Interface (Arayüz)

- Bir arayüz soyut sınıfların genelleştirilmesidir. Bir arayüz ile soyut sınıfın arasındaki en önemli fark şudur; arayüz içinde tanımlanmış tüm yöntemler soyut iken soyut sınıfı bazı yöntemleri bildirebilir ve gerçekleştirebilir. Bir arayüz içinde bildirilmiş yöntemler arayüzü gerçekleştiren (veya miras alan) sınıflar içinde gerçekleştirilmelidir.
- Bir başka ayrım da şudur; bir sınıfın sadece bir süper sınıfı olabilirken, arayüzün tüm yöntemlerini gerçekleştirdiği sürece, herhangi bir sayıda arayüzü gerçekleştirebilir. Bu yüzden arayüzler birkaç çeşit kalıtım sağlayabilirler diyebiliriz ki bu sınıflar için geçerli değildir.
- Java dilinde bir arayüz basitçe interface anahtar kelimesi kullanılarak bildirilebilir.

Interface (Arayüz)

new ile üretemezmişiz

```
interface Animal {

   public void eat();
   public void travel();
}

class MammalInt implements Animal{

   public void eat(){
      System.out.println("Mammal eats");
   }

   public void travel(){
      System.out.println("Mammal travels");
   }

   public int noOfLegs(){
      return 0;
   }
}

public class Ana {
    public static void main(String args[]){
      MammalInt m = new MammalInt();
      m.eat();
      m.travel();
   }
}
```

- Bir arayüz soyut sınıfların genelleştirilmesidir. Bir arayüz ile soyut sınıfın arasındaki en önemli fark şudur; arayüz içinde tanımlanmış tüm yöntemler soyut iken soyut sınıfı bazı yöntemleri bildirebilir ve gerçekleştirebilir. Bir arayüz içinde bildirilmiş yöntemler arayüzü gerçekleştiren (veya miras alan) sınıflar içinde gerçekleştirilmelidir.
- Bir başka ayrım da şudur; bir sınıfın sadece bir süper sınıfı olabilirken, arayüzün tüm yöntemlerini gerçekleştirdiği sürece, herhangi bir sayıda arayüzü gerçekleştirebilir. Bu yüzden arayüzler birkaç çeşit kalıtım sağlayabilirler diyebilirz ki bu sınıflar için geçerli değildir.
- Java dilinde bir arayüz basitçe interface anahtar kelimesi kullanılarak bildirilebilir.
- C# : kullanılırken bir inteface'den türetme işleminde implements kelimesi kullanılır.

Abstract

bunu da new ile üretemez mişiz

```
abstract class Employee
{
    private String name;
    private String address;
    private int number;

    public abstract double computePay();
}

class Salary extends Employee
{
    private double salary; // Annual salary

    @Override
    public double computePay()
    {
        return salary/52;
    }
}
```

- Java programlama dilinde bir sınıfın bazı yöntemleri abstract (soyut) olarak bildirilebilir ve eğer bir sınıf soyut yöntemlere sahipse o sınıf içinde soyut sınıftır denir. Eğer bir yöntem abstract olarak bildirilmişse bu yöntemi gerçekleştirmeye gerek yoktur, alt sınıflar bu yöntemi gerçekleştirmek zorundadır.
- Ayrıca bir sınıfı abstract yöntemleri olmasa da abstract olarak bildirmek mümkündür. Bu da anlamsız örnekler yaratılmasına engel olur.
- C#'a çok benzer. Tanımlamasında : yerine extends kelimesi kullanılır. Ayrıca abstract metot @Override kelimesi metot üstüne yazılır. Bu bakımdan da C#'tan farklıdır.

final

- C#'taki sealed yerine geçer.
- Eğer kalıtımcıların sınıfınızın yöntemlerini geçersizleştirmesini istemiyorsanız bu yöntemleri final anahtar kelimesini kullanarak belirtebilirsiniz:

```
class sinifAdi{
    final int aFinalMethod(){
    return 5;
  }
}
```

- Final yöntemler gibi final sınıflar da bildirebilirsiniz. Eğer bir sınıfı final olarak bildirirseniz, hiçbir sınıf ondan miras alamaz.

```
final class finalSinifAdi{

}
```