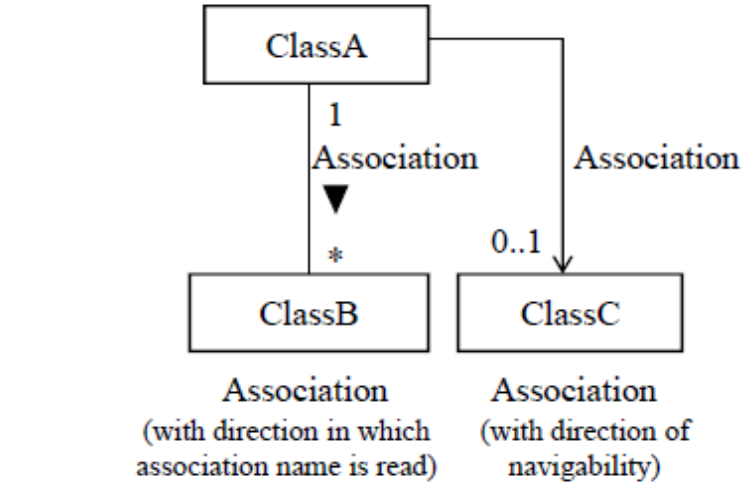


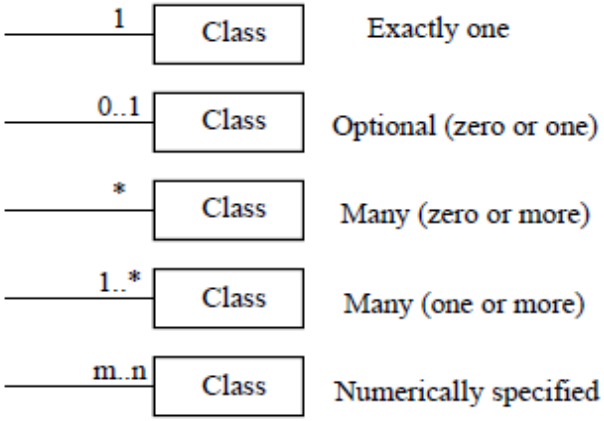
Objects with values

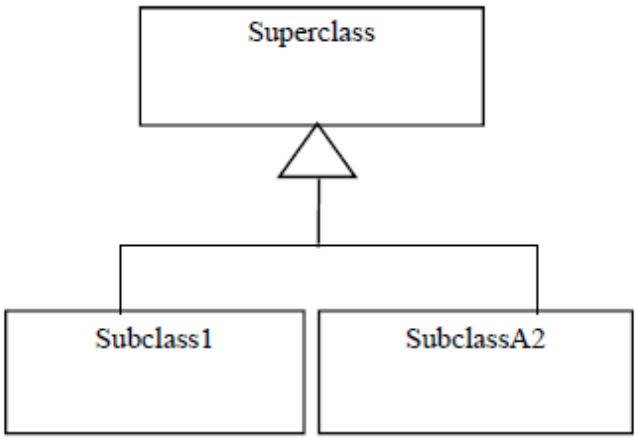




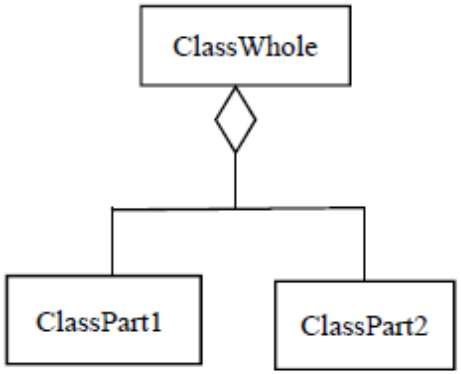
Sınıf Diyagramlarının UML Gösterimi



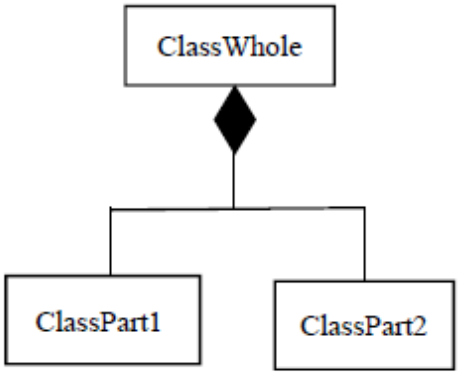




Generalization/specialization

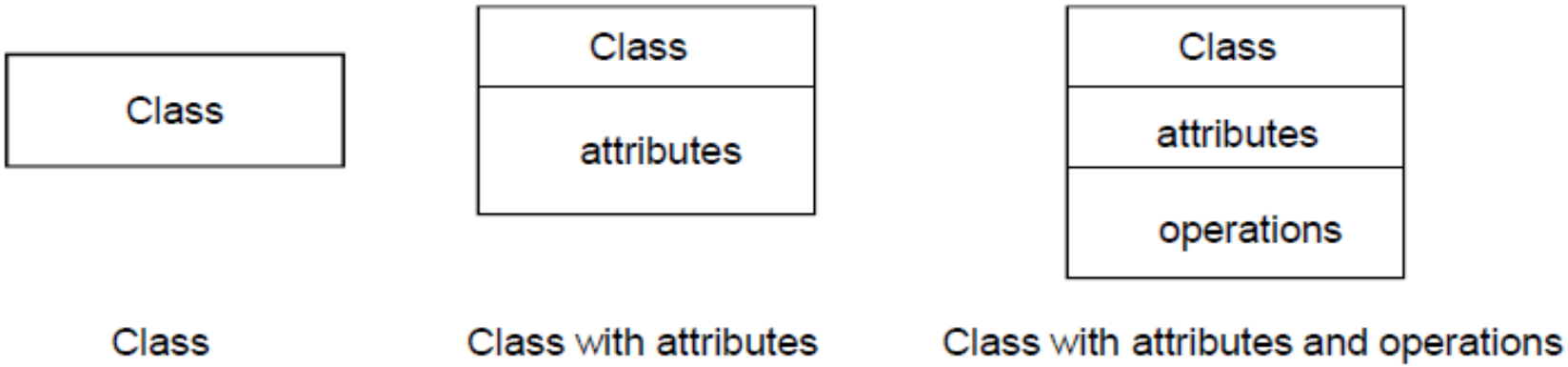


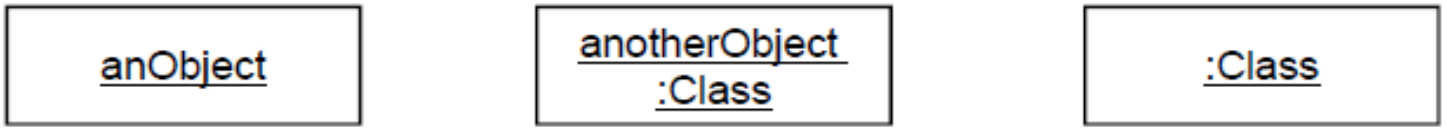
Aggregation hierarchy



Composition hierarchy

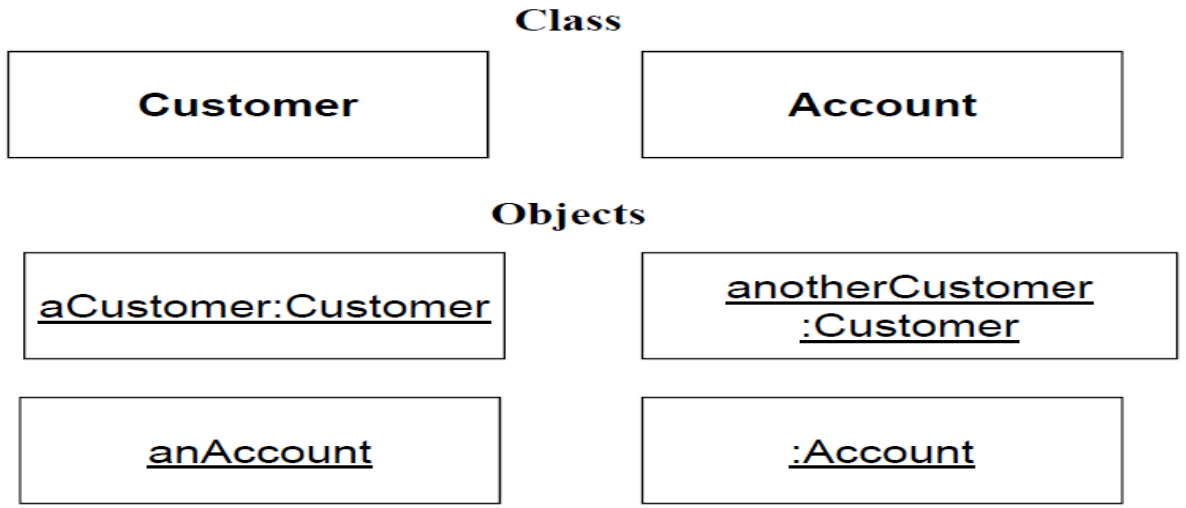
Sınıf ve Nesnelerin UML Gösterimi

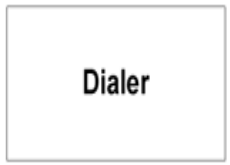




Objects

Sınıf (class) ve Nesne (Objects) Örnekleri






```
public class Dialer
{
}
```

Genel sınıf tanımı

Sınıfın öznitelik (değişkenleri) ve metotları (fonksiyonları) ile bildirimi

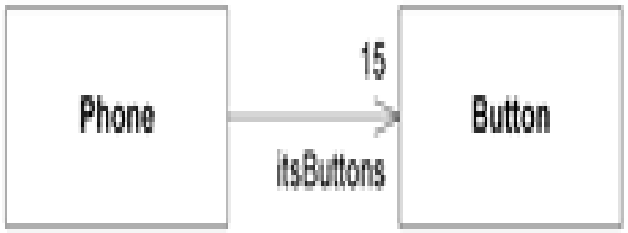

```
public class Dialer
{
  private Vector digits;
  int nDigits;
  public void digit(int n);
  protected boolean recordDigit(int n);
}
```

Sınıflar arası Birliktelikler (Associations)

- Sınıflar arasındaki yapısal ilişkiler tanımlanır (structural relationships)
- Sınıflar arası ilişkiler
  - Associations
  - Composition / Aggregation
  - Generalization / Specialization
- Analiz sırasında statik modelleme
  - Sistem «context» (içerik) sınıf diyagramları (System Context Class Diagram)
    - ✓Dışsal (harici) sınıflar ve sistem sınırlarını gösterir (Depict external classes and system boundary)
  - Varlık «Entity» sınıflarının statik modellemesi (Static Modeling of Entity classes)
    - ✓Verilerin depolandığı sınıflar

Association I

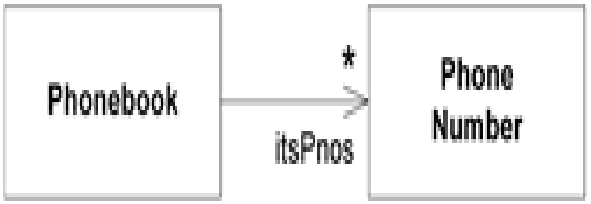
- Sınıflar arası birlikteliklerde (associations) çoğunlukla bir sınıfın örnek değişkenleri diğer nesneleri referans olarak alır.
- Phone ve Button sınıfları arasındaki birliktelikte okun yönü Phone sınıfının Button sınıfına bir referans gerçekleştirdiği görülmektedir.
  - Phone sınıfının bir değişeninin tipi diğer sınıftır (Button)
- Okun yanındaki isim sınıfın örnek değişkenidir (instance variable); sayı ise kaç tane referans tutulacağını ifade eder (15 elemanlı dizi anlamındadır).




```
public class Phone
{
  private Button itsButtons[15];
}
```

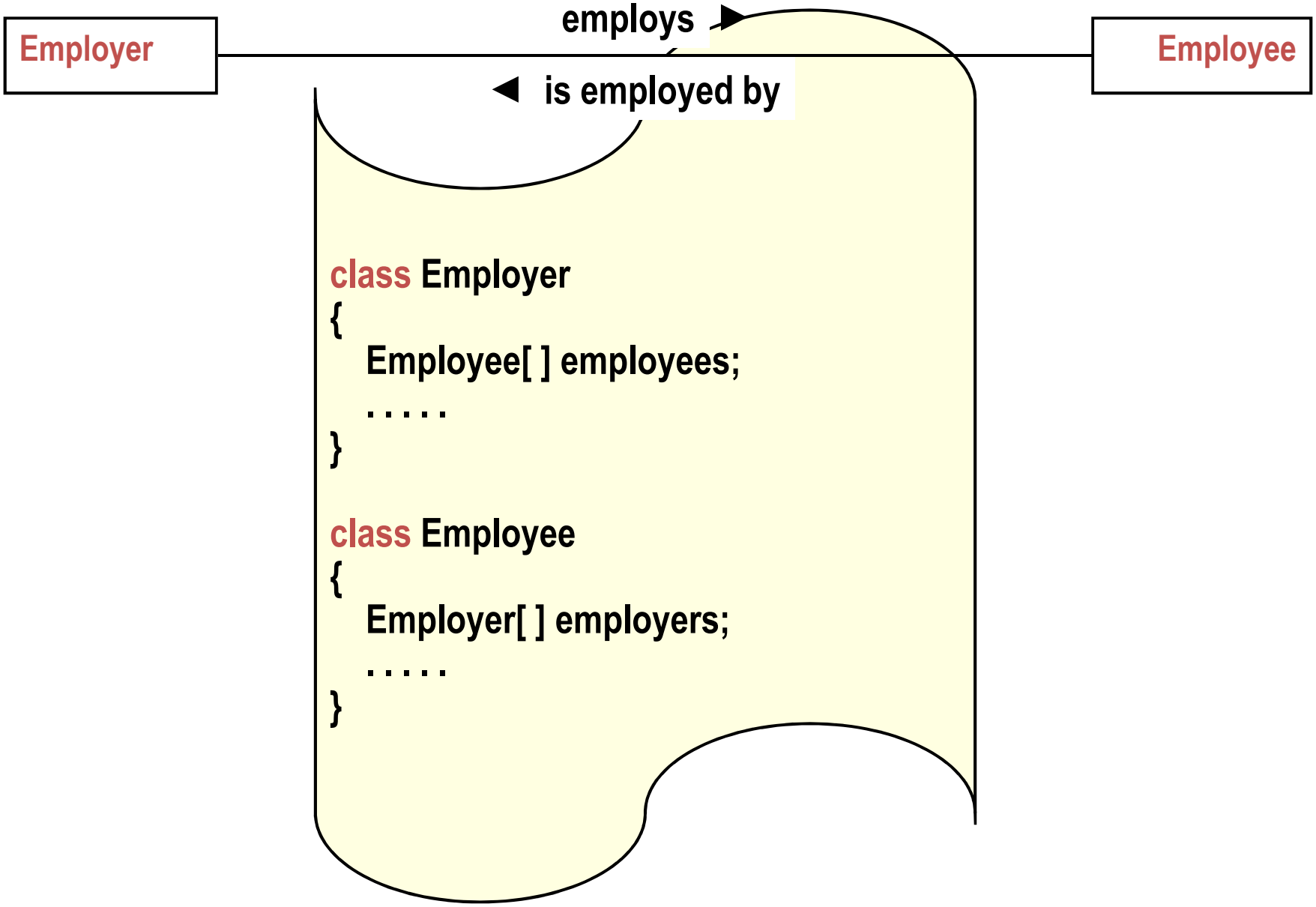
Association II

- “ Bir telefon rehberinde sıfır veya daha fazla sayıda (*) telefon numarası bulunur»
- Nesneye yönelik tasarımda bu ilişki sözlü olarak HASA (has a) ve ISA (is a) şekline ifade edilir.

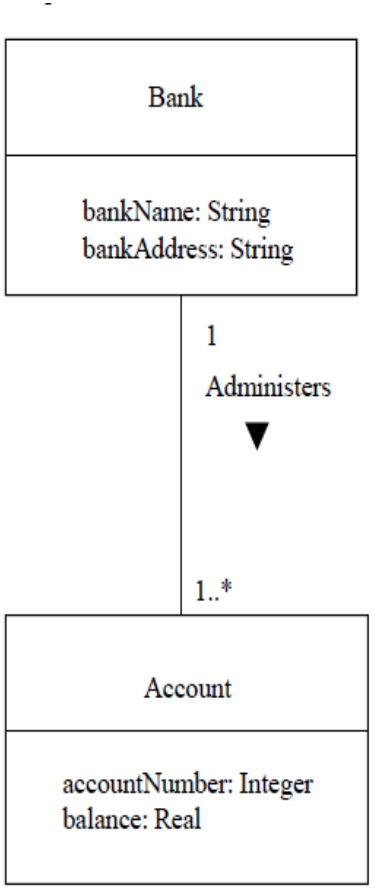



```
public class Phonebook
{
  private Vector itsPnos;
}
```

Association : UML Gösterimi



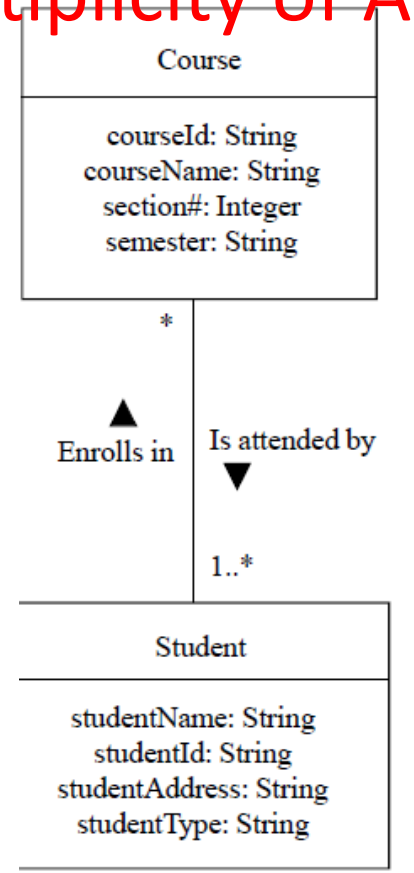
Birlikteliklerde Çokluklar
(Multiplicity of Associations)



One -to-many association (1..*)

Her bankada *1 veya daha fazla hesap bulunur.*

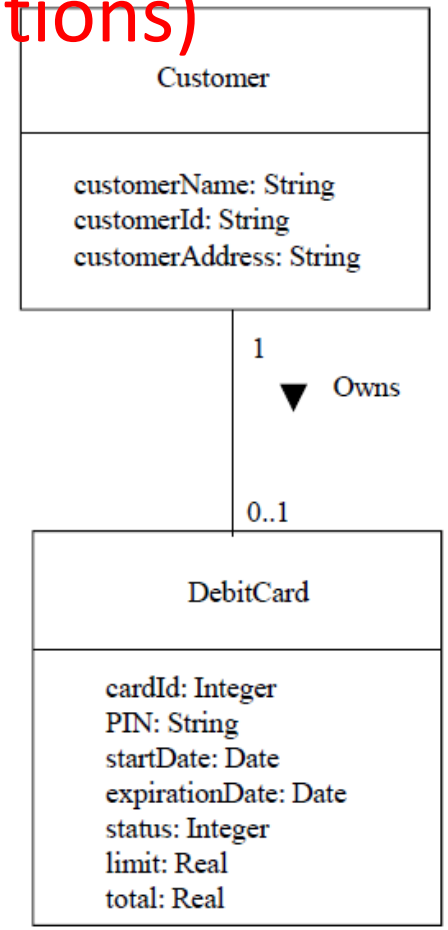
Her hesap *bir bankaya aittir.*



Many-to-Many association

Her ders *1 veya daha fazla sayıda (1..*)öğrenci tarafından alınır.*

Her öğrenci sıfır veya daha fazla (*) sayıda derse kayıt olur.

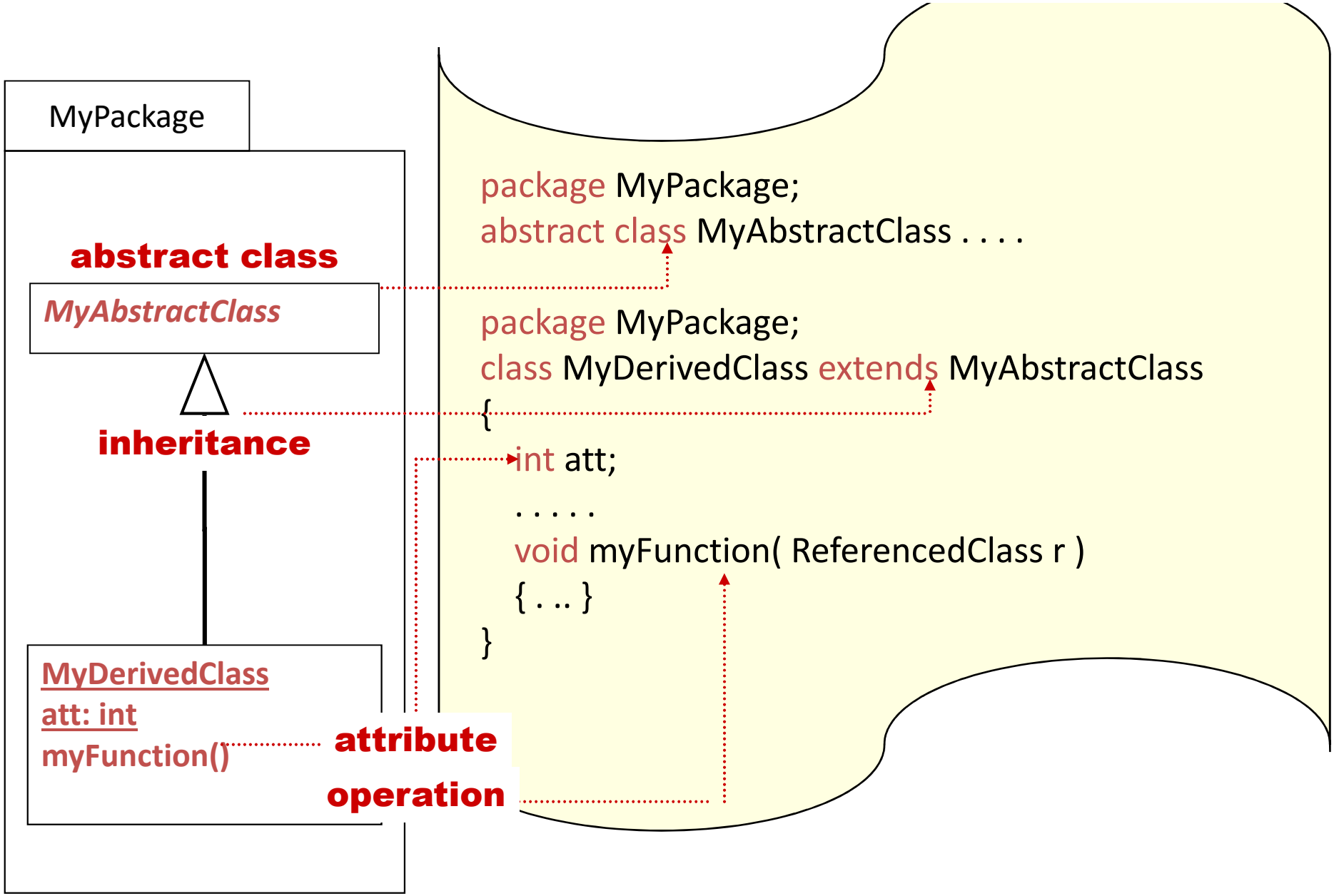


Optional association (0 ya da 1)

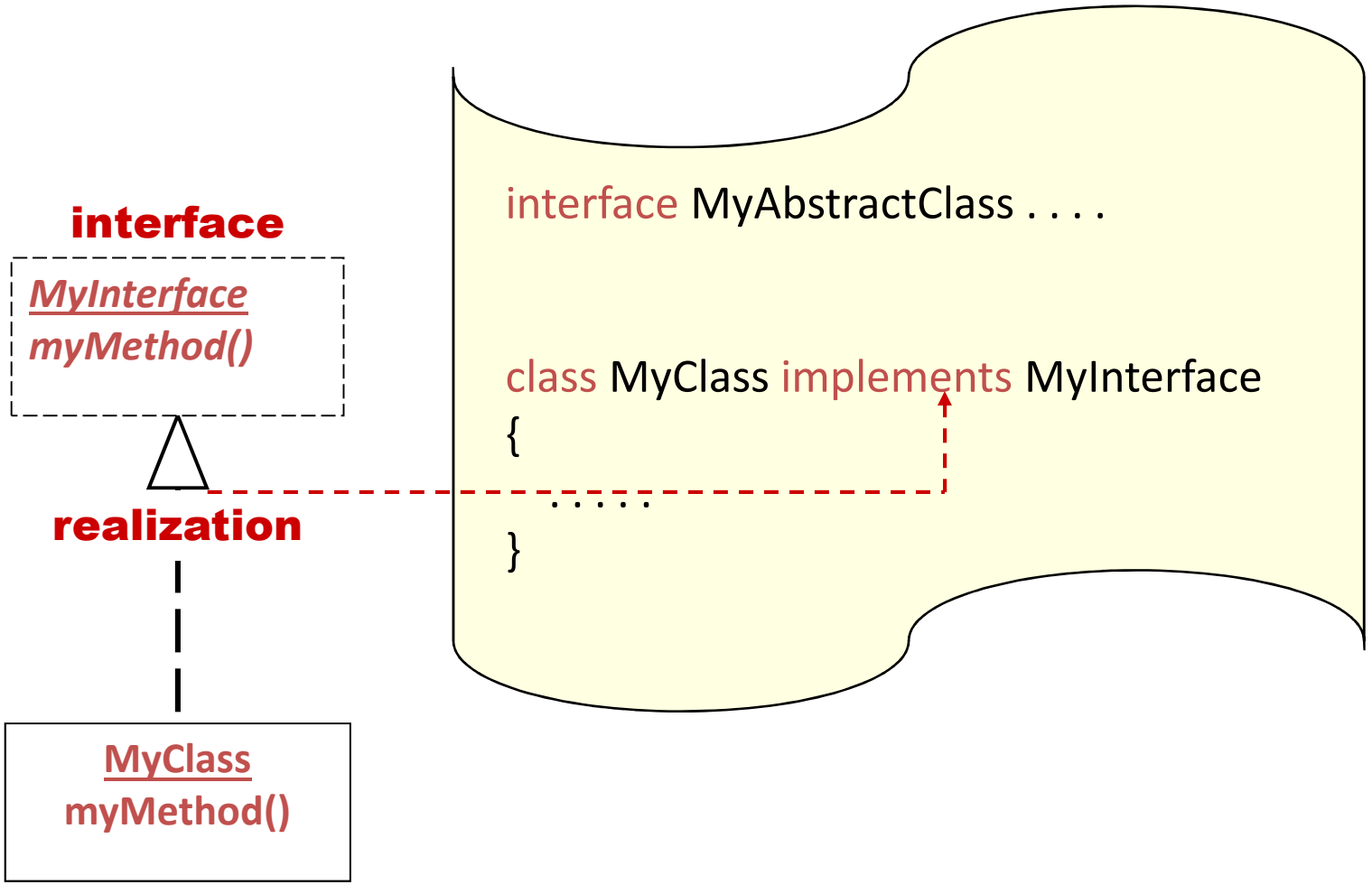
Her müşterinin *kartı yoktur ya da bir tane vardır (0..1)*

Her kart *bir müşteriye aittir.*

Inheritance : UML Notation



Interfaces: UML Notation

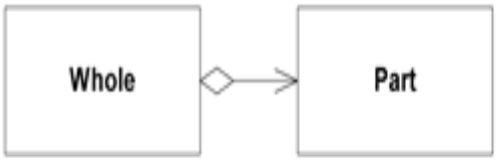


«Aggregation» Hiyerarşisi

- *Aggregation* "has a« birliktelik (association) ilişkisinin bir başka şeklidir.
- «Aggregation» ilişkisi «association» ilişkisinden daha güçlüdür.
- «Aggregation» «part-whole» ya da (part-of relationship) birlikteliğini tanımlar.

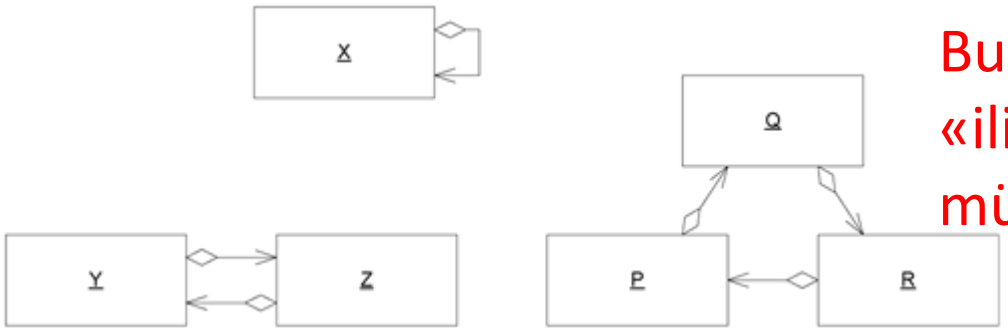
«Aggregation» Hiyerarşisi

- «Aggregation» "whole/part" bütünün parçası ilişkisi olarak betimlenir.




```
public class Whole
{
  private Part itsPart;
}
```

- Bir bütün(whole) kendisinin bir parçası olamaz (A whole cannot be its own part)
- Bu nedenle de bu ilişkinin örnekleri döngüsel olarak gerçekleşemez.
- Tek bir nesne kendisi ile ilişkili olamaz (aggregation ilişkisi).
- İki nesne karşılıklı olarak birbirleri ile «aggregation» birlikteliği oluşturmaz.
- Üç nesne bir «aggregation» ilişkisi döngüsü oluşturmaz.



Bu üç « aggregation «ilişkisinin gerçekleşmesi mümkün değildir.

Aggregation Hiyerarşisi

- Parça nesneleri (part objects) birbirlerinden bağımsız olarak oluşturulabilirler (created) ve bağımsız olarak silinebilirler (deleted independently)

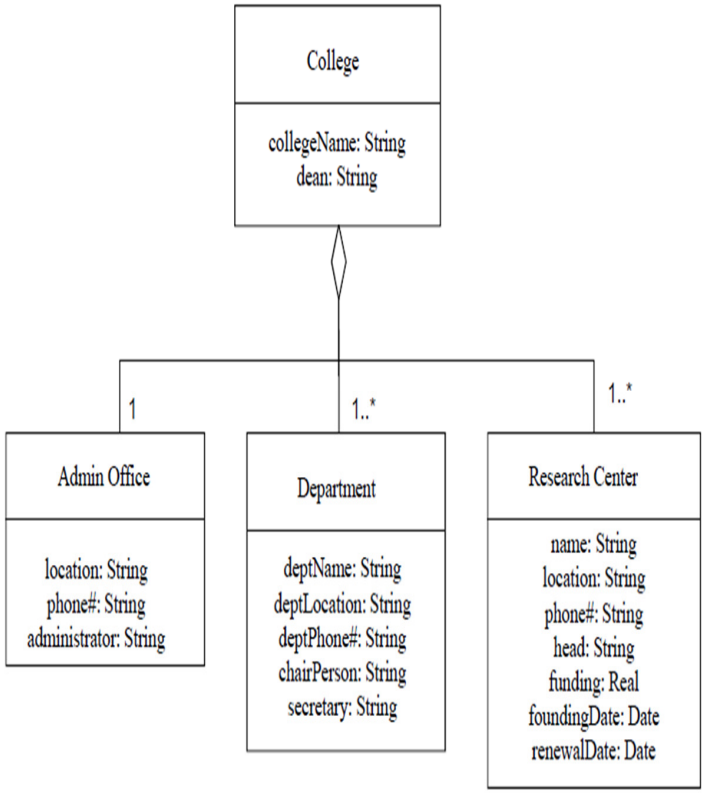
Aggregate class - College

Part classes –

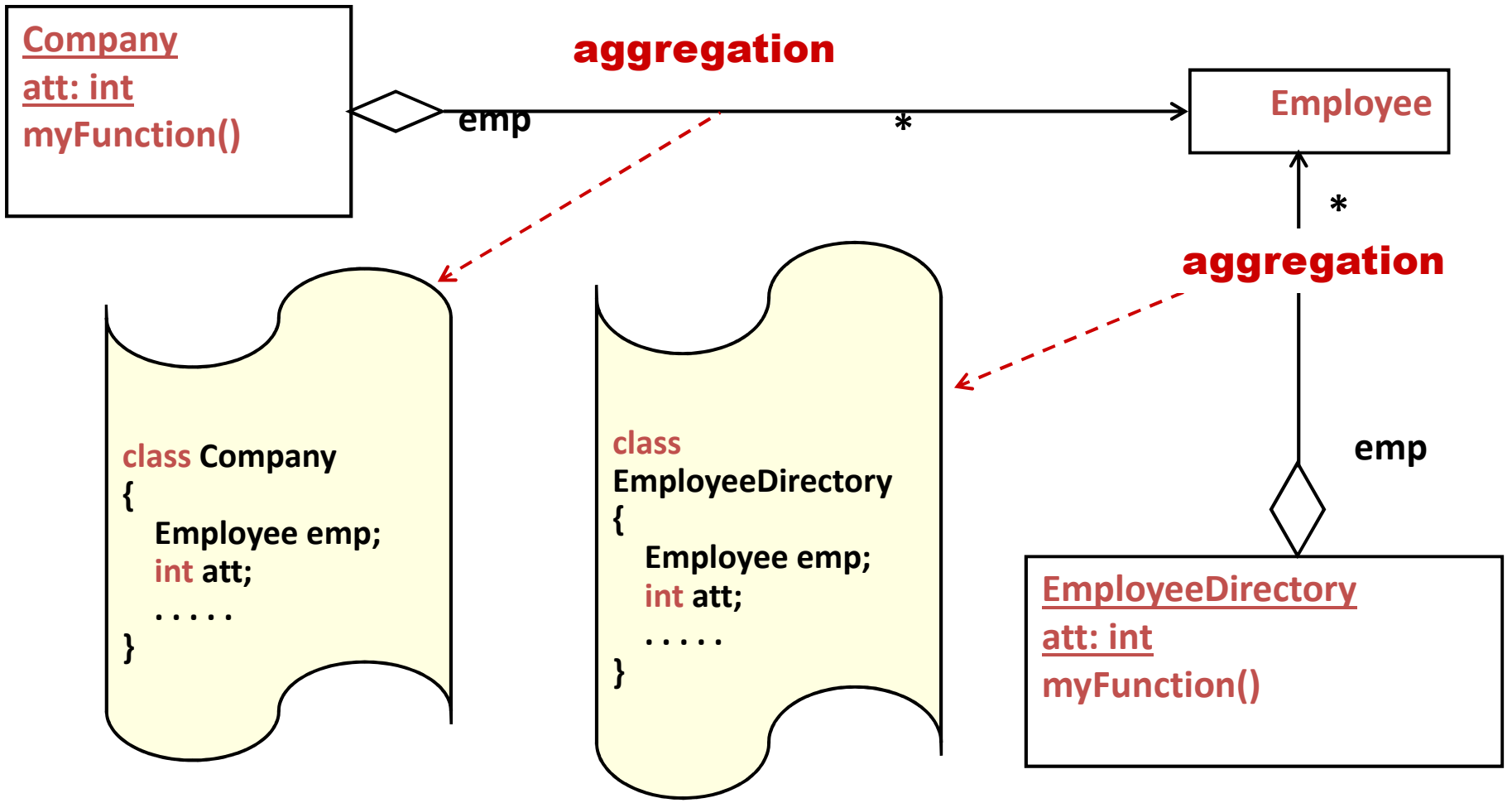
AdminOffice IS PART OF College,

Department IS PART OF College,

ResearchCenter IS PART OF College



Aggregation : UML Gösterimi



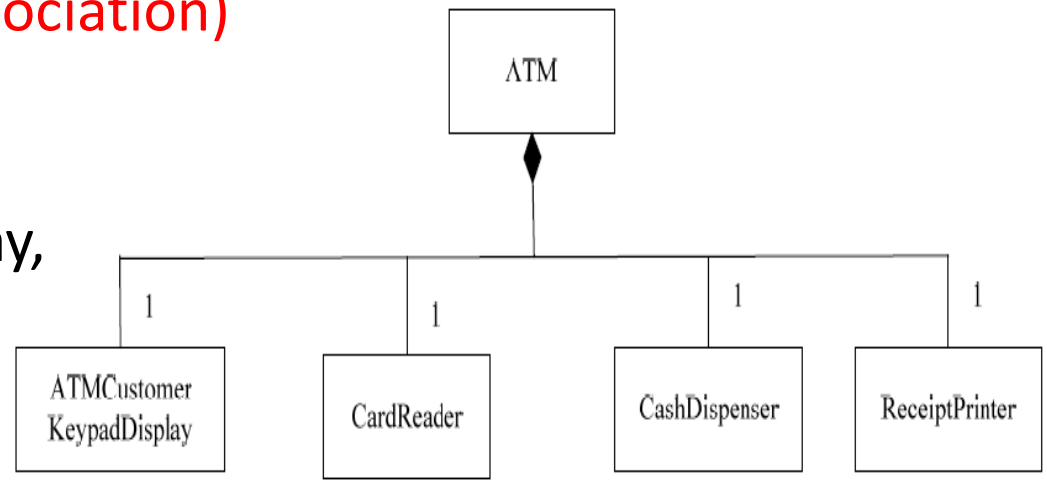
«Composition» Hiyerarşisi

- Bütün(whole) ve parça nesneleri (part objects) birlikte tanımlanırlar (created), işlemlerini gerçekleştirirler (live) ve birlikte yok olurlar (die).
- Genellikle fiziksel bir birliktelik oluştururlar (physical association)

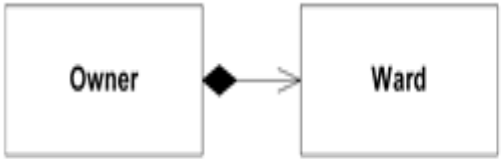
Composite class - ATM

Part classes

ATMCustomerKeypadDisplay,
CarReader,
CashDispenser,
ReceiptPrinter

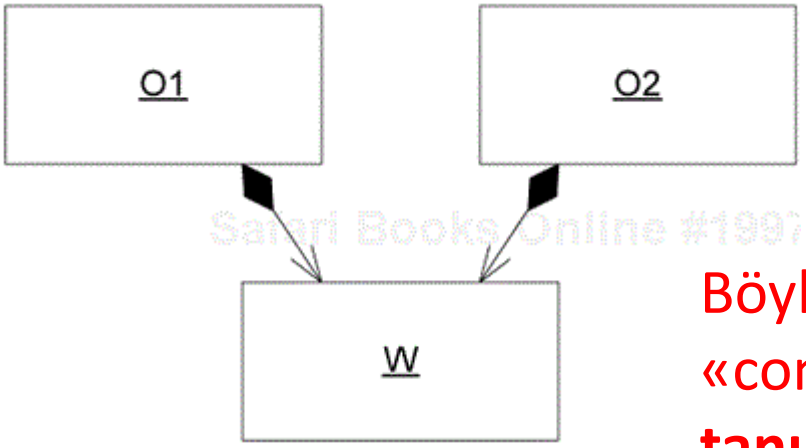


«Composition» Hiyerarşisi



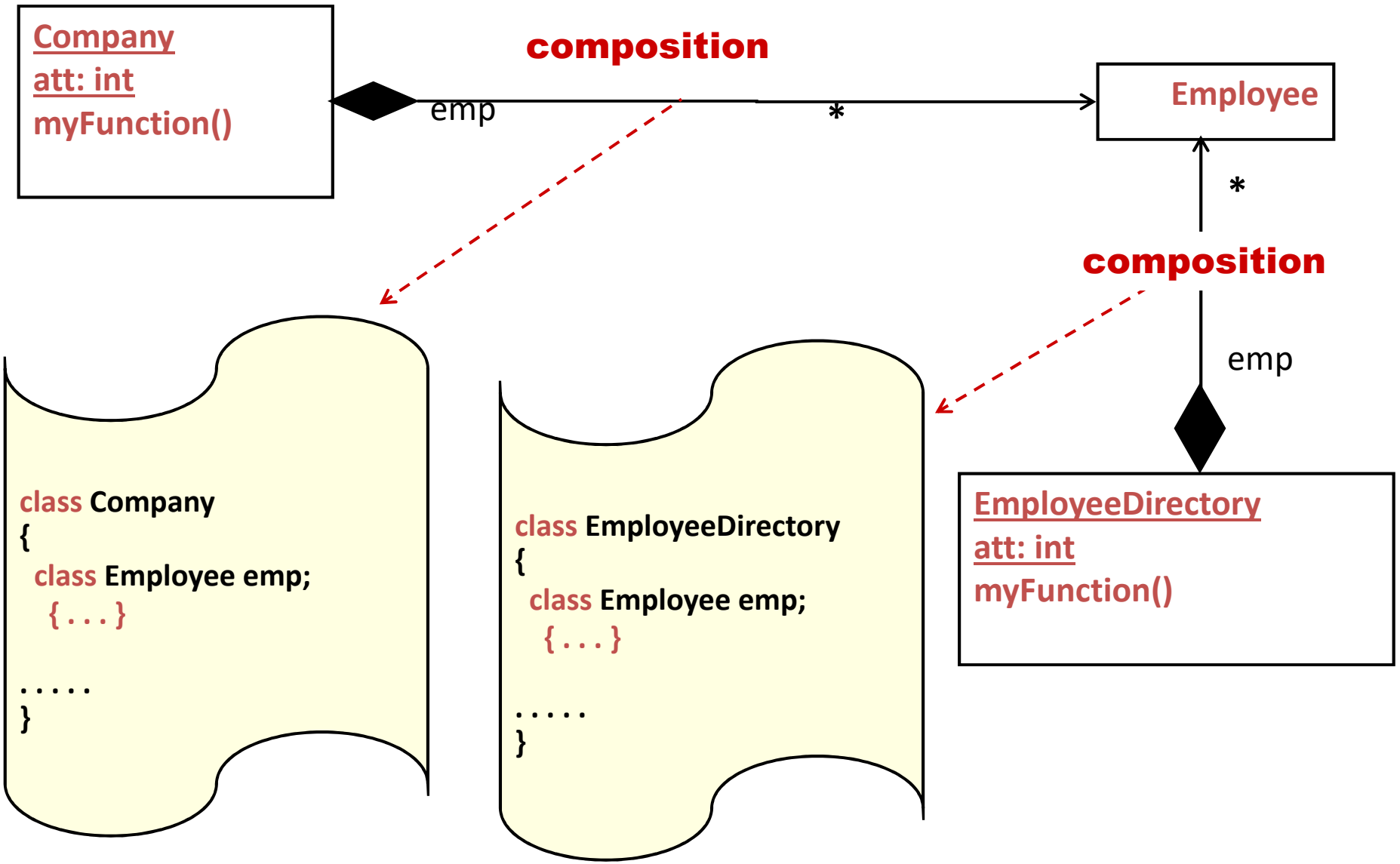

```
public class Owner
{
  private Ward itsWard;
}
```

« Composition» , «aggregation» ilişkisinin özel bir halidir.

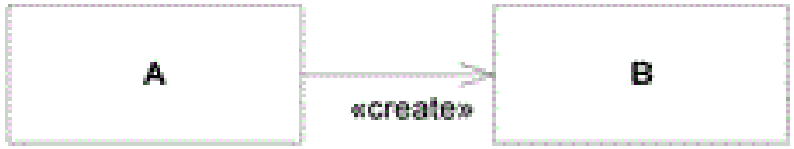


Böyle bir
«composition» ilişkisi
tanımlanamaz.

Composition: UML Gösterimi

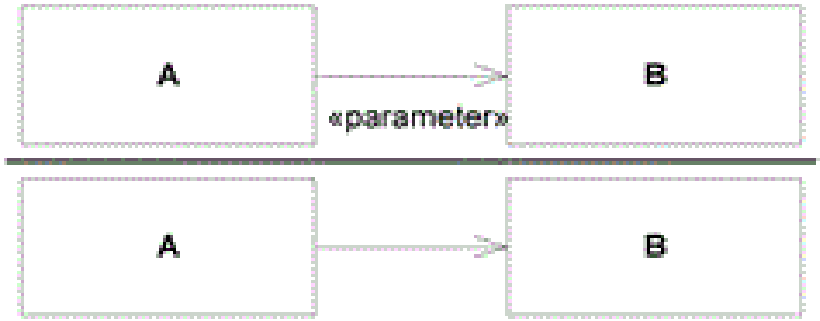


Association Stereotypes

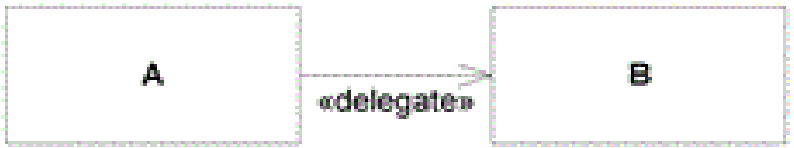



```
public class A {
  public B makeB() {
    return new B();
  }
}
```


```
public class A {
  public void f() {
    B b = new B();
    // use b
  }
}
```




```
public class A {
  public void f(B b) {
    // use b;
  }
}
```




```
public class A {
  private B itsB;
  public void f() {
    itsB.f();
  }
}
```

Generalization / Specialization Hierarchy

- Bazı sınıflar benzerdir; fakat birbirlerine denk değildir.
- Bu sınıfların bazı ortak öznitelikleri (attributes) vardır; fakat farklı öznitelikleri de mevcuttur.
- Ortak olan öznitelik (Common attributes) Generalized Class (superclass) şeklinde üst sınıfta soyut olarak tanımlanırlar.

  Account (Account number, Balance)
- Alt sınıf (subclass) ve üst sınıf (superclass) arasında IS A bağıntısı vardır.

  Savings Account IS A Account