







OPERATÖRLER

Hedefler

Bu üniteyi çalıştıktan sonra;

	Java dilindeki aritmetiksel operatörleri öğreneceksiniz,
	Mantıksal operatörleri öğreneceksiniz,
	Artırma ve Azaltma operatörlerini öğreneceksiniz,
	Karşılaştırma operatörlerini öğreneceksiniz
	Bit işlem operatörlerini öğreneceksiniz,
	Matematiksel Operatörlerin öncelik sırasını öğreneceksiniz.

İçindekiler

OPERATÖRLER

1. Aritmetiksel Operatörler
2. Artırma Ve Azaltma Operatörleri
3. Aritmetiksel Atama Operatörleri
4. Karşılaştırma Operatörleri
5. Mantıksal Operatörler
6. Bit İşlem Operatörleri
7. Matematiksel Operatörler Ve Öncelik Sıraları
8. Konunun Özeti
9. Çoktan Seçmeli Değerlendirme Soruları

OPERATÖRLER

İşlem yapmamızı sağlayan işaretlere **operatör (işleç)** adı verilir. Genellikle, **aritmetiksel** { +, -, *, /, gibi }, **mantıksal** { ve, veya, değil, gibi } ve **karşılaştırma** { <, >, =, gibi } operatörleri isimleri altında sınıflandırılırlar ve her programlama dilinde farklı şekillerde gösterilebilirler. Ayrıca bu genel sınıflandırmaya ek olarak artırma ve azaltma operatörleri, aritmetiksel atama operatörleri ve bitset operatörler başlıkları altında Java dilinde kullanılan diğer operatörleri de açıklayacağız.

Aritmetiksel Operatörler

Aritmetiksel işlemlerde kullanılan (dört işlem gibi) operatörlerdir. Matematiksel işlemlerde kullanılan ilgili operatörler ve bilgisayar ortamlarındaki gösterimleri aşağıdaki tabloda verilmiştir.

İŞLEMLER	Aritmetik Operatör	Matematiksel gösterimi	JAVA dili gösterimi
Toplama	+	$X+Y$	$X + Y$
Çıkartma	-	$X-Y$	$X - Y$
Çarpma	*	$(XY), (X.Y), (X*Y)$	$X * Y$
Bölme	/	$\frac{X}{Y}$ X/Y ve $\frac{X}{Y}$	X / Y
Üs alma	Yoktur	3^2	$3*3$ <code>Math.pow(3,2);</code>
Karekök alma	Yoktur	$\sqrt{3}$	<code>Math.sqrt(3);</code>
Mod Alma (Kalan)	%	$X \text{ Mod } Y$	$X \% Y$
String Birleştirme	+	BadeSare	"Bade" + "Sare"
Negatif	-	$-Y$	$-Y$
Değer Aktarma	=	$Y \Rightarrow X$	$X=Y$

Java'da üs ve karekök almak için özel bir operatör yoktur. Bunun yerine Math sınıfının pow() ve sqrt() yöntemleri kullanılır. Üs almak için Math.pow(), Karekök almak için Math.sqrt() kullanılır. Örneğin b^2 işlemi Math.pow(b,2) şeklinde ifade edilir.

Örnek 1.

a) $z = -\frac{5 \cdot a^{3/2}}{4}$ şeklindeki matematiksel ifadenin Java dilinde kodlaması nasıl yapılır?

Çözüm:

Kodlanması : `z = - (5*Math.pow(a,(3/2)) / 4);`

b) a=9 için z değerini bulan programı yazınız.

Çözüm:

```
public class AritmetikselOperatorler {  
    public static void main(String[] args) {  
        double z;  
        int a = 9;  
        float x = (float)3/2;  
        z = -(5 *Math.pow(a, x) / 4);  
        System.out.printf("z = " + z);  
    }  
}
```

Açıklama

Bu üslü ifadenin özellikle float yani ondalıklı sayı tipinde olduğunu programa söylemeliyiz, bunun için `float x = (float)3/2;` komut satırı ile x değişkenine atayıp, programı aşağıdaki gibi yeniden yazalım. Buna göre doğru sonuçta; z = -33.75 olacaktır.

Artırma ve Azaltma Operatörleri

Artırma ve azaltma operatörleri sayaç operatörleridir. Otomatik olarak bir artırma ya da bir azaltma işlemi yaparlar. Bu operatörlerin, Ön Artırma (preincrement), Son Artırma (Postincrement), Ön Azaltma (predecrement) ve Son Azaltma (postdecrement) olmak üzere dört farklı şekilde kullanımı vardır.

İŞLEMLER	Aritmetik Operatör	Kullanılışı	Açıklama
Ön Artırma	++	++Değişken (A=++B)	Önce değişken değerini 1 artır, sonra kullan. (B=B+1; A=B ;)
Son Artırma	++	Değişken++ (A=B++)	Önce değişken değerini al, kullan sonra 1 artır.

			(A=B;B=B+1)
Ön Çıkarma	--	--Değişken (A=--B)	Önce değişken değerini 1 azalt, sonra kullan. (B=B-1;A=B;)
Son Çıkarma	--	Değişken-- (A=B--)	Önce değişken değerini al, kullan sonra 1 azalt. (A=B; B=B-1)

Örnek 2. Aşağıdaki ‘operator.java’ isimli programın ekran çıktısını inceleyiniz.

Ekran çıktılarının açıklama satırları ‘//’ ile verildiğine dikkat ediniz.

```
class operator {
public static void main(String[] args) {
    int B=5;
    System.out.println (B);    // Ekrana 5 yazar
    System.out.println (B++); // Ekrana 5 yazar ama B nin değerini 1 artırır
    System.out.println (B);    // Ekrana 6 yazar ama B nin değerini 1 artırır
    System.out.println (++B); // Ekrana 7 yazar
    System.out.println (--B); // Ekrana 6 yazar
    System.out.println (B--); // Ekrana 6 yazar ama B nin değerini 1 azaltır
    System.out.println (B);    // Ekrana 5 yazar
}
}
```

Aritmetiksel Atama Operatörleri

Aritmetiksel atama operatörleri, standart aritmetiksel operatörler ile atama operatörünün birleşmesi ile üretilen operatörlerdir ve gösterimi, aritmetiksel operatör ve ‘=’ parametresinin yan yana birleştirilmesi şeklindedir. { Örneğin; +=, -=, *=, /= gibi }

Mesela, ‘+=’ operatörü, program içerisinde aşağıdaki gibi kullanılabilir;

```
Toplam +=5; //bu şekildeki tanımlama,
Toplam=Toplam + 5; // işlemi ile eşdeğerdir.
```

Bu örnekte += operatörünün yaptığı işlevi, “**değişkene eşitliğin sağındaki değeri ekle ve sonucu yine değişkene aktar**” şeklinde açıklayabiliriz.

İşlem	Operatör	Java Gösterimi	Eşdeğeri
-------	----------	----------------	----------

Topla, aktar	+=	A += B	A = A + B
Çıkar, aktar	-=	A -= B	A = A - B
Çarp, aktar	*=	A *= B	A = A * B
Böl, aktar	/=	A /= B	A = A / B
Kalanı al, aktar	%=	A %= B	A = A % B
Sola kaydır, aktar	<<=	A<<=B	A=A<<B
Sağa kaydır, aktar	>>=	A>>=B	A=A>>B
Aktar	=	A = B	A = B

Örnek 3. Aşağıdaki aritmetiksel atama operatörlerin ekran çıktısı ne olur, inceleyiniz.

```

public class Atama {
    public static void main(String[] args) {
        int u=3; int v=4;
        u += v;
        System.out.println("u += v :"+ u);
        u -= v;
        System.out.println("u -= v :"+ u);
        u *= v;
        System.out.println("u *= v :"+ u);
        u /= v;
        System.out.println("u /= v :"+ u);
        u <<= v;
        System.out.println("u <<= v :"+ u);
    }
}

```

Programı çalıştırdığımızda aşağıdaki ekran çıktısını elde ettik.

```

<terminated> Atama [Java Application]
u += v :7
u -= v :3
u *= v :12
u /= v :3
u <<= v :48

```

Karşılaştırma Operatörleri

Verilerin birbiriyle karşılaştırılmasında kullanılır. Sonuç **doğru** ise **1** ya da **true**, yanlış ise **0** ya da **false** değerini alır. Bir karşılaştırmanın, koşulun söz konusu olduğu döngü veya karar yapılarında bu karşılaştırma operatörlerinden biri mutlaka kullanılır.

İŞLEM	OPERATÖR	KARŞILAŞTIRMA NOTASYONU	JAVA NOTASYONU
Küçüktür	<	$A < B$	<code>A < B</code>
Büyüktür	>	$A > B$	<code>A > B</code>
Küçük eşit	<=	$A \leq B$	<code>A <= B</code>
Büyük eşit	>=	$A \geq B$	<code>A >= B</code>
Eşit mi?	==	$A = B$	<code>A == B</code>
Eşit değil	!=	$A \neq B$	<code>A != B</code>

Ayrıca Java’da iki sayıyı doğrudan karşılaştırmak için Math sınıfının **max** ve **min** fonksiyonlarını kullanabilirsiniz.

- `Math.max(78, 45)` // 78 sonucunu üretir
- `Math.min(78, 45)` // 45 sonucunu üretir

Karşılaştırma örnekleri;

Karşılaştırma	Sonuç
<code>4 < 7</code>	<code>true</code>
<code>23.1 >= 54.34</code>	<code>false</code>
<code>'a' != 'A'</code>	<code>true</code>
<code>'a' == 'a'</code>	<code>true</code>
<code>3 <= 6</code>	<code>true</code>
<code>Math.max(12, 56)</code>	<code>56</code>
<code>Math.min(12, 56)</code>	<code>12</code>

Not: `A=B` işlemi, “B nin değerini A’ya aktar” anlamına gelirken `A==B` işlemi, “A değişkeninin değeri B değişkenin değerine eşit mi?” anlamına gelir.

Mantıksal Operatörler

Mantıksal operatörler, birden fazla verinin birbiri ile kıyaslanması durumunda kullanılırlar. Kıyaslanmanın sonucuna bağlı olarak da hangi iş(lem)lerin yapılacağına karar verilir. Mantıksal operatörlerin sonucu, boolean tipi değişkenlerde tutulur ve sonuç olarak sadece birbirinin tersi olan iki değerden biri (true-false, yes-no, evet-hayır, doğru-yanlış, ‘1’-‘0’ gibi) üretilir.



Mantıksal operatörlerin işlevlerini şöyle açıklayabiliriz;

İŞLEM	OPERATÖR	KARŞILAŞTIRMA NOTASYONU	JAVA NOTASYONU
Mantıksal VE	&&	A VE B	A && B
Mantıksal VEYA		A VEYA B	A B
Değil	!	A DEĞİL B	A ! B
Özel VEYA	^	A ÖZELVEYA B	A ^ B

- **&& Operatörü;** Lojik VE işlemi operatörüdür, lojik ifadelerin her ikisinin de doğru olması halinde **true** sonucunu aksi takdirde **false** sonucunu üretir.
- **|| Operatörü;** Lojik VEYA işlemi operatörüdür, lojik ifadelerden herhangi birinin doğru olması halinde **true** sonucunu aksi takdirde **false** sonucunu üretir.
- **^ Operatörü;** Lojik özel VEYA işlemi operatörüdür, lojik ifadelerden her ikisinin de aynı olması halinde **false** aksi takdirde **true** sonucunu üretir.
- **! Operatörü;** lojik DEĞİL işlemi operatörüdür, lojik ifadenin değilini (tersini) alır. İfade doğru ise **false**, yanlış ise **true** sonucunu üretir.

X	Y	! X	! Y	X && Y	X Y	X ^ Y
true	true	false	false	true	true	false
true	false	false	true	false	true	true
false	true	true	false	false	true	true
false	false	true	true	false	false	false

BİT İŞLEM OPERATÖRLERİ

Bit işlem operatörleri, verilen sayının (değerin) bit karşılığını alıp, bu bitler üzerinde tek tek mantıksal işlem gerçekleştirirler. Bit işlemleri ile ilgili değişkenlerin tamsayı veri tipinde (**long**, **int**, **short** ve **byte**) olmasına dikkat edilmelidir, diğer veri tipleri (**boolean**, **float**, **double gibi**) kullanılmaz.

Ayrıca bit düzeyinde bir sayının bir basamak (bit) sola (<<) kaydırılması o sayının 2 ile çarpılması, bir bit sağa (>>) kaydırılması ise o sayının 2'ye bölünmesi anlamına gelir.

İşlem	Operatör	Karşılaştırma Notasyonu	Java Notasyonu	Örnek	Sonuç
Bit düzeyinde VE	&	A VE B	A & B	3 & 5;	1
Bit düzeyinde VEYA		A VEYA B	A B	3 5	7
Bit düzeyinde Özel VEYA	^	A Özel VEYA B	A ^ B	3 ^ 5	6
Değil veya 1'e tümleyen	~	Değil A	~ A	~3	-4
Sola kaydır	<<	A << basamak	A << değer	3 << 2	12
Sağa kaydır	>>	A >> basamak	A >> değer	9 >> 2	2
Sağa kaydır (işaretsiz sayılar için)	>>>	A >>> basamak	A >>> değer	-4 >>> 28	15

Ayrıca yukarıdaki tabloda verilen bit işlem operatörlerinin atama operatörleri ile birleşiminden oluşan ' >>=, <<=, >>>=, &= ^= |= ' operatörlerini de kullanabilirsiniz.

Bit işlem operatörlerinin mantıksal işlevini bir tablo halinde örneklersek;

A	B	A B	A & B	A ^ B	~A
0	0	0	0	0	1
1	0	1	0	1	0
0	1	1	0	1	1
1	1	1	1	0	0

Tabi bit işlem operatörleri sayıları bit bit ele aldığı için ondalık sayıların ikili (Binary) sayı karşılıklarını bilmemiz gerekir. Örneğin 3 sayısının ikili(binary) karşılığı "0000 0000 0000 0000 0000 0000 0000 0011" dir, 5 sayısının ikili karşılığı "0000 0000 0000 0000 0000 0000 0000 0101" dır. Negatif sayıların gösterimi ise ikiye tümleyen (*two's complement*) aritmetiği ile gösterilir. Örneğin -4 sayısı; +4 sayısının ikili karşılığının ters çevrilip (değilinin alınıp) , +1 eklenmesi ile elde edilir ve değeri "1111 1111 1111 1111 1111 1111 1111 1100" dır.

Herhangi bir tamsayıyı ikili sayı formatında 32 bit halinde String ifade şeklinde ekranda görmek için "**Integer.toBinaryString(sayı)**" komutunu kullanmamız gerekir.

Bu operatörlerle ilgili olarak aşağıdaki örnekleri inceleyiniz.

Örnek 4. Aşağıdaki programın ekran çıktısı ne olur?

```
public class test2 {
    public static void main(String[] args) {
        int x = 2; //10 (ikili)
        int y = 3; //11 (ikili)
        System.out.println("10 & 11 = "+ Integer.toBinaryString (x&y));
        System.out.println("10 | 11 = "+ Integer.toBinaryString (x|y));
        System.out.println("10 ^ 11 = "+ Integer.toBinaryString (x^y));
        System.out.println(" ~10 = "+ Integer.toBinaryString (~x));
    }
}
```

Programın ekran çıktısı aşağıdaki gibi olur.

```
10&11 = 10  
10|11 = 11  
10^11 = 1  
~10 = 11111111111111111111111111111111
```

Kendiniz Uygulayınız: Aşağıdaki programın ekran çıktısı ne olur?

```
public class test {
    public static void main(String[] args) {
        int b =1, c=2;
        System.out.println(" (not b) = " + Integer.toBinaryString(~b ));
        System.out.println(" (b VEYA c) = " + Integer.toBinaryString(b | c) );
    } }

```

Örnek 5. Aşağıdaki programın ekran çıktısı ne olur?

```
public class Ornek9{
    public static void main(String[] args) {
        int a = 8;
        System.out.println(" (a >> 1)= " + (a >> 1) );
        System.out.println(" (a << 2) = " + (a << 2) );
        System.out.println(" (a >>> 2) = " + (a >>> 2) );
    }
}
```

Çözüm:

Bit kaydırma komutlarının işlevlerini gösteren bir örnek. Programın adım adım çalışmasını inceleyelim.

a=8 işleminin ikili (bitsel) karşılığı **(1000)₂** dır. Buna göre;

```
System.out.println(" (a >> 1)= " + (a >> 1) );
```

(a >> 1) işlemi ile **a** sayısının içeriği 1 basamak sağa kaydırılır ve sol baştaki ilk bit(boştaki bit) 0 değerini alır. **(0100)₂** değerinin karşılığı **4** sayısı ekranda görülür.

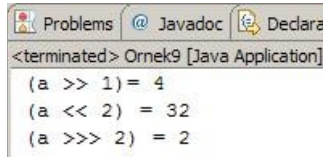
```
System.out.println(" (a << 2) = " + (a << 2) );
```

(a << 2) işlemi ile **a** sayısının içeriği 2 basamak sola kaydırılır ve her kaydırmada sağ baştaki ilk bit (boştaki bitler) 0 değerini alır. **(100000)₂** değerinin karşılığı **32** sayısı ekranda görülür.

```
System.out.println(" (a >>> 2) = " + (a >>> 2) );
```

(a >>> 2) ifadesi ile (a >> 2) arasında işlemsel olarak bir fark yoktur, sonuç olarak **a** sayısının içeriği 2 basamak sağa kaydırılır. **(0010)₂** değerinin karşılığı **2** sayısı ekranda görülür.

Programın ekran çıktısı aşağıdaki gibidir;



```
<terminated> Ornek9 [Java Application]
(a >> 1) = 4
(a << 2) = 32
(a >>> 2) = 2
```

Matematiksel Operatörler Ve Öncelik Sıraları

Matematiksel operatörlerin öncelik sırası aşağıdaki tabloda verilmiştir. Aynı önceliğe sahip işlemlerde soldaki daha önce işlem görür.

Öncelik Sırası	İşlemler	İşlem Simgesi
1	Sayıları negatifleştirme	-
2	Parantez içi	(...)
3	Matematiksel fonksiyonlar	Sin, Cos, v.b.
4	Son artırma/azaltma	<ifade>++, <ifade>--
5	Ön Artırma/Azaltma	++<ifade>, -- <ifade>, !<ifade>
6	Çarpma, Bölme veya Kalan	*, /, %
7	Toplama veya Çıkarma	+ veya -
8	Kaydırma	<< veya >>, >>>
9	Karşılaştırma	<, <=, >, >=, ==, !=
10	Bit düzeyinde AND, XOR, OR	&, ^,
11	Mantıksal AND, OR	&&,
12	Atama	=, +=, -=, *=, /=, %=
13	Bitsel Atama	>>=, <<=, >>>=
14	Boolean atama	&=, ^=, =

Örnek 6. $\frac{6+2.4}{2} - 3^2 + \frac{6/3}{2-3}$ Şeklindeki matematiksel ifadenin bilgisayar ortamındaki kodlanması ve işlem sonucu aşağıdaki şekilde olur.

Kodlanması : `(6 + 2 *4) /2 - 3 *3 + (6/3) / (2-3)`

İşlem önceliği :

$$\begin{array}{rcl}
 (6 + \underline{8}) /2 - 3 *3 + (6/3) / (2-3) & & \\
 \underline{14} /2 - 3 *3 + (6/3) / (2-3) & & \\
 \underline{7} - 3 *3 + (6/3) / (2-3) & & \\
 7 - \underline{9} + (6/3) / (2-3) & & \\
 7 - 9 + \underline{2} / (2-3) & & \\
 7 - 9 + 2 / \underline{-1} & & \\
 7 - 9 + \underline{-2} & & \\
 7 - 9 - \underline{2} & & \\
 & & -2-2 \\
 & & -4
 \end{array}$$

NOT: Aynı öncelikteki işlemlerin uygulanmasının soldan sağa doğru yapıldığına dikkat ediniz.

Örnek 7. Aşağıdaki aritmetik ve mantıksal ifadelerde x ve y değerleri ondalık sayı değişkenleridir. Bu ifadelerin Java dilinde kodlamasını yapınız

(a)

$$\left| \frac{x-3}{2\pi y} \right|$$

Kodlanması: `Math.abs((x-3)/(2*Math.Pi*y))`

(b)

$$b \leq a \leq 2$$

Kodlanması: `(a >= b) && (a <= 2.0)`

Kendiniz Uygulayınız:

$$\frac{-B + \sqrt{B^2 - 4AC}}{2A}$$

şeklindeki matematiksel ifadenin Java dilinde

kodlanmasını yazınız?

Çoktan Seçmeli Değerlendirme Soruları

1. $a \leq b \leq 2$ şeklindeki matematiksel ifadenin Java dilinde kodlanması nasıl olur?

- a) $a \leq b \ \&\& \ b \geq 2$ b) $(a \leq b) \ \&\& \ (b \leq 2)$
c) $(b \geq a) \ \text{AND} \ (a \leq 2)$ d) $(a \leq b) \ || \ (b \leq 2)$

2. Aşağıdaki program parçasının ekran çıktısı ne olur?

```
int a = 4;  
System.out.println((a >> 1)+(a << 2) );
```

- a) 10 b) 4 c) 8 d) 18

3. Aşağıdaki program parçasının ekran çıktısı ne olur?

```
byte b = 44;  
int a = b++;  
long c = a--;  
System.out.print(c);
```

- a) 10 b) 44 c) 8 d) 18

4. Aşağıdaki program parçasının ekran çıktısı ne olur?

```
int x = 4;  
System.out.println(++x-x++-x--+x++);
```

- a) 0 b) -1 c) 8 d) 1

5. Aşağıdaki programın eşdeğeri olan kod satırı hangisidir?

```
Toplam = Toplam + i;  
i = i + 1;
```

- a) Toplam += i++; b) Toplam++ c) Toplam+=i; d) ++Toplam

6. $y=4x^2+2x+5$ şeklindeki matematiksel ifadenin Java dilinde kodlanması hangisidir?

- a) $y=4*x*x+2*x+5$ b) $y=4x^2+2x+5$ c) $y= 5$ d) $y=4x*x+2x+5$

7. $z=\frac{-5 \cdot a^{\frac{3}{2}}}{4}$ şeklindeki matematiksel ifadenin Java dilinde kodlanması hangisidir?

- a) $z= - (5*\text{Math.pow}(a,(2/3)) / 4)$ c) $z= - (5*a^{(3/2)}) / 4)$
b) $z= - (5*\text{Math.sqrt}(a,(2/3)) / 4)$ d) $z= - (5*a^{**}(2/3)) / 4)$

Kaynaklar

[“Bülent Çobanoğlu”, Java ile Programlama ve Veri Yapıları, 2. Baskı, ISBN:978-9944-711-10-4, Sayfa 5-21, 377-394](#)