Programı Implementasyon Yerine Arayüze(Interface) Yapmak

Arayüz **interface** veya **süpertipe(supertype)** anlamına gelmektedir.

Programlama yaparken **polymorphism'i** yaygın bir şekilde kullanmak gerekir. Bu sayede gerçekte yaratılacak nesneler kod içerisine **kilitli kalmaz.**
Programlamayı **süpertipe** (genelde soyut sınıfa veya interface'e) yapmak demek, değişkenlerin tanımlanmış tipleri, **süpertip** olmalı anlamına gelir. Böylelikle, bu değişkenlere atanan nesneler süper tipin herhangi bir **somut** implementasyonu olabilir. Bunun anlamı, o sınıfları deklare eden bir sınıfın, gerçekte yaratılan nesneleri **bilmesine gerek yoktur.**

Örnek olarak **abstract** tanımlanmış **Animal** sınıfı ile bu sınıfı implement eden **somut** sınıflar **Dog** ve **Cat** olsun.

Programı implementasyona yaparsak şu şekilde kod yazarız:

```
1 Dog d=new Dog();
2 d.bark(); //havlamak demek
```

Programı interface'e/supertipe yapmış olsaydık şu şekilde kod yazardık:

```
1 Animal animal=new Dog();
2 animal.makeSound();
```

Nesne üretmeyi **hard-coding** yapmak yerine, **somut** implementasyonu runtime zamanında yapmak daha iyidir:

```
1 a=getAnimal();
2 a.makeSound();
```

Burada gerçekte hangi **Animal** alt sınıf nesnesi atanıyor bilmiyoruz. Tüm bildiğimiz şey **makeSound()** metoduna nasıl yanıt vereceğidir. Bu metod Dog sınıfında **bark()** metodunu, Cat sınıfında ise **meow()** metodunu çağıracaktır.

Sonuç olarak, kod yazarken yukarıda anlatılan prensibi uygularsak daha esnek uygulamalar geliştirebiliriz. Ayrıca kodu değiştirme ihtiyacı doğduğunda zorlanmadan değiştiririz.

Bu örnekle ilgili UML diagramı: