static kelimesi

static Java içerisinde değişkenler, metodlar, iç sınıflar ve import tanımları üzerinde kullanılabilir. Bunun haricinde Java içerisinde static alanlarda tanımlanabilir. static kelimesini Türkçe' ye çevirdiğimizde gelen sabitlik anlamı bellek adresinin sabitliğini ifade etmektedir.

static metod ve değişkenler bellek üzerinde static alan üzerinde tutulurlar. Bu yüzden nesne ile ilgili alanlara erişemezler. Örnek vermek gerekirse static bir metod static olmayan bir metod veya değişkene direkt olarak erişemez. Bu metod ve değişkenin bulunduğu nesne referansına ihtiyacı bulunmaktadır.

Aşağıdaki static ile tanımlanmış örnekler bulunmaktadır.

```
class Ornek {

	{
		System.out.println("static alan...");
	}

	static int x;
	static Ogrenci o;

	static void ornekMetod(){
		///...
	}

	static class AltSinif{

	}

	public static void main(String[] args) {

	}
}
```

Bu örnekte şimdiye kadar kullanmış olduğumuz static main metoduda bulunmakta. Main metodunun neden static olduğunu bölüm içerisinde görüyor olacağız.

static değişkenler

static kelimesinin nesne ve ilkel tipler üzerindeki çalışma prensibi bellek adresinin sabitlenmesidir. Daha doğru anlatımla başına static eklediğiniz herhangi bir ilkel tip veya nesne bellek üzerinde tekil olarak adreslenir.

Örnek vermek gerekirse aşağıdaki kod içerisinde static olarak tanımlanmış y değişkeni için o sınıftan 100 tane de nesne yaratılsada bellekte tekil alanda saklanmaktadır.

```
class Ornek {
```

```
	int x ;
	static int y ;

	public static void main(String[] args) {
		Ornek o1 = new Ornek();
		o1.x = 4;
		o1.y = 2;

		Ornek o2 = new Ornek();
		o2.x = 6;
		o2.y = 8;

		System.out.println("o1.x : " + o1.x);
		System.out.println("o1.y : " + o1.y); // y aynı referansa sahip

		System.out.println("o2.x : " + o2.x);
		System.out.println("o2.y : " + o2.y); // y aynı referansa sahip

	}
}
```

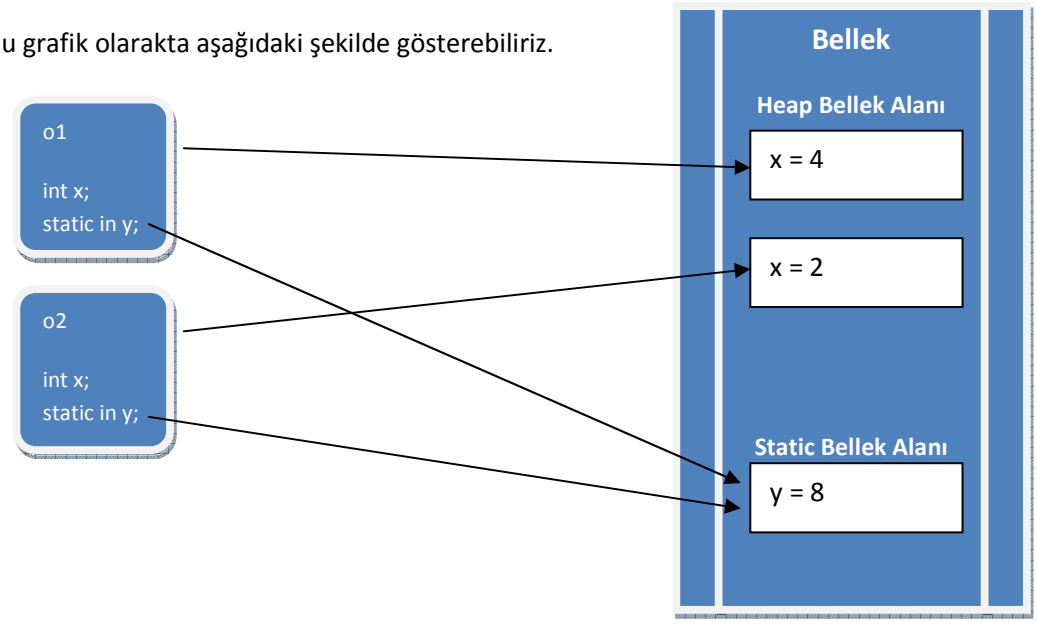
Yani yukarıdaki koda göre o1.y demekle o2.y demek arasında bir fark yok. Bunun nedeni y değişkeninin static olması ve bellek üzerinde sadece bir alanı referans etmesidir.

Buna göre yukarıdaki kodun çıktısı aşağıdaki gibi olacaktır.

```
o1.x : 4
o1.y : 8
o2.x : 6
o2.y : 8
```

Dikkat ettiyseniz x değişkenleri o1 ve o2 için verdiğimiz değerleri taşırken y değişkeni son verilen değeri taşıyor. Yukarıdaki tanımda da dediğimiz gibi burada o1.y demekle o2.y demek arasında bir fark bulunmuyor.

Bunu grafik olarakta aşağıdaki şekilde gösterebiliriz.



Yukarıdaki grafik tanımında x ler o1 ve o2 referans alanlarını gösterirken y' ler tek bir referans alanını belirtmektedir. Bu alana Java içerisinde static bellek alanı denir.

Aslında yukarıda yazdığımız uygulamada static olarak tanımlanmış değişkenlere nesne üzerinden erişmemiz bir hataya neden olamasada kullanım olarak çok doğru değildir. static değişkenlere direkt olarak sınıf üzerinden erişebiliriz ve doğru yöntem olarak bu şekilde erişmeliyiz. Aşağıdaki kod içerisinde daha önce yaptığımız örneğin doğru kullanımı bulunmaktadır.

```
class Ornek {

	int x ;
	static int y ;

	public static void main(String[] args) {
		Ornek o1 = new Ornek();
		o1.x = 4;
		o1.y = 2;

		Ornek o2 = new Ornek();
		o2.x = 6;
		o2.y = 8;

		System.out.println("o1.x : " + o1.x);
		//System.out.println("o1.y : " + o1.y); // hatalı kullanım
		System.out.println("Ornek.y : " + Ornek.y); // doğru kullanım

		System.out.println("o2.x : " + o2.x);
		//System.out.println("o2.y : " + o2.y); // hatalı kullanım
		System.out.println("Ornek.y : " + Ornek.y); // doğru kullanım
	}
}
```

Dikkat : static değişkenin nesne ile bir bağı yoktur ve nesne üzerinden erişilmemelidir. Bu tarz değişkenlere sınıf değişkeni denilmektedir. Bunun haricinde yine sınıf içerisinde tanımlanmış ve static olmayan nesne değişkenleri ile metodlar içerisinde yer alan local yani yerel değişkenler bulunmaktadır.

Not : static bir değişken bellek üzerinde tekildir ve nesneye ihtiyaç duymadan direkt yaratılır. Peki bir uygulamamızda yer alan static değişken ne zaman bellekte yer almaya başlar ? İlk bakışta uygulama ayağa kalktığı gibi denebilir ancak bu işlem değişkenin yer aldığı sınıfa ilk erişildiğinde yapılır.

static metodlar

static metodlar aynı static değişkenler gibi nesneye ihtiyaç duymadan direkt sınıf üzerinden erişilebilirler. Örnek vermek gerekirse matematik ile ilgili bir sınıf yaptık ve basit anlamda topla çıkar gibi metodlarımız var. Bu durumda matematik nesnesinde bir tane yaratmak hem gereksiz kod yazımı hemde yaşam döngüsü sırasında gereksiz nesne yaratılmasına neden olur. Bu durumda bu metodları static yapmak en mantıklısıdır.

Aşağıdaki örnekte Matematik sınıfına ait static ve static olmayan iki metod bulunmaktadır.

```
public class Matematik {

	static int topla(int x, int y) {
		return x + y;
	}

	int cikar(int x, int y) {
		return x - y;
	}

	public static void main(String[] args) {
		int toplamSonuc = Matematik.topla(6, 2);

		Matematik mat = new Matematik();
		int cikarmaSonuc = mat.cikar(6, 2);

	}
}
```

Bu örneğe göre topla metodu için nesne oluşturmazken çıkar metodu için oluşturmak zorunda kaldık.

Dikkat : static metodlar static olmayan yani nesne metod ve değişkenlerine erişemezler. Örneğin aşağıdaki kullanımda hata bulunmaktadır.

```
class Ornek {

	int x = 4;

	void metodBir(){
		metodUc(); // Nesne metodu static metod erişebilir
	}

	static void metodIki(){
		metodBir(); // static metod nesne metoduna erişemez
		x = 6; // static metod nesne değişkenine erişemez
	}

	static void metodUc(){

	}
}
```

Dikkat : static metodlar içerisinde instance yani nesne referansına erişemediğini için this kelimesini kullanamazsınız.

main metodu neden static ?

Baştan beri kullandığımız main metodunun static yapılmasının nedeni nedir ? Daha önceki konularda da belirttiğimiz gibi Java' da bir sınıf çalıştırıldığında önce main metoduna uğrar. Peki bu durumda sınıftan bellek üzerinde yaratılır mı ? Tabiki hayır. Aşağıdaki kodu inceleyecek olursak Ornek sınıfından herhangi bir nesne yaratılmıyor.

```
class Ornek {

	public static void main(String[] args) {
		System.out.println("Ornek çalıştı...");
	}
}
```

İşte bu yüzden main metodu sınıfın bellek üzerinde bir referansına sahip olmadan çalıştırılabilmelidir. Bunun yapmanın tek yoluda o metodu static yapmaktır.

static alanlar

Daha önceki konularda Java içerisinde süslü parantezler ile alanlar tanımlayabileceğimizi belirtmiştik. Sınıflar içerisinde de static alan tanımları yapabiliriz. Bunun anlamı sınıfa erişilmesi ile birlikte tek seferlik çalıştırılabilecek bir kod alanıdır.

Örnek olarak aşağıdaki kod içerisinde static bir alan bulunmaktadır ve bu alan main metodundan önce çalıştırılır. Bunun nedeni main thread mekanizması main metodunu çalıştırmadan sınıfa uğrar bu arada static alan çalıştırılır. Biz burada sınıfa erişimle birlikte bir değer ataması veya bir hazırlık işlemi yaptırabilirdik.

```
class Ornek {

	static {
		System.out.println("static alan çalıştı");
	}

	public static void main(String[] args) {
		System.out.println("main metodu çalıştı");
	}
}
```

Buna göre bu kodun çıktısı aşağıdaki gibidir.

```
static alan çalıştı
main metodu çalıştı
```

Not : static alanlar sadece static metod ve değişkenlere erişebilirler.

Dikkat : Java' da çalıştırılan yani iş yapan kodlar metodlar veya static alanlar içerisinde olmalıdır. Örnek olarak aşağıdaki örnekte sınıf içerisine yazılmış bir yazdırma kodu derleme hatasına neden olur.

```
class Ornek {

	System.out.println("Merhaba Dünya"); // Bu satır hata verir
}
```

static sınıflar

Bir sınıf tanımı static olabilir mi ? Aslında Java içerisinde sadece inner class yani iç sınıflar static olabilir. Bu durumda iç sınıf bulunduğu nesne referansına sahip olmadan oluşturulabilir.

Bunu aşağıdaki örnekle açıklayacak olursak. SinifBir ve SinifIki iç sınıf olarak tanımlanmışlardır ve SinifIki static olarak belirtilmiştir. Bu durumda görüldüğü gibi StaticIki Ornek sınıfından bir nesneye ihtiyaç duyulmadan direkt olarak yaratılabiliyor.

```
class Ornek {

	class SinifBir{

	}

	static class SinifIki{

	}

	public static void main(String[] args) {

		SinifBir s1 = new Ornek().new SinifBir();
		SinifIki s2 = new SinifIki();
	}
}
```

static import

Henüz import kelimesini görmesekte Java 5 sürümü ile birlikte gelen static import özelliğinden bahsedelim.

Klasik import tan farklı olarak static import gelen sınıfın tamamı yerine belirli bir static metod veya değişkenin ilgili sınıf içerisine getirilmesini sağlar. Örneğin aşağıdaki kod içerisinde java.lang.Math sınıfının içerisinde bulunan static PI değişkeni ile static cos() metodunu direkt olarak Ornek sınıfı içerisine aldık.

```
import static java.lang.Math.PI;
import static java.lang.Math.cos;


class Ornek {

	public static void main(String[] args) {
		System.out.println(PI);
		System.out.println(cos(20));
	}
}
```

Dikkat : Sadece static metod ve değişkenler static import ile getirilebilir.