

**DERS :** ALGORİTMA VE PROGRAMLAMA 2

Doç. Dr. Ahmet ÇINAR

İsmail KIRCI

190260306

**KALITIM**

Nesne Yönelimli Programlama dillerinde kalıtım olgusu, bir sınıfta (class) tanımlanmış değişkenlerin ve/veya metotların (fonksiyon, procedure) yeniden tanımlanmasına gerek olmaksızın yeni bir sınıfa taşınabilmesidir. Bunun için yapılan iş, bir sınıftan bir alt-sınıf (subclass) türetmektir. Türetilen alt-sınıf, üst-sınıfta tanımlı olan bütün değişkenlere ve metotlara sahip olur. Bu özeliğe kalıtım özeliği (inheritance) diyoruz.

Programcı, yeni alt-sınıfları tanımlarken, üst-sınıftan (superclass) kalıtsal olarak geleceklere ek olarak, kendisine gerekli olan başka değişken ve metotları da tanımlayabilir.

Bu yolla, bir kez kurulmuş olan sınıfın tekrar tekrar kullanılması olanaklı olur. Böylece, programlar daha kısa olur, programın yazılma zamanı azalır ve gerektiğinde değiştirilmesi ve onarılması (debug) kolay olur.

Alt-sınıf türetme hiyerarşik bir yapıda olur. Bir alt-sınıfın türetildiği sınıf, o alt-sınıfın üst-sınıfıdır. Java’da bir alt-sınıfın ancak bir tane üst-sınıfı olabilir (C++ ‘dakinden farklı olduğuna dikkat ediniz). Ama bir sınıftan birden çok alt-sınıf türetilebilir.

Üst-sınıfa ata (parent), alt-sınıfa da oğul (child) denir.

**KOMPOZİSYON**

Java dilinde kompozisyon kullanımı inheritance(kalıtım)’a alternatiftir.  
Kalıtımın ana kuralı olan bir sınıf sadece bir sınıfı miras alabilir kuralı bazen bizi engelleyebilir.  
Örnek verecek olursak farklı sınıflarda işimize yarayan bir metod varsa, kullanabilmek için kompozisyon yaparız.

**SOYUT SINIF**

Ortak özellikleri olan nesneleri modellemek için Java dilinde kullanılan sınıftır. Soyut sınıfları oluşturmak için class ismi yerine “abstract class” kelimeleri kullanılır. Soyut sınıfı oluşturulan tasarımda benzer özellikleri olan sınıflar varsa kullanılabilir. Tekrar eden sınıflarda kod kalabalığı olmaması için soyut sınıf kullanımı daha kolaylık sağlamaktadır.

**INTERFACE**

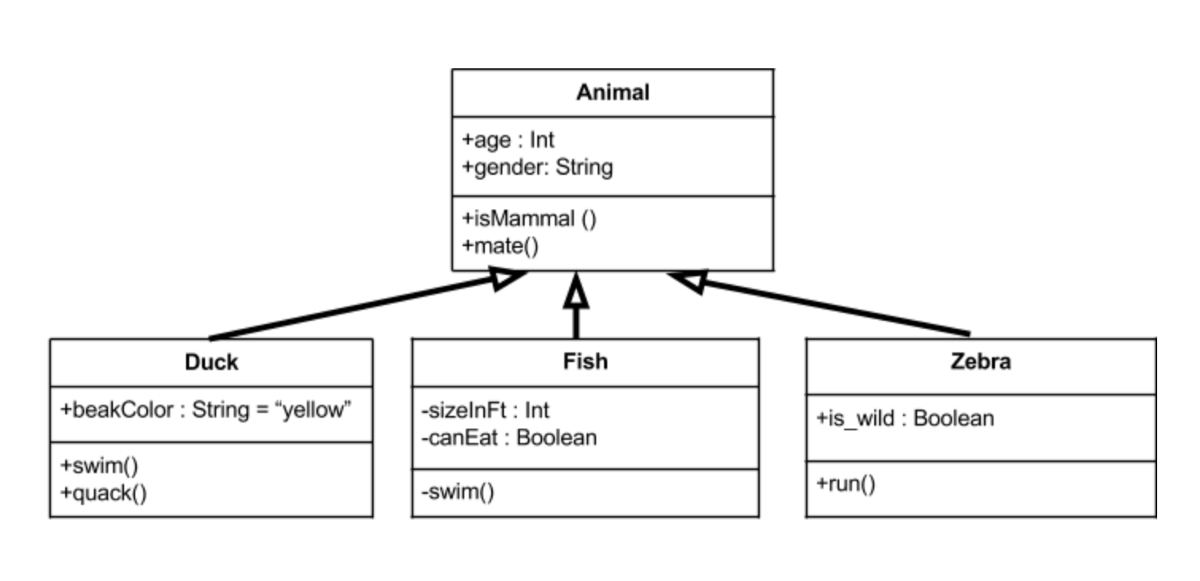
Interface Java’da class ile benzer şekilde tanımlanır. Sadece class yerine interface yazılır. Class’tan farklı olarak statik değişkenleri bulunmaktadır. Java desteklemez. Çoklu miras alabilmek için de interface çözümlerden biridir.

Interface; Constructor ve destructor kesinlikle barındırmaz. Arayüz içerisinde new() operatörü ile nensnne tanımlananmaz. Tanımlamanız durumunda “Cannot instantiate the type…” hatasıyla karşılaşırsınız.

Interface sınıflar dış dünyaya yapılacak hizmetleri söyler. Nasıl kullanılır kısmını alt sınıflar belirler. Kullannıcı hangi üst sınıfı olduğunu bilmek zorunda kalmaz. Böylece esneklik ve sistem kullanıcıları etkilemeden değişiklik yapmamızı sağlar. Servis sunucu ve sağlayıcı arasındaki bağımlılığı azaltır.

1: Çoklu miras (Multiple Inheritance) bir sınıfın iki veya daha fazla üst sınıftan miras almasını sağlar. Pratikte karışıklığa sebep olabildiği için C++ içinde kullanılabilen bu yapı c# ve Java’da kaldırılmış ve yerine arayüz (interface) mantığı getirilmiştir (Kaynak: Wiki/Çoklu\_Kalıtım)

**UML ÖRNEĞİ**

****