عملگر ها در جاوا

عملگر انتساب

عملگرهای انتساب در جاوا برای اختصاص مقادیر به متغیرها استفاده می شوند. مثلا

```
int numreh;
numreh = 15;
```

عملگر انتساب مقدار سمت راست خود را به متغیر در سمت چپ خود اختصاص می دهد. در اینجا ، ۱۵ با استفاده از عملگر = به متغیر numreh اختصاص داده شده است.

مثال ۱: عملگر انتساب

```
1.  class AssignmentOperator {
2.  public static void main(String[] args) {
3.
4.  int number1, number2;
5.
6.  // Assigning 5 to number1
7.  number1 = 5;
8.  System.out.println(number1);
9.
10. // Assigning value of variable number2 to number1
11. number2 = number1;
12. System.out.println(number2);
13. }
14. }
```

خروجی

```
۵
۵
```

عملگرهای ریاضی

از عملگرهای ریاضی برای انجام عملیات ریاضی مانند جمع ، تفریق ، ضرب و غیره استفاده می شود.

عملگر	معنی
+	جمع(همچنین برای الحاق رشته ها هم استفاده می شود)
-	تفریق
*	ضرب
/	تقسیم
%	باقی مانده

مثال ۲: عملگر ریاضی

```
1.  class ArithmeticOperator {
2.  public static void main(String[] args) {
3.
4.  double number1 = 12.5, number2 = 3.5, result;
5.
6.  // Using addition operator
7.  result = number1 + number2;
8.  System.out.println(“number1 + number2 = ” + result);
9.
10. // Using subtraction operator
11. result = number1 – number2;
12. System.out.println(“number1 – number2 = ” + result);
13.
14. // Using multiplication operator
15. result = number1 * number2;
16. System.out.println(“number1 * number2 = ” + result);
17. // Using division operator
18. result = number1 / number2;
19. System.out.println(“number1 / number2 = ” + result);
20.
21. // Using remainder operator
22. result = number1 % number2;
23. System.out.println(“number1 % number2 = ” + result);
24. }
25. }
```

خروجی

```
number1 + number2 = 16.0

number1 – number2 = 9.0

number1 * number2 = 43.75

number1 / number2 = 3.5714285714285716

number1 % number2 = 2.0
```

در مثال بالا ، تمام عملوند های مورد استفاده متغیر هستند. می توان از اعداد هم استفاده کرد لزوما نباید متغیر باشند.

```
result = number1 + 5.2;

result = 2.3 + 4.5;

number2 = number1 -2.9;
```

همچنین می توان از عملگر + برای الحاق دو یا چند رشته استفاده کرد.

مثال ۳: عملگر ریاضی

```
1.  class ArithmeticOperator {
2.  public static void main(String[] args) {
3.
4.  String start, middle, end, result;
5.
6.  start = “Talk is cheap. “;
7.  middle = “Show me the code. “;
8.  end = “- Linus Torvalds”;
9.
10. result = start + middle + end;
11. System.out.println(result);
12. }
13. }
```

خروجی

Talk is cheap. Show me the code. – Linus Torvalds

عملگر های یگانی

عملیات را فقط روی یک عملوند انجام می دهند.

عملگر	معنی
+	مثبت یگانی(استفاده از آنها ضروری نیست زیرا اعداد بدون استفاده از آن مثبت هستند)
-	منهای یگانی (علامت منفی بودن عدد)
++	عملگر افزایشی (مقدار عملوند را یک واحد افزایش می دهد.)
--	عملگر کاهشی (مقدار عملوند را یک واحد کاهش می دهد.)
!	عملگر مکمل منطقی(مقدار boolean را معکوس می کند)

مثال ٤ : عملگر یگانی

```
1.  class UnaryOperator {
2.  public static void main(String[] args) {
3.
4.  double number = 5.2, resultNumber;
5.  boolean flag = false;
6.
7.  System.out.println(“+number = ” + +number);
8.  // number is equal to 5.2 here.
9.
10. System.out.println(“-number = ” + -number);
11. // number is equal to 5.2 here.
12.
13. // ++number is equivalent to number = number + 1
14. System.out.println(“number = ” + ++number);
15. // number is equal to 6.2 here.
16. // — number is equivalent to number = number – 1
17. System.out.println(“number = ” + –number);
18. // number is equal to 5.2 here.
19. System.out.println(“!flag = ” + !flag);
20. // flag is still false.
21. }
22. }
```

خروجی

```
+number = 5.2
-number = -5.2
number = 6.2
number = 5.2
!flag = true
```

همچنین می توانید از ++ و — به عنوان پیشوند و پسوند در جاوا استفاده کنید. عملگر ++ مقدار را ۱ واحد افزایش و — مقدار را ۱ واحد کاهش می دهد.

```
int myInt = 5;
++myInt // myInt becomes 6
myInt++ // myInt becomes 7
–myInt // myInt becomes 6
myInt– // myInt becomes 5
```

هنگام استفاده از عملگر افزایشی و کاهش به عنوان پیشوند و پسوند ، تفاوت اساسی وجود دارد. مثال زیر را در نظر بگیرید ،

```
1.  class UnaryOperator {
2.  public static void main(String[] args) {
3.
4.  double number = 5.2;
5.  System.out.println(number++);
6.  System.out.println(number);
7.  System.out.println(++number);
8.  System.out.println(number);
9.  }
10. }
```

خروجی

```
۵,۲
۶,۲
۷,۲
۷,۲
```

وقتی دستور

System.out.println(number++)

اجرا می شود ، ابتدا مقدار اصلی چاپ می شود و پس از آن یک واحد افزایش می یابد. به همین دلیل خروجی۵/۲ است.

سپس ، هنگامی که

System.out.println(number)

اجرا می شود، مقدار ۶/۲ چاپ می شود.

اما ، هنگامی که

System.out.println(++number)

اجرا می شود ، قبل از چاپ روی صفحه ، مقدار آن ۱ واحد افزایش می یابد.

برای – هم به همین شکل است.

عملگرهای برابری و رابطه ای

عملگرهای برابری و رابطه ای ، رابطه بین دو عملگر را تعیین می کنند. بررسی می کند که یک عملوند از دیگری بزرگ تر ، کوچک تر ، مساوی ، نا مساوی و ... است. بسته به رابطه ، نتیجه آن درست یا نادرست است.

عملگر	معنی	مثال	
==	برابر است با	5==3	False
!=	برابر نیست با	5!=3	True
>	بزرگ تر	5>3	True
<	کوچک تر	5<3	False
>=	بزرگ تر مساوی	5>=5	True
<=	کوچک تر مساوی	5<=5	True

عملگر های برابری و رابطه ای در تصمیم گیری و حلقه ها استفاده می شود.

مثال ٦ : عملگر های برابری و رابطه ای

```
1.  class RelationalOperator {
2.  public static void main(String[] args) {
3.
4.  int number1 = 5, number2 = 6;
5.  if (number1 > number2)
6.  {
7.  System.out.println(“number1 is greater than number2.”);
8.  }
9.  else
10. {
11. System.out.println(“number2 is greater than number1.”);
12. }
13. }
14. }
```

خروجی

```
number2 is greater than number1.
```

در اینجا ، ما از >برای بررسی اینکه آیا number1 از number2 بیشتر است یا خیر استفاده کردیم.

از آنجا که ، number2از number1 بیشتر است ، عبارت number1> number2 غلط ارزیابی می شود.

از این رو ، کد قسمت else اجرا می شود و در صورت درست بودن شرط ، کد درون بدنه ی if اجرا می شود.

به یاد داشته باشید که عملگرهای برابری و رابطه ای ، دو عملگر را با یکدیگر مقایسه می کنند و به صورت trueیا false ارزیابی می شوند.

علاوه بر عملگر های رابطه ای ، یک نمونه عملگر مقایسه ای نیز وجود دارد که نمونه ای از شی را با یک نوع خاص مقایسه می کند.

عملگر instanceof

در اینجا مثالی از عملگر instanceof آورده شده است.

```
1.  class instanceofOperator {
2.  public static void main(String[] args) {
3.
4.  String test = “asdf”;
5.  boolean result;
6.
7.  result = test instanceof String;
8.  System.out.println(result);
9.  }
10. }
```

وقتی برنامه را اجرا می کنید ، خروجی true خواهد بود. به این دلیل است که test نمونه ای از کلاس Stringاست.

عملگر های منطقی

عملگرهای منطقی ||(شرطی (OR -و) &&شرطی (AND-با عبارات بولی کار می کنند.

عملگر	توضیح	مثال	
||	شرطی OR ; هر یک از عبارات درست باشد، نتیجه درست است.	`false || true`	`true`
&&	شرطی AND ; هر دو عبارت باید درست باشند تا نتیجه درست باشد.	`false && true`	`false`

مثال ۸ : عملگر منطقی

```
1.  class LogicalOperator {
2.  public static void main(String[] args) {
3.
4.  int number1 = 1, number2 = 2, number3 = 9;
5.  boolean result;
6.
7.  // At least one expression needs to be true for result to be true
8.  result = (number1 > number2) || (number3 > number1);
9.  // result will be true because (number1 > number2) is true
10. System.out.println(result);
11.
12. // All expression must be true from result to be true
13. result = (number1 > number2) && (number3 > number1);
14. // result will be false because (number3 > number1) is false
15. System.out.println(result);
16. }
17. }
```

خروجی

```
true
false
```

عملگرهای منطقی در تصمیم گیری و حلقه استفاده می شوند.

عملگرهای سه گانه

عملگر شرطی یا عملگر سه گانه ?: کوتاه شده ی ساختار if-else است. نحو عملگر شرطی عبارت است از:

```
variable = Expression ? expression1 : expression2
```

در اینجا نحوه عملکرد آن آمده است.

- اگر Expression ، trueباشد مقدار expression1 به variable اختصاص می یابد.
- اگر Expression ، falseباشد مقدار expression2 به variable اختصاص می یابد.

مثال ۹ : عملگر سه گانه

```
1.  class ConditionalOperator {
2.  public static void main(String[] args) {
3.
4.  int februaryDays = 29;
5.  String result;
6.
7.  result = (februaryDays == 28) ? “Not a leap year” : “Leap year”;
8.  System.out.println(result);
9.  }
10. }
```

خروجی

```
Leap year
```

عملگر های Bitwise وBit Shift

برای انجام عملیات بیتی و شیفت بیتی در جاوا از این عملگرها استفاده می شود.

عملگر	تعریف
~	عملگر NOT
<<	شیفت چپ
>>	شیفت راست
>>>	شیفت به راست بدون علامت
&	عملگرANDبیتی
^	عملگر XOR بیتی
|	عملگر OR بیتی

این عملگرها معمولا مورد استفاده قرار نمی گیرند.

دیگر عملگرهای انتساب

ما فقط در مورد یک عملگر انتساب= در ابتدای آموزش صحبت کردیم. به جز این عملگر ، عملگر های انتساب دیگری هم وجود دارد که کمک می کند تا کد تمیز تری بنویسیم.

عملگر	مثال	برابر با
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x / 5
<<=	x <<= 5	x = x << 5
>>=	x >>= 5	x = x >> 5
&=	x &= 5	x = x & 5
^=	x ^= 5	x = x ^ 5
|=	x |= 5	x = x | 5