



clarusway



clarusway



clarusway



clarusway

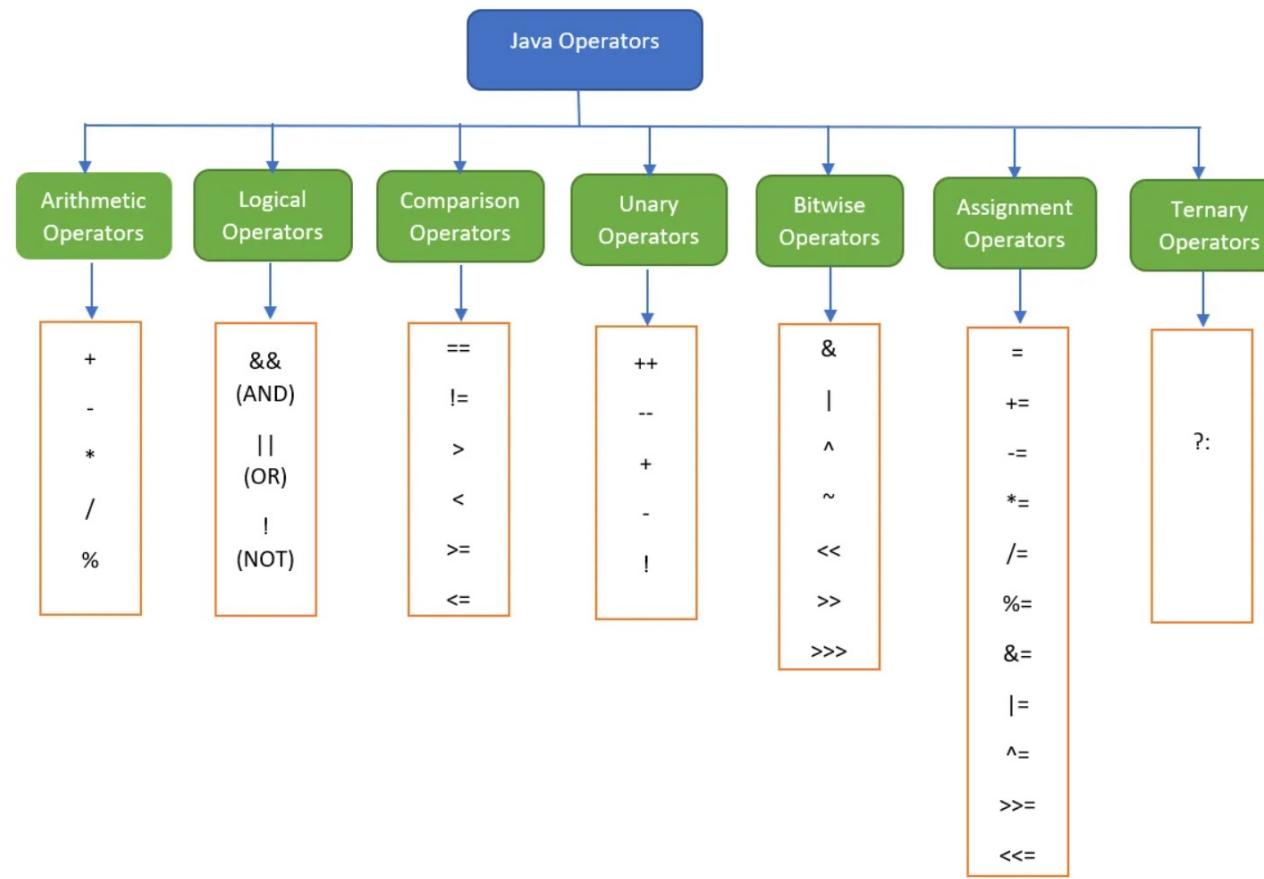


clarusway

Mathematical Operators Pre & Post Increment Modulus

Lesson:
JAVA Chapter 04

JAVA Operatörler



Aritmetik Operatörler

Operatör	Kullanılış	Açıklama
+	<code>değişken1 + değişken2</code>	değişken1 ile değişken2 yi toplar
-	<code>değişken1 - değişken2</code>	değişken1 ile değişken2 yi çıkarır
*	<code>değişken1 * değişken2</code>	değişken1 ile değişken2 yi çarpar
/	<code>değişken1 / değişken2</code>	değişken1 ,değişken2 tarafından bölünür
%	<code>değişken1 % değişken2</code>	değişken1 in değişken2 tarafından bölümünden kalan hesaplanır.



Unary (Tekli) Aritmetik Operatörler

Bir Arttırma ve Azaltma Tablosu

Operatör	Kullanılış Şekli	Açıklama
++	değişken++	Önce değişkenin değerini hesaplar sonra değişkenin değerini bir arttırır.
++	++değişken	Önce değişkenin değerini arttırır sonra değişkenin değerini hesaplar.
--	değişken--	Önce değişkenin değerini hesaplar sonra değişkenin değerini bir azaltır.
--	--değişken	Önce değişkenin değerini azaltır sonra değişkenin değerini hesaplar.



Increment (Variable Değerini Artırma)

```
int numA = 2;  
numA = numA + 3;
```

veya

```
numA += 3
```

?

```
int numB = 10;  
numB = numB * 7;
```

veya

```
numB *= 7
```

?



```
int numC = 7;  
numC++;
```

?

```
int numD = 11;  
numD++;
```

?



Decrement (Variable Değerini Azaltma)

```
int numA = 2;  
numA = numA - 3;
```



```
numA -= 3
```

?

```
int numB = 20;  
numB = numB / 5;
```



```
numB /= 5
```

?

```
int numD = 7;  
numD --;
```

?

```
int numE = 11;  
numE --;
```

?



Pre & Post Increment

- **Pre-Increment ve Post Increment operatörleri variable'ı artırma için kullanılır.**
- **Pre-Increment işleminde variable statement'da kullanılmadan önce artırılır veya azaltılır.**

```
public class OperatorExample{  
    public static void main(String args[]){  
        int x=10;  
        System.out.println(x++);//10 (11)  
        System.out.println(++x);//12  
        System.out.println(x--);//12 (11)  
        System.out.println(--x);//10  
    }  
}
```

- **Post Increment işleminde variable statement'da kullanılır, sonra artırılır veya azaltılır.**

```
public class OperatorExample{  
    public static void main(String args[]){  
        int a=10;  
        int b=10;  
        System.out.println(a++ + ++a);//10+12=22  
        System.out.println(b++ + b++);//10+11=21  
    }  
}
```



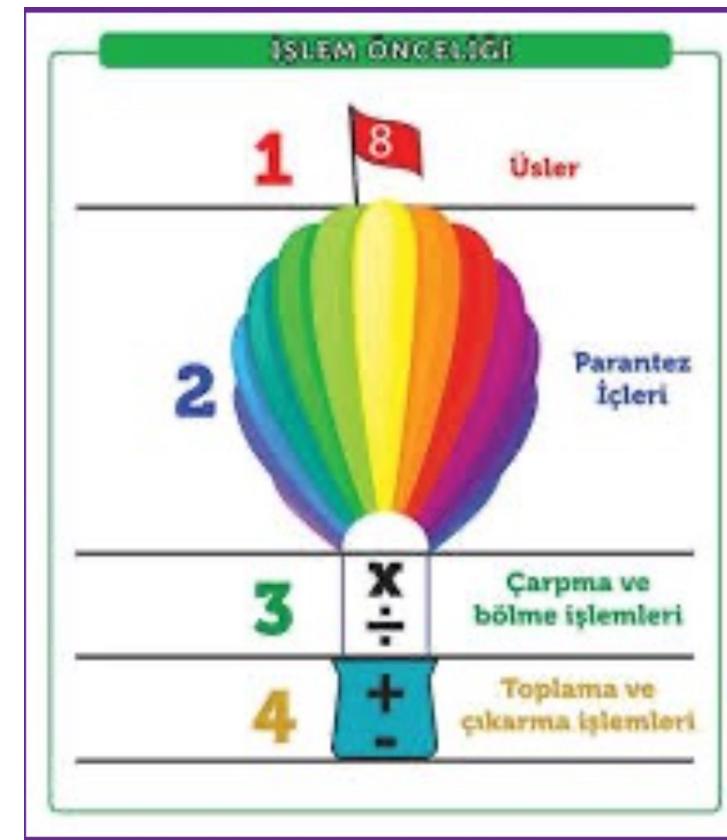
Matematiksel Operatörler

İşlem önceliği...

- 1- Üstel işlemler
- 2- Parantez içi işlemler
- 3- Çarpma Bölme işlemleri
- 4- Toplama Çıkarma işlemleri

ÖRNEK 1 $34 / 2 * (4-3) * 3 =$

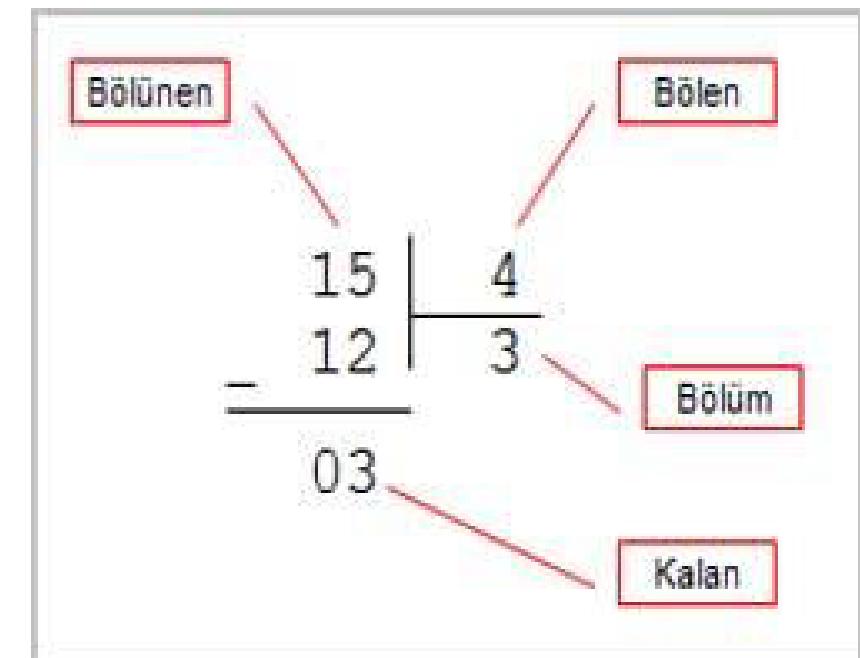
ÖRNEK 2 $7 + 3 * (21 + 9 / 3) - 23 =$



Modulus (%) Kalan Bulma

Modulus (%) bölme işleminde kalan değeri verir.

```
public class ClarusWay {  
  
    public static void main(String[] args) {  
        int num = 15 % 4;  
        System.out.println(num); // 3  
    }  
}
```





CLARUSWAY

WAY TO REINVENT YOURSELF

Assignment Operators (Atama Operatörleri)



clarusway



clarusway



clarusway



clarusway



clarusway

Assignment (Atama) Operatörleri

Java'da Atama Operatörleri

Atama operatörleri, değişkenlere değer atamak için kullanılan simgelerdir.
Java dilinde aşağıdaki atama operatörleri kullanılır;

- = Sağdaki değeri soldaki değişkene atar.
- += Soldakine sağdakini ekler, sonucu soldakine atar.
- = Soldakinden sağdakini çıkarır, sonucu soldakine atar.
- *= Soldakini sağdaki ile çarpar, sonucu soldakine atar.
- /= Soldakini sağdakine böler, sonucu soldakine atar.
- %= Soldakini sağdakine böler, kalanı (module) soldakine atar.



Assignment (Atama) Operatörleri

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3



Assignment (Atama) Operatörleri

```
public class OperatorExample{  
    public static void main(String args[]){  
        int a=10;  
        int b=20;  
        a+=4;//a=a+4 (a=10+4)  
        b-=4;//b=b-4 (b=20-4)  
        System.out.println(a);  
        System.out.println(b);  
    }  
}
```

```
public class OperatorExample{  
    public static void main(String[] args){  
        int a=10;  
        a+=3;//10+3  
        System.out.println(a);  
        a-=4;//13-4  
        System.out.println(a);  
        a*=2;//9*2  
        System.out.println(a);  
        a/=2;//18/2  
        System.out.println(a);  
    }  
}
```



Assignment (Atama) Operatörleri

```
public class OperatorExample{  
    public static void main(String args[]){  
        short a=10;  
        short b=10;  
        //a+=b;//a=a+b internally so fine  
        a=a+b;//Compile time error because 10+10=20 now int  
        System.out.println(a);  
    }  
}
```

Output:

Compile time error

```
public class OperatorExample{  
    public static void main(String args[]){  
        short a=10;  
        short b=10;  
        a=(short)(a+b);//20 which is int now converted to short  
        System.out.println(a);  
    }  
}
```

Output:

20





CLARUSWAY

WAY TO REINVENT YOURSELF

Comparison Operators (Karşılaştırma Operatörleri)



clarusway



clarusway



clarusway



clarusway



clarusway

Comparison (Karşılaştırma) Operatörleri

Karşılaştırma Operatörleri

Karşılaştırma operatörler, değişkenler içerisindeki verilerin arasında karşılaştırma yapmaya yarar. Bunları şu şekilde sıralayabiliriz:

Küçükfür (<)

Büyükfür (>)

Küçük Eşittir (<=)

Büyük Eşittir (>=)

Eşittir (==)

Eşit Değildir (!=)

Yukarıda sıralanan bu operatörler iki verinin karşılaştırmasını yapar.

Dönen değer **boolean** türündeki **true** veya **false**'dur.

Syntax:

variable1 relation_operator variable2



Comparison (Karşılaştırma) Operatörleri

Operatör 1: 'Eşittir' operatörü (==)

Bu operatör, verilen iki işlenenin eşit olup olmadığını kontrol etmek için kullanılır.

Operatör, sol taraftaki işlenen sağ tarafa eşitse true, aksi takdirde false döndürür.

Sözdizimi:

```
var1 == var2
```

İllüstrasyon:

```
var1 = "GeeksforGeeks"  
değişken2 = 20  
var1 == var2 false ile sonuçlanır
```



Comparison (Karşılaştırma) Operatörleri

Operatör 2: 'Eşit değil' Operatör(!=)

Bu operatör, verilen iki işlenenin eşit olup olmadığını kontrol etmek için kullanılır.

Operatöre eşit olanın tersi işlev görür. Sol taraftaki işlenen sağ tarafa eşit değilse true, aksi takdirde false döndürür.

Sözdizimi:

```
var1 != var2
```

İllüstrasyon:

```
var1 = "GeeksforGeeks"  
değişken2 = 20
```

```
var1 != var2 sonucu true
```



Comparison (Karşılaştırma) Operatörleri

Operatör 3: 'Büyükür' operatörü(>)

Bu, ilk işlenenin ikinci işlenenden büyük olup olmadığını kontrol eder. Operatör, sol taraftaki işlenen sağ taraftan daha büyük olduğunda true değerini döndürür.

Sözdizimi:

```
var1 > var2
```

İllüstrasyon:

```
var1 = 30  
değişken2 = 20
```

```
var1 > var2 sonucu true
```



Comparison (Karşılaştırma) Operatörleri

Operatör 4: 'Küçüktür' Operatör(<)

Bu, ilk işlenenin ikinci işlenenden küçük olup olmadığını kontrol eder. Operatör, sol taraftaki işlenen sağ taraftan küçük olduğunda true değerini döndürür. Büyüktür operatörünün tersi çalışır.

Sözdizimi:

```
var1 < var2
```

İllüstrasyon:

```
var1 = 10  
değişken2 = 20
```

```
var1 < var2 true ile sonuçlanır
```



Comparison (Karşılaştırma) Operatörleri

Operatör 5: Büyüktür veya eşittir ($>=$)

Bu, ilk işlenenin ikinci işlenenden büyük veya ona eşit olup olmadığını kontrol eder.

Operatör, sol taraftaki işlenen sağ taraftan büyük veya ona eşit olduğunda true değerini döndürür.

Sözdizimi:

```
var1 >= var2
```

İllüstrasyon:

```
var1 = 20  
değişken2 = 20  
var3 = 10
```

```
var1 >= var2 sonucu true  
var2 >= var3 sonucu true
```



Comparison (Karşılaştırma) Operatörleri

Operatör 6: Küçüktür veya eşittir (<=)

Bu, ilk işlenenin ikinci işlenenden küçük veya ona eşit olup olmadığını kontrol eder.

Operatör, sol taraftaki işlenen sağ taraftakinden küçük veya ona eşit olduğunda true değerini döndürür.

Sözdizimi:

```
var1 <= var2
```

İllüstrasyon:

```
var1 = 10  
değişken2 = 10  
var3 = 9
```

```
var1 <= var2 sonucu true  
var2 <= var3 false ile sonuçlanır
```





CLARUSWAY

WAY TO REINVENT YOURSELF

Logical Operators (Mantıksal Operatörler)



clarusway



clarusway



clarusway



clarusway



clarusway

Logical (Mantıksal) Operatörler

Mantıksal Operatörler

Mantıksal operatörler adından da anlaşılacağı gibi elde edilen verilerin mantıksal olarak test edilmesinde kullanılan işlemcilerdir. Mantıksal operatörler **doğru (true)** ve **yanlış (false)** gibi karşılaştırmalar yapmak için kullanılan operatörlerdir.

Operatör	İsim	Açıklama	Örnek
&&	Ve (AND)	Her iki ifade de doğruysa true döndürür	$x < 10 \&\& x < 25$
	Veya (OR)	İfadelerden biri doğruysa true döndürür	$x < 5 x < 4$
!	Ters İşlem	Sonucu tersine çevirir	$!(x < 5 \&\& x < 10)$



Logical (Mantıksal) Operatörler

- **&& Koşullu VE(AND) işlemi yapılmaktadır. Bütün sonuçlar kendi içinde true ise sonuç true, en az birisi bile false ise sonuç false olur.**

```
1 System.out.println(10==10 && 5<10); //Verdiğimizörnekte iki karşılaştırmada true olduğu için çıktıda true olsun  
2 System.out.println(10==10 && 10<5); //Verdiğimizörnekte 5 büyüktür 10 karşılaştırması yanlış olduğu için false olsun
```

Ekran Çıktısı

```
1 true  
2 false
```



& ile && Arasındaki Fark

& isareti kullanıldığında Java isaretin iki yanındaki mantıksal ifadelerin ikisini de kontrol eder. Bu işlem kodumuzu yavaşlatır

40<30 & 50==50 & 60>50

ilk karşılaştırma yanlış olmasına rağmen Java tüm karşılastırmaları kontrol etmeye devam eder.

&& isareti kullanıldığında ise Java en baştan kontrol etmeye başlar, mantıksal ifadelerin birinde yanlışlı bulursa sonrakileri kontrol etme ihtiyacı duymaz. Bu işlem kodumuzu hızlandırır

40<30 && 50==50 && 60>50

ilk karşılaştırma yanlış olduğunu görünce Java diğer karşılastırmaları kontrol etmeden alt satıra geçer.



Logical (Mantıksal) Operatörler

- || Koşullu VEYA işlemi yapar. Karşılaştırılan iki değerden en az biri true olduğu sürece, true döndürmektedir.

```
1 | System.out.println(10==10 || 10<5); //10==10 true değerini döndürdüğü için sonuç true olarak çıkmaktadır. 
```

Ekran Çıktısı

```
1 | true 
```



Logical (Mantıksal) Operatörler

- ! NOT (DEĞİL) işlemi yapar. Sonucu true ise false, false ise true yapmaktadır.

```
1 | System.out.println(!(10==10));
```



Ekran Çıktısı

```
1 | false
```



!= Esit degildir işaretü

boolean sonuc1= 5+2 != 7;

sonuc1 degeri **false** olur

System.out.println(5*2 != 15);

true yazdirir



Logical (Mantıksal) Operatörler

genel sınıf OperatorÖrneği{

```
public static void main(String args[]){
```

```
int a= 10 ;
```

```
int b= 5 ;
```

```
int c= 20 ;
```

```
System.out.println(a>b||a<c); //doğru || doğru = doğru
```

```
System.out.println(a>b|a<c); //doğru | doğru = doğru
```

//|| vs |

```
System.out.println(a>b||a++<c); //doğru || doğru = doğru
```

```
System.out.println(a); //10 çünkü ikinci koşul kontrol edilmedi
```

```
System.out.println(a>b|a++<c); //doğru | doğru = doğru
```

```
System.out.println(a); //11 çünkü ikinci koşul kontrol edildi
```

```
}
```

genel sınıf OperatorÖrneği{

```
public static void main(String args[]){
```

```
int a= 10 ;
```

```
int b= 5 ;
```

```
int c= 20 ;
```

```
System.out.println(a<b&&a++<c); //yanlış && doğru = yanlış
```

```
System.out.println(a); //10 çünkü ikinci koşul kontrol edilmedi
```

```
System.out.println(a<b&a++<c); //yanlış && doğru = yanlış
```

```
System.out.println(a); //11 çünkü ikinci koşul kontrol edildi
```

```
}
```

genel sınıf OperatorÖrneği{

```
public static void main(String args[]){
```

```
int a= 10 ;
```

```
int b= 5 ;
```

```
int c= 20 ;
```

```
System.out.println(a<b&&a<c); //yanlış && doğru = yanlış
```

```
System.out.println(a<b&a<c); //yanlış ve doğru = yanlış
```

```
}
```



Concatenation (String Variable Birleştirme)

Birden çok String variable'a "+" işlemi yapılrsa JAVA String variable'ları **concatenation** (birleştirme) yaparak yeni bir String oluşturur.

```
String a = "Selam";  
String b = "Javacan'lar";
```

```
System.out.println(a + b); // -> SelamJavacan'lar
```

```
System.out.println(a + " " + b); // -> Selam Javacan'lar
```



Concatenation (String Variable Birleştirme)

* TRICK :) Bir aritmetik işlem içinde String kullanılırsa, aritmetik işlem önceliğine göre işlem yapılip String variable'a Concatenation yapılır.

```
String a="Clarusway";
int b=14;
int c=53;

System.out.println(a+b+c); // -> Clarusway1453

System.out.println(c+b+a); // -> 67Clarusway

System.out.println(a+(b+c)); // -> Clarusway67

System.out.println(a+b*c); // -> Clarusway742
```



THANKS!

Any questions?

HalUk Bilgin | JAVA Backend Developer

BAŞARI GAYRETE AŞIKTIR ☺

