final kelimesi

final' kelimesinin kulanım yerlerine göre değişik özellikleri bulunmaktadır. Java' da final değişken, metod veya sınıflara verilebilir.

final değişkenler

Bir değişkeni final yaparsanız onun bellek alanını koruma altına almış olursunuz. Bu ilkel tipler ve nesne tipleri için farklı davranış gösterirler.

İlkel tipler için final direkt olarak stack alana atanmış olan değeri korur.

Örneğin aşağıdaki kod içerisinde yer alan ikinci x ataması hata verir. Çünkü tanımlama ile birlikte x değişkenini final yapıp atamasını yaptık. Bu durumda x' e yeni bir değer atayamayız. Ancak y değişkenine sadece bir kez atama yapıldığı için hata oluşmayacaktır.

```
class Ornek {

	public static void main(String[] args) {
		final int x = 4;
		int y;
		x = 6; // hata : x değişkeni ilk atamayla birlikte koruma altında
		y = 2; // y değerine ilk defa atama yapıldığından hata oluşmaz
	}
}
```

Dikkat : Eğer bir nesne değişkenini final olarak tanımlamışsak ilk değerini atamak zorundayız. Aşağıdaki örnekte x tanımı ilk değeri tanımlanmadığı için derleme anında hata verecektir.

```
class Ornek {

	final int x ; //atama yapılmadığından hata oluşur

	public static void main(String[] args) {

	}
}
```

Peki ilkel tiplerden farklı olarak bir nesneyi final yaparsak ne olur ? Nesnelerin final olması referans adreslerinin değiştirilmeyeceği anlamına gelir. Bunun anlamı o nesneyi new ile veya başka bir değere eşitleyerek değiştiremeyiz demektir.

Aşağıdaki kod içerisinde sizce hata oluşacakmıdır ?

```
class Ornek {
```

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

		ogr.ad = "Ahmet"; // Bu satırda hata oluşur mu ?
	}
}
```

Yukarıdaki kod herhangi bir hata vermez. Peki ogr nesnesi final olmasına karşın neden hata vermiyor ? Nedeni basit ogr nin referans adresinde herhangi bir değişiklik yapmadık. Sadece bağlı olduğu değişkeni değiştirdik.

Ancak aşağıdaki kodu incelersek ogr nin tekrar oluşturulması bir hataya neden olur. Bunun nedeni referans alanının değiştirilmeye çalışılmasıdır.

```
class Ornek {

	public static void main(String[] args) {
		final Ogrenci ogr = new Ogrenci();

		ogr = new Ogrenci();

	}
}
```

Not : Herhangi bir metod parametresini final yaparsak bu sadece o metod içerisinde o parametrenin değerini değiştiremeyeceğimiz anlamına gelmektedir.Aşağıdaki örnekte metod içerisindeki atama hata verirken sonraki atamada hata alınmayacaktır.

```
class Ornek {

	static void ornekMetod(final int sayi) {
		sayi = 8; // metod içerisinde değişiklik yapılamaz
	}

	public static void main(String[] args) {
		int x = 4;
		ornekMetod(x);
		x = 6; // değişken final değil istenildiği gibi değiştirilebilir.

	}
}
```

final metodlar

Inheritance yani miras alma kurallarına göre final olan bir metod override edilemez yani ezilemez. Örneğin aşağıdaki kod içerisinde yer alan topla metodu bu sınıfı miras alan sınıf içerisinde ezilemez.

```
class Ornek {
```

```
	final void topla(int x, int y){
		// Bu metod ezilemez
	}
}
```

final sınıflar

Yine miras alma kurallarında final olan bir sınıf miras alınamaz. Örneğin aşağıdaki sınıf miras alınamaz.

```
final class Ornek {

}
```

Dikkat : Java yazım kurallarında bahsettiğimiz üzere static ve final tanımlanmış bir değişken Java içerisinde tamamen büyük harfle ve alt çizgi ile ayrılarak yazılır. Buna aşağıdaki gibi örnek verilebilir.

```
class Ornek {
	final static int SABIT_SAYI = 2;
	final static String DEMO_DEGERI = "ornek";
}
```