Sınıflar ve Nesne Kavramları

Sınıflar

Sınıflandırmak gerçek dünyada tasarladığımız neste tipleridir diyebiliriz. Ornek olarak insan dediğimiz zaman iki göz, iki kulak gibi belirli kalıba sahip bir tanımlama yapıyoruz. Bir başka örnekte araba dediğimizde renk, motor gücü gibi özellikler ile hareket et gibi aksiyonlar düşünebiliyoruz. Yani bunlar bizim bir nesne için yaptığımız sınıflandırmalardır.

Java' da da uygulamalarımızda benzer şekilde sınıflar tasarlayabilir veya var olan sınıfları kullanabiliriz. Zaten kitabın başında her şeyin bir sınıf içerisinde olması gerektiğinden bahsetmiş ve şimdiye kadar yaptığımız tüm örnekleri sınıflar içerisine almıştık.

Sınıfların kullanımı Java içerisinde aşağıdaki gibidir. Sınıf tanımlamak için class anahtar kelimesini kullanıyoruz.

```
class [Sınıf_Adı] {
      //Sınıf içeriği
}
```

Aşağıda Ogrenci adında bir sınıfımız bulunuyor.

```
class Ogrenci {
      //Sınıf içeriği
}
```

Dikkat : Yaratılan java dosyalarının sınıf isimleri ile aynı olması gerekiyor. Örnek olarak yukarıdaki Ogrenci sınıfının dosya ismi büyük küçük harf duyarlı olmak üzere **Ogrenci.java** dır.

Sınıflar tek başlarında bir işe yaramazlar içlerinde değer taşıyan değişkenler ve iş yapan metodlar bulunmalıdır. Bu anlamda aşağıdaki örnekteki gibi sınıflara ait özellikler ve metodlar olmalıdır.

```
class Ogrenci {
      String ad;
      String soyad;
      int yas;

      public void ortalamaHesaplama(){
            //Ortalama hesaplama işlemleri
      }
}
```

Peki bu sınıf ile bir şey yapabilirmiyiz ? Aslına bakarsak şu an için hayır. Eğer sınıf üzerinde bir iş yapılması gerekiyorsa bunun bellek üzerinde bir alanda oluyor olması gerekmektedir.

Dikkat : Sınıf isimlendirilmesi büyük harf ile başlar ve takip eden kelimelerde yine büyük harf ile başlatılır. Bu bir isimlendirme standartıdır ve uyulmaması derleme hatasına neden olmaz.

Nesne Kavramı

Sınıf ile tanımladığımız aslında bir kavramdır. Bunun gerçeğe dönüşmüş haline ise nesne denir. Örnek olarak Ogrenci adında tanımladığımız sınıftan birde çok nesne yaratabiliriz.

Java' da nesne yaratmak için new kelimesini kullanıyoruz. Aşağıdaki örnekte iki adet Ogrenci sınıfı kullanıyoruz.

```
Ogrenci o1 = new Ogrenci();
Ogrenci o2 = new Ogrenci();
```

Burada o1 ve o2 isminde ki adet değişken tanımı yapıyoruz ve ilk değerlerinin atanmasını sağlıyoruz. Değişken atamasının yani `Ogrenci o1` in aslında int i demekten bir farkı bulunmuyor. `new Ogrenci()`ise bizim bellek üzerinde heap alanda bir yer ayırmamızı sağlıyor. Yani bizim nesnemizi yaratan taraf `new Ogrenci()` oluyor.

Dikkat : Yolda yürürken bir otomobili gösterirsek bu gerçek bir nesneyi ifade eder. Çünkü gösterdiğimiz otomobil evren üzerinde var olan somut bir nesnedir. Java' da da nesneler bellekte gerçekten yer kapsayan alanlardır ancak sınıflar sadece kalıpları ifade eder.

Peki bir sınıfı nasıl oluştururuz. Bu iki şekilde örneklendirilebilir. Birincisinde aşağıdaki gibi sınıfın kendi içerisindeki bir main metodunda sınıfın bir instance yani bellek üzerinde bir varlığını oluşturabiliriz.

```
class Ogrenci {
	public void ortalamaHesaplama(){
		Ogrenci o1 = new Ogrenci();
	}
}
```

Bir diğer örnekte ise ilgili sınıfı dışarıda bir sınıf içerisinde kullanabiliriz.

```
class Ogrenci {

}

class Ornek2{
	public static void main(String[] args) {
		Ogrenci o = new Ogrenci();
	}
}
```

Nesne içerisindeki değerlere ulaşım

Peki oluşturduğumuz sınıfların içeriklerini nasıl atayabiliriz ? Bunun için aşağıdaki örnekteki gibi tanımladığımız değişken üzerinden ne içeriklerine erişebiliriz.

```
class Ogrenci {
	String ad;
	String soyad;
	int yas;

	public static void main(String[] args) {
		Ogrenci o1 = new Ogrenci();
		o1.ad = "Melih";
		o1.soyad = "Sakarya";
		o1.yas = 30;
	}
}
```

Bu örnekte Ogrenci nesnesinden main metodu içerisinde bir tane oluşturduk fakat aşağıdaki şekilde aynı nesneden birden fazla oluşturma şansınız da var.

```
class Ogrenci {
	String ad;
	String soyad;
	int yas;

	public static void main(String[] args) {
		Ogrenci o1 = new Ogrenci();
		o1.ad = "Melih";
		o1.soyad = "Sakarya";
		o1.yas = 30;

		Ogrenci o2 = new Ogrenci();
		o2.ad = "Ahmet";
		o2.soyad = "Dursun";
		o2.yas = 36;
	}
}
```

Değişkenler kısmında nesne tiplerinin temel tiplerden farklı olarak üzerlerinde değer taşımak yerine referans adreslerini gösterdiklerini belirtmiştik. Bu durumda gerek atama gerek karşılaştırma işlemlerinde durum biraz değişebiliyor.

Örnek olarak aşağıdaki kod çalıştırıldığı anda uygulama içerisinde iki adet değişken bellek üzerinde bir adet nesneyi işaret ederler. Bu durumda kod içerisinde o1 ve o2 demenin bir farkı yoktur da diyebiliriz.

```
class Ogrenci {

	public void ortalamaHesaplama(){
		Ogrenci o1 = new Ogrenci();
		Ogrenci o2 = o1;
	}
}
```

Bunu daha net görebilmek için değişken değerleri üzerinden kontrol yapalım. Sizce aşağıdaki kodun çıktısı ne olacaktır ?

```
class Ogrenci {
	String ad;

	public static void main(String[] args) {
		Ogrenci o1 = new Ogrenci();
		Ogrenci o2 = o1;

		o1.ad = "Melih";
		o2.ad = "Ahmet";

		System.out.println(o1.ad);
		System.out.println(o2.ad);
	}
}
```

Belki yanılıyor olabilirsiniz ? Aşağıda kodun çıktısı bulunuyor ?

```
Ahmet
Ahmet
```

Peki o1.ad "Melih" ken neden "Ahmet" yazdırdı ? Bunun nedeni o1 ve o2 değişkenlerinin aynı referans alanlarını göstermesidir. Yani o1.ad demem ile o2.ad demem arasında bir fark yoktu.

İç Sınıflar – Inner Class

Java içerisinde inner class yani sınıf içerisinde sınıf kavramı kullanılabilir. Peki buna ne zaman ihtiyaç duyarız ? Eğer tanımlayacağınız sınıf sadece onu saran sınıf içerisinde kullanılacaksa bunu iç sınıf haline getirebilirsiniz.

Örnek vermek gerekirse aşağıdaki kod içerisinde OgrenciBilgisi sadece Ogrenci sınıfı içerisinde kullanılacaktır. Bu durumda bunu ayrı bir sınıf yapmak yerine iç sınıf haline getirebiliriz.

```
public class Ogrenci {

	String ad;
	String soyad;
	OgrenciBilgi ogrenciBilgisi = new OgrenciBilgi();

	class OgrenciBilgi {
		int okulNo;
		String bolum;
	}

	public static void main(String[] args) {
		Ogrenci ogr = new Ogrenci();
		ogr.ad = "Melih";
		ogr.soyad = "Sakarya";
```

```
            ogr.ogrenciBilgisi.okulNo = 1234;
            ogr.ogrenciBilgisi.bolum = "Matematik";
      }

}
```

Dikkat : İç sınıflar derleme sonrasında ayrı bir class haline getirilir ve $ işareti ile ayrıştırılmış bir isimlendirmeye sahip olurlar. Örneğin yukarıdaki sınıf derlendiğinde aşağıdaki gibi iki class dosyası oluşturulur.

Ogrenci.class
Ogrenci$OgrenciBilgi.class