Sınıf Diyagramları

Dr. Öğr. Üyesi Fatih ÖZYURT

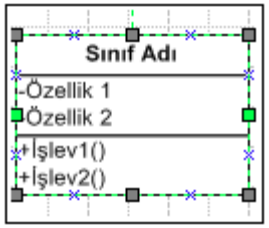
Fırat Üniversitesi Yazılım Mühendisliği Bölümü

Bölüm-6

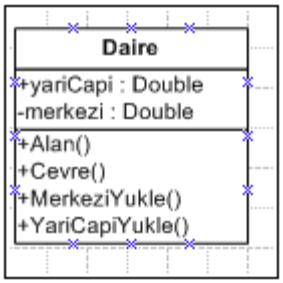
Amaç

- Class diyagramları, OOP(Nesne Tabanlı Programlama) temel alınarak tasarlanmıştır. Amaç yazılım içindeki sınıflar ve aralarındaki ilişkileri tanımlamaktır.

Sınıf Diyagramları



Bir dikdörtgeni 3 parçaya bölüyoruz. En üst bölüm sınıf adını, orta kısım özellik listesini (üye değişkenler) ve en son kısım, işlev listesini (üye fonksiyonlar) göstermektedir.



sınıf diyagramını incelerken +, - işaretleri görülmektedir.
Sınıf diyagramında yer alan nitelik ve method isimlerinin önünde aşağıda sıralanan işaretler kullanılabilir.
Private (-); Nitelik yada metota sınıf dışında erişim engellenmiştir.
Protected (#); Nitelik yada metota erişim sınırlandırılmıştır.
Public (+); Nitelik yada metot genel kullanıma açıktır.

Sınıf Diyagramları

Ancak sınıfın tek başına gösterimi bir şey ifade etmiyor. Bu sınıflar arasındaki bağlantıların gösterimi de önemli. UML'de ilişkilerin listesi şu şekilde:

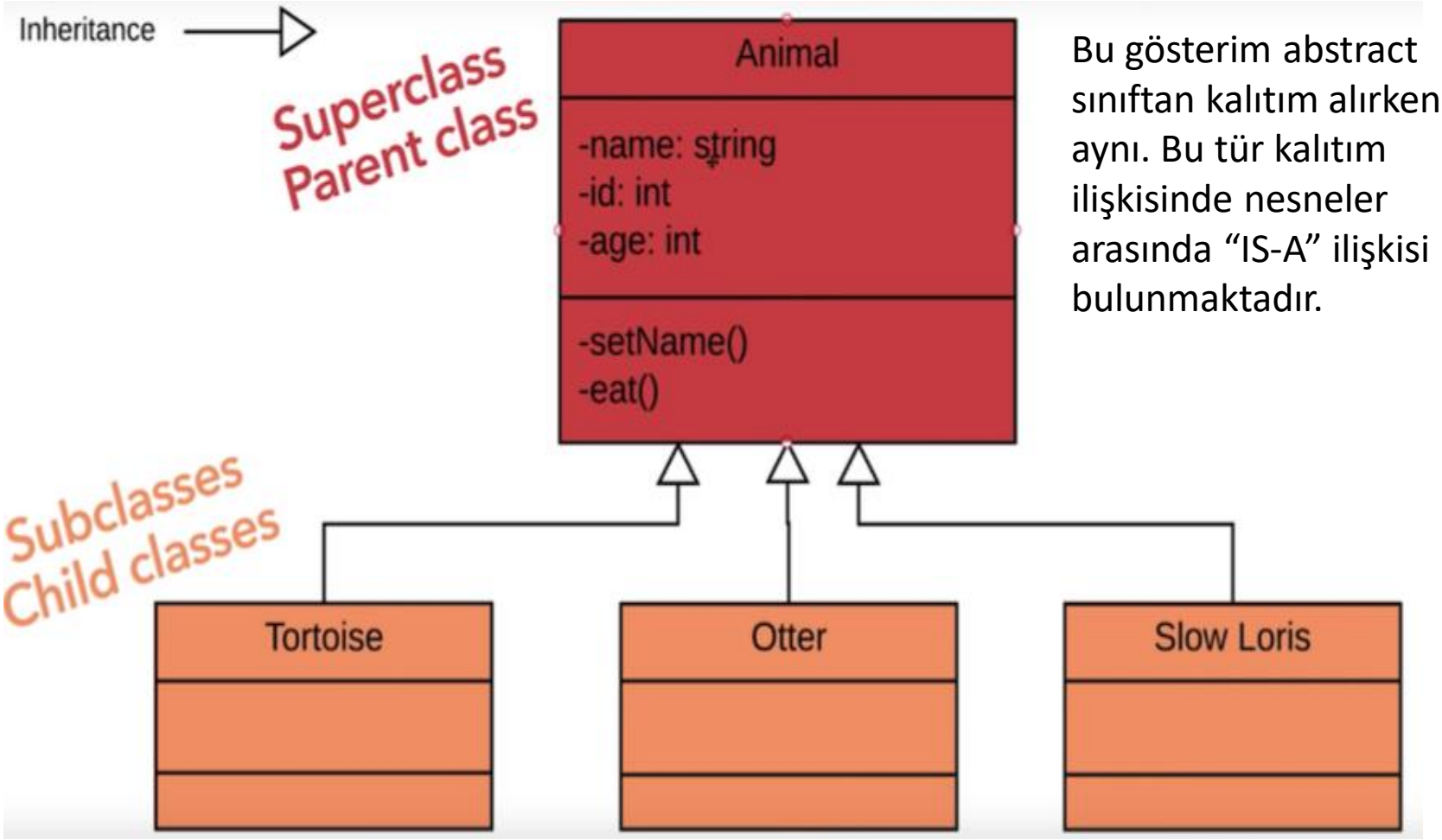
1- Generalization/Inheritance

2- Realization/Implementation

3- Association

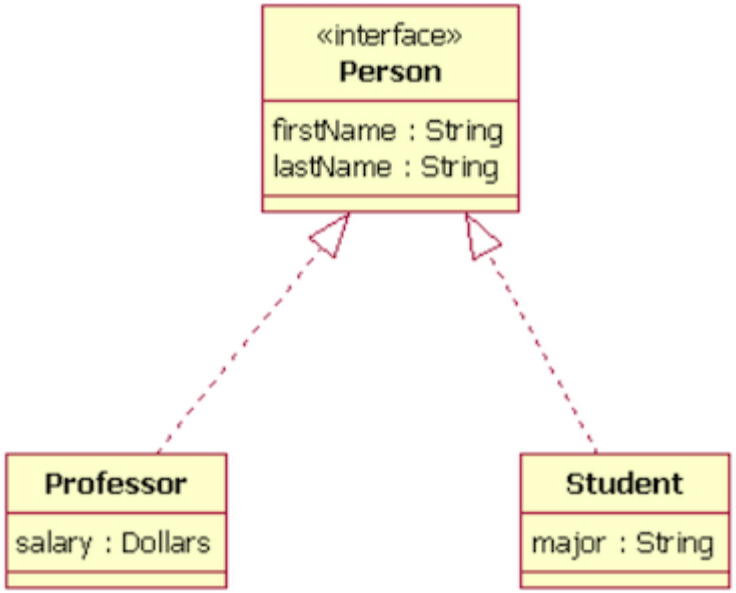
4- Dependency (Aggregation & Composition)

1- Generalization/Inheritance



Bu gösterim abstract sınıftan kalıtım alırken de aynı. Bu tür kalıtım ilişkisinde nesneler arasında “IS-A” ilişkisi bulunmaktadır.

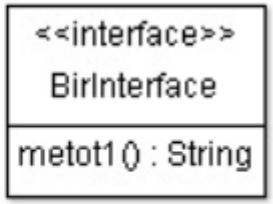
2- Realization/Implementation



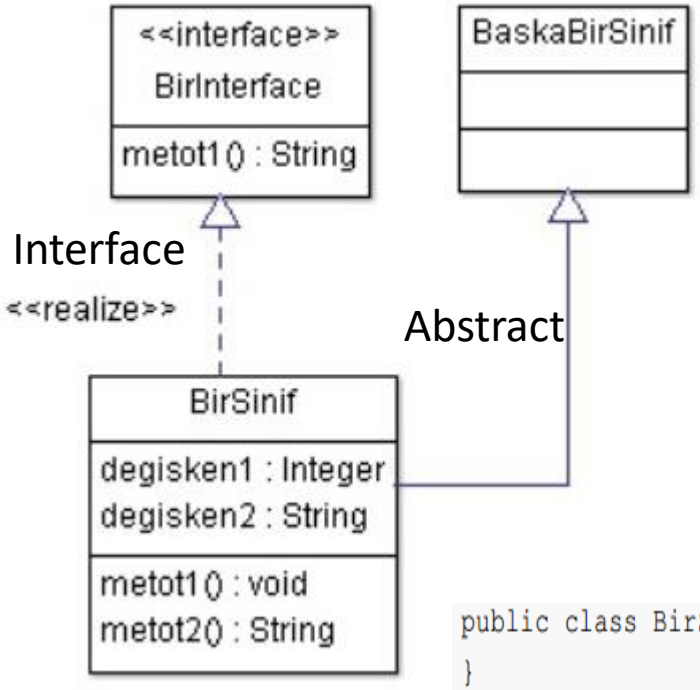
Arayüzler ile sınıflar arasındaki ilişkiyi modellemek için kullanılır. Dashed(kesikli) çizgi ile ifade edilir. Kalıtımdaki çizginin kesik kesik olan halidir.

Sınıf Diyagramları

```
public class BirSinif{

    private Integer degisken1;
    private String degisken2;

    public void metot1(){
    }

    public String metot2(){
    }
}

public interface BirInterface{

    public String metot1();
}
```



Sınıf Diyagramları



BirSinif sınıfı, BirInterface sınıfını implemente etmekte ve BaskaBirSinif sınıfını genişletmektedir.

```
public class BirSinif extends BaskaBirSinif implements BirInterface{
}
```

3- Association

Design patternlerde karşımıza çıkmaktadır.

Associationlar 4 çeşide ayrılmaktadır. Bunlar;

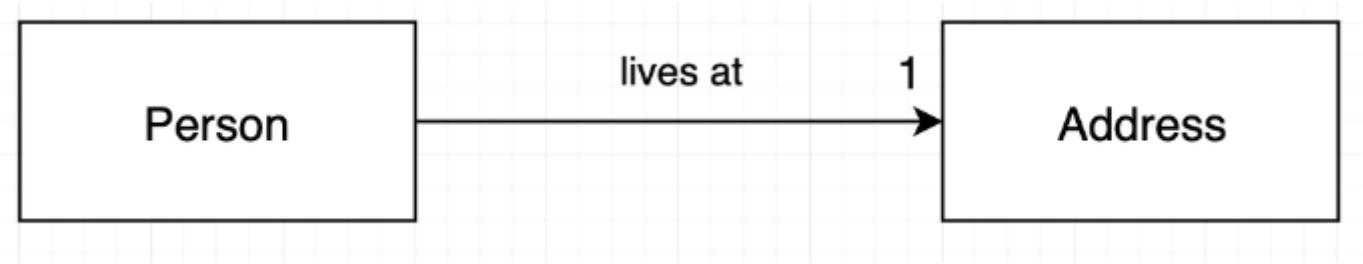
- Bi-directional(tek yönlü),
- Uni-directional(çift yönlü),
- Reflexive
- Aggregation&composition.

0	No instances (rare)
0..1	No instances, or one instance
1	Exactly one instance
1..1	Exactly one instance
0..*	Zero or more instances
*****	Zero or more instances
1..*	One or more instances

Has-a ilişkisi kurar

3- Association

Bi-directional Association (Tek Yönlü)



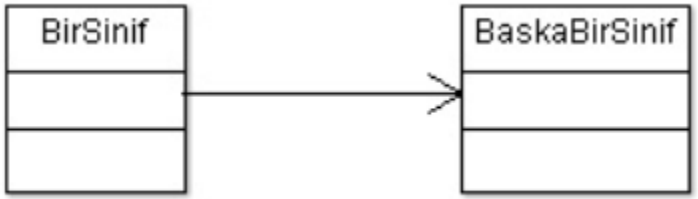
Burada 1 yerine 0..n şeklinde bir ifade de kullanılabilirdi. Üstteki ilişkide 1 yazdığı için, Person sınıfı içerisinde Address sınıfı tipinde bir attribute bulunacağını belirtiyor. Ancak Address sınıfında Person ile ilgili bir bilgi yer almıyor. Çünkü ilişki türü bi-directional(tek yönlü).

3- Association

İki sınıf arasındaki ilişki modellenmiştir.

Bu durumda BirSinif BaskaBirSinif sınıfını kullanmaktadır ve bunun tersi geçerli değildir.

Eğer düz çizgi ok ihtiva etmemiş olsaydı, her iki sınıfın karşılıklı birbirlerini kullandıkları söylenebilirdi.



3- Association

Uni-directional Association (Çift Yönlü)

Öğrenci sıfır ya da sonsuz tane kursa kayıt olmuş olabilir, bir kursa en az 1 yada sonsuz öğrenci kayıt olmalı.

3- Association

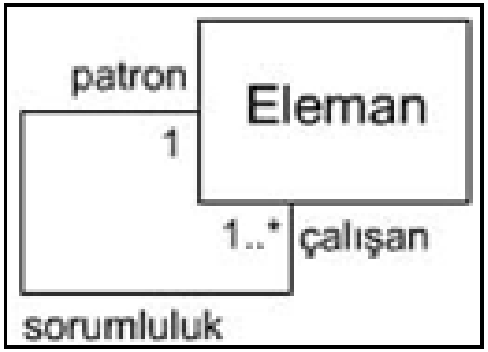
Sınıflar arasındaki ilişkiyi daha net tanımlamak için aşağıda yer alan yapılar kullanılabilir.


```
public class BirSinif{

    public List<BaskaBirSinif> list;
}
```

BirSinif sınıfı bir BaskaBirSinif nesnelerinden oluşan bir listeyi sınıf değişkeni olarak kullanmaktadır.
Hangi nesnenin kaç adet kullanıldığını UML diyagramında rakamlarla tanımlamak mümkündür. Yukarıdaki örnekte BirSinif sıfır ya da birçok BaskaBirSinif nesnesini kullanırken,
BaskaBirSinif hiçbir BirSinif nesnesini kullanmamaktadır.

3- Association

Reflexive (Dönüşlü)



Reflexive(dönüşlü) yani sınıfın kendisi ile yaptığı ilişkidir.

patron bir eleman olmasına rağmen kendisi gibi eleman olan birden çok çalışandan sorumludur diyebiliriz.

4- Dependency (Aggregation & Composition)

Aggregation

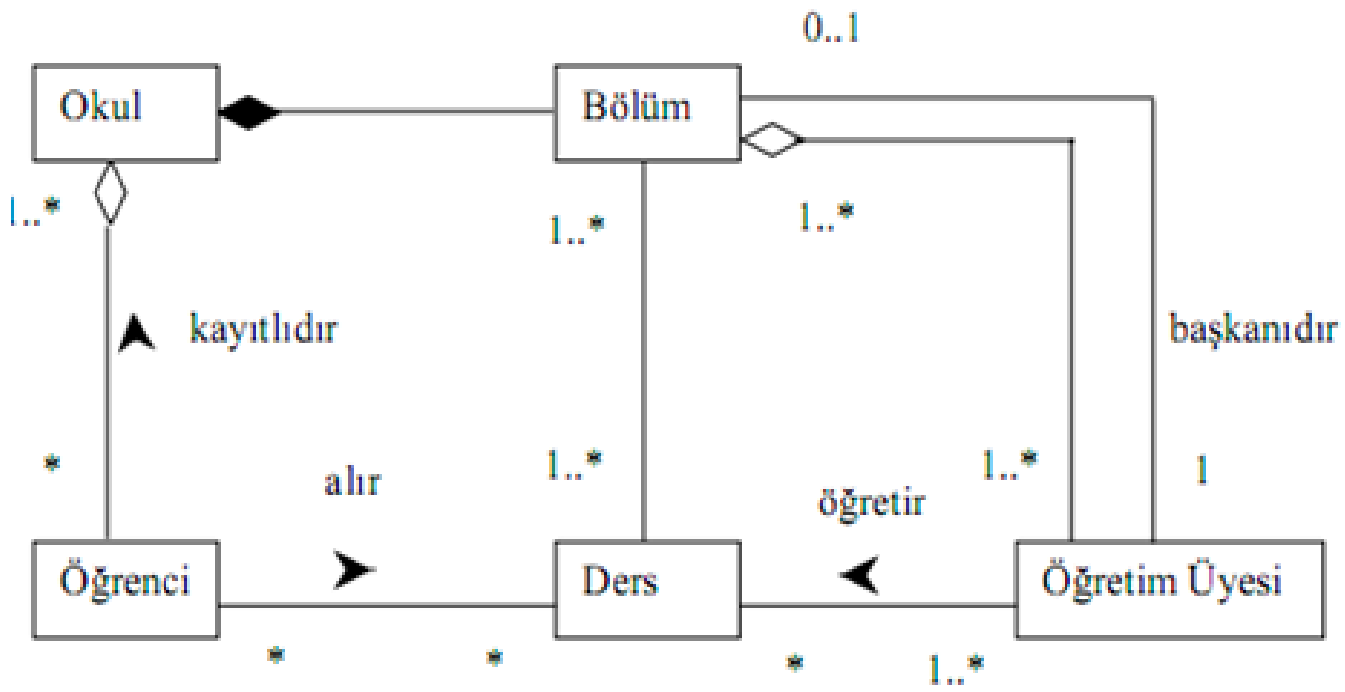
Tekerlek araba sınıfının bir parçasıdır. Ancak araba sınıfı yok olduğunda tekerlek yok olmak zorunda değildir. Aralarında zayıf bir parça ilişkisi vardır.

4- Dependency (Aggregation & Composition)

Composition

Kalp, insan sınıfının bir parçasıdır. İnsan sınıfı yok olduğunda kalpte yok olacaktır. İki sınıf arasında güçlü bir parça ilişkisi vardır. Composition ile agreegation arasındaki fark budur

4- Dependency (Aggregation & Composition)



Referanslar

1. Özcan acar Design pattern kitabı

2. https://medium.com/@tugrulbayrak

3. http://univera-ng.blogspot.com/2009/12/uml-ve-modelleme-bolum-4-class-snf.html

Sorularınız