# Operatörler

Java' da aritmetiksel, mantıktal, bit tabanlı ve ilişkisel olmaz üzere operatorler bulunmaktadır. Bunlardan bir kısmını daha doğrusu en çok kullanılanları görüyor olacağız. Diğer bölümleri internet yardımıyla araştırıyor olabilirsiniz.

## Aritmetiksel Operatörler

Java içerisinde kullanılan operatörler aşağıdaki gibidir. Bu operatörler sayısal operasyonlarda kullanılmanın yanında + ve += operatörü String işlemleri için birleştirme ve ekleme operasyonlarında da kullanılırlar.

| Operatör | Açıklama |
|---|---|
| = | Atama |
| + | Toplama |
| - | Çıkarma |
| * | Çarpma |
| / | Bölme |
| % | Mod alma |
| ++ | Bir artırma |
| -- | Bir eksiltme |
| += | Kendisine toplayıp atama |
| -= | Kendisinden çıkarıp atama |
| *= | Kendisi ile çarpıp atama |
| /= | Kendisine bölüp atama |
| %= | Mod alıp atama |

Aşağıdaki örnekte operatörlerin kullanımı bulunuyor.

```
public class Ornek {
	public static void main(String[] args) {
		int x = 4;
		int y = 2 + 6;
		int z = y - x;
		y += 2;
	}
}
```

## Artırma ve azaltma

Aritmetik operatörler içerisinde yer alan ++ ve -- operatörleri değer üzerinde artırma ve eksiltme yapar. Buraya dikkat etmek gerekiyor. Artırma ve eksiltme atanacak değere değil değerin kendisine etki ediyor.

Bunu örneklendirmek gerekirse aşağıdaki kodun çıktısı sizce ne olur ?

```
public class Ornek {
	public static void main(String[] args) {
		int x = 4;
		int y = x++;

		System.out.println("x : " + x);
		System.out.println("y : " + y);
	}
}
```

Burada bir çok kişi gibi sanırım x in 4 y nin 5 olmasını beklediniz. Ancak sonuç aşağıdaki gibi olacaktır.

```
x : 5
y : 4
```

Bunun sebebi öncelikle y değerine atama yapılıp sonrasında x değerinin artırılmasıdır.

**Dikkat :** ++ ve -- operatorleri aslında bir sonraki satırda işlem görürler. Bu artırma yada eksiltme öncesinde atama yapılmasını sağlar. Eğer Java' da öncelikli olarak artırma ve eksiltme yapılmasını istiyorsak bu operatorleri aşağıdaki gibi değişkenin başına eklemeliyiz.

```
public class Ornek {
	public static void main(String[] args) {
		int x = 4;
		int y = ++x;

		System.out.println("x : " + x);
		System.out.println("y : " + y);
	}
}
```

Bu durumda çıktı aşağıdaki gibi olacaktır.

```
x : 5
y : 5
```

**Not :** Mod alma matematikten de bilebileceğiniz gibi bir sayının bir sayıya bölümünden kalandır. Örnek olarak aşağıdak işlemin sonucun x değeri 10 un 3 bölümünden kalan olan 1' dir.

```
public class Ornek {
	public static void main(String[] args) {
		int x = 10;
		int sonuc = x % 3;
		System.out.println(sonuc);
	}
}
```

## İlişkisel Operatörler

İlişkisel operatörler iki değerin karşılaştırılması sonucunda geriye boolean yani mantıksal (true veya false) değer döndüren operatörlerdir.

| Operatör | Açıklama |
|---|---|
| == | Eşit mi ? |
| != | Eşit Değil mi ? |
| > | Büyük mü ? |
| < | Küçük mü = |
| >= | Büyük veya eşit mi ? |
| <= | Küçük veya eşit mi ? |

Aşağıdaki örnek iki karşılaştırma için çıktılar bulunuyor.

```
public class Ornek {
	public static void main(String[] args) {
		int x = 2;
		int y = 4;
		System.out.println(x == y);
		System.out.println(x != y);
		System.out.println(x > y);
		System.out.println(x < y);
		System.out.println(x >= y);
		System.out.println(x <= y);
	}
}
```

Buna göre kodun çıktısı aşağıdaki gibidir.

```
false
true
false
true
false
true
```

**Not :** Bu tarz ilişkisel operatörlerde özellikle döngü ve koşul gibi program gidişatını etkileyen operasyonlarda ihtiyaç duyacağız.

## Mantıksal Operatörleri

Uygulama içerisinde birden fazla koşula ihtiyacınız varsa bunun için mantıksal operatörleri kullabilirsiniz. Örneğin bir rakam 0 dan büyük ve 100 den küçükse bir işlem yapmak istiyorsanız bunu and koşulu ile birbirine bağlayabilirsiniz.

Aşağıda Java' da kullanılan mantıksal operatörler bulunmakta.

| Operatör | Açıklama |
|---|---|
| && | Ve – and |
| \|\| | Veya – or |

Bu operatörlerden && iki koşulu birbirine bağlayıp ikisininde true yani doğru döndürmesini bekler. Örneğin aşağıdaki kodun çıktısında ilk satır true dönerken ikinci satır false' tur.

```
public class Ornek {
	public static void main(String[] args) {
		int x = 2;
		int y = 4;
		System.out.println(x == 2 && y == 4);
		System.out.println(x == 2 && y == 6);
	}
}
```

Bunun sebebi ilk satırda iki değerde true dönerken ikinci satırda ikinci durum false' tur. Bu durumda ilk koşul doğru olsada ikinci koşul olumsız döndüğünden geriye false döndürülür.

Bir diğer operaratör olan or || or ile aynı işlemi yaparsak her iki durumda true döner. Bunun sebebi her iki satırda da koşullardan birinin true dönmesidir.

```
public class Ornek {
	public static void main(String[] args) {
		int x = 2;
		int y = 4;
		System.out.println(x == 2 || y == 4);
		System.out.println(x == 2 || y == 6);
	}
}
```

**Not :** && operatöründe ilk koşul false dönerse ikinci koşul sorgulanmaz. || operatöründe ise ilk koşul true dönerse ikinci koşula uğranmaz. Buna uygulama içerisinde kod işleyişinizi etkileyebileceği için dikkat etmeniz gerekmektedir.