

# Introduction to Programming

Rania Baghernejad

Java – part 3

## Operators

### مباحث

- ۲ ..... عملگرهای ریاضی (Arithmetic Operators)
- ۵ ..... عملگرهای انتساب ترکیبی (Compound Assignment Operators)
- ۷ ..... عملگرهای مقایسه‌ای (Comparison Operators)
- ۹ ..... عملگرهای منطقی (Logical Operators)

## عملگرهای ریاضی (Arithmetic Operators)

### معرفی عملگرهای پایه

جاوا از عملگرهای استاندارد ریاضی برای انجام محاسبات استفاده می‌کند. این عملگرها شامل جمع، تفریق، ضرب، تقسیم و باقیمانده هستند:

عملگر	توضیح	مثال	نتیجه
+	جمع	$10 + 5$	15
-	تفریق	$10 - 5$	5
*	ضرب	$10 * 5$	50
/	تقسیم	$10 / 2$	5
%	باقیمانده تقسیم	$10 \% 3$	1

(در زبان جاوا عملگر توان نداریم، برای استفاده از توان از کتابخانه `Math` می‌توان استفاده نمود. `Math.pow(a, b)`)

### مثال ساده

دو عدد صحیح `a`, `b` را تعریف کنید و عملگرهای جمع، تفریق، ضرب، تقسیم و باقیمانده تقسیم `a` بر `b` را محاسبه کنید.

```
public class Main {  
    public static void main(String[] args) {  
        int a = 12;  
        int b = 5;  
  
        System.out.println("a + b = " + (a + b));  
        System.out.println("a - b = " + (a - b));  
        System.out.println("a * b = " + (a * b));  
        System.out.println("a / b = " + (a / b));  
        System.out.println("a % b = " + (a % b));  
    }  
}
```

خروجی:

```
a + b = 17  
a - b = 7  
a * b = 60  
a / b = 2  
a % b = 2
```

توجه: چون `a` و `b` از نوع `int` هستند، در عملیات تقسیم فقط بخش صحیح نتیجه حفظ می‌شود. اگر نیاز به نتیجه اعشاری دارید، باید از `double` استفاده کنید.

استفاده از نوع **double** در تقسیم

```
public class Main {  
    public static void main(String[] args) {  
        double x = 12;  
        double y = 5;  
  
        System.out.println("x / y = " + (x / y));  
    }  
}
```

خروجی:

```
x / y = 2.4
```

## تقدیم عملگرها (Operator Precedence)

جاوا مانند ریاضیات، برخی عملیات‌ها را زودتر از بقیه اجرا می‌کند. ترتیب اولویت اجرای عملگرها به صورت زیر است:

۱. پرانتزها ()
۲. ضرب، تقسیم، باقیمانده % / \*
۳. جمع و تفریق - +

مثال:

```
int result = 10 + 5 * 2;  
System.out.println(result); // خروجی 20
```

چون ابتدا ضرب انجام می‌شود ( $5 * 2 = 10$ ) و سپس با  $10 +$  جمع می‌شود.

اگر بخواهیم ابتدا جمع انجام شود:

```
int result = (10 + 5) * 2;  
System.out.println(result); // خروجی 30
```

## افزایش یا کاهش

دو عملگر مهم دیگر برای افزایش یا کاهش مقدار متغیرها استفاده می‌شوند:

عملگر	کاربرد	مثال	نتیجه
++	افزایش مقدار متغیر به اندازه ۱	++a یا a++	a = a + 1
--	کاهش مقدار متغیر به اندازه ۱	--a یا a--	a = a - 1

:مثال

```
int a = 5;
a++;
System.out.println(a); // خروجی 6
```

```
int b = 8;
b--;
System.out.println(b); // خروجی 7
```

---

:نکته تكميلی

در تقسيم دو عدد صحيح بر هم بهتر است برای گرفتن خروجی دقيق پيش از آن نوع داده تقسيم را داخل پرانتز مشخص کرده:

```
int a = 5;
int b = 2;

double q = (double) a/b
```

## عملگرهای انتساب ترکیبی (Compound Assignment Operators)

### معرفی

این عملگرها برای انجام یک عملیات ریاضی روی متغیر و سپس اختصاص دادن نتیجه به همان متغیر استفاده می‌شوند. به این ترتیب، کد ساده‌تر و خواناتر می‌شود.

عملگر	معادل کامل	توضیح
$+=$	$a = a + b$	افزودن مقدار $b$ به $a$
$-=$	$a = a - b$	کم کردن مقدار $b$ از $a$
$*=$	$a = a * b$	ضرب کردن $a$ در $b$
$/=$	$a = a / b$	تقسیم $b$ بر $a$
$\%=$	$a = a \% b$	باقیمانده تقسیم $b$ بر $a$

### مثال کاربردی با `int`

```
public class Main {
    public static void main(String[] args) {
        int a = 10;

        a += 5;
        System.out.println("a += 5 → " + a); // Output: 15

        a -= 3;
        System.out.println("a -= 3 → " + a); // Output: 12

        a *= 2;
        System.out.println("a *= 2 → " + a); // Output: 24

        a /= 4;
        System.out.println("a /= 4 → " + a); // Output: 6

        a %= 5;
        System.out.println("a %= 5 → " + a); // Output: 1
    }
}
```

---

### استفاده با **:double**:

```
public class Main {  
    public static void main(String[] args) {  
        double x = 10.5;  
  
        x += 2.5;  
        System.out.println("x += 2.5 → " + x); // Output: 13.0  
  
        x *= 2;  
        System.out.println("x *= 2 → " + x); // Output: 26.0  
  
        x /= 5;  
        System.out.println("x /= 5 → " + x); // Output: 5.2  
    }  
}
```

---

### نکات مهم:

- این عملگرها به صورت خلاصه‌سازی شده برای نوشتن محاسبات عددی استفاده می‌شوند.
- در برنامه‌نویسی حرفه‌ای، استفاده از این عملگرها رایج است زیرا باعث افزایش خوانایی کد می‌شود.
- این عملگرها با انواع داده‌ی عددی مانند short, int, float, double, long سازگار هستند.

## عملگرهای مقایسه‌ای (Comparison Operators)

### معرفی

عملگرهای مقایسه‌ای برای مقایسه دو مقدار استفاده می‌شوند. نتیجه‌ی این مقایسه‌ها همیشه یک مقدار بولی (true, false) خواهد بود.

عملگر	معنا	مثال	نتیجه
==	برابر است با	$5 == 5$	true
!=	نابرابر است با	$4 != 3$	true
>	بزرگ‌تر از	$7 > 2$	true
<	کوچک‌تر از	$2 < 7$	true
>=	بزرگ‌تر یا مساوی	$5 >= 5$	true
<=	کوچک‌تر یا مساوی	$3 <= 4$	true

مثال عملی:

```
public class Main {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;

        System.out.println("a == b → " + (a == b)); // false
        System.out.println("a != b → " + (a != b)); // true
        System.out.println("a > b → " + (a > b)); // false
        System.out.println("a < b → " + (a < b)); // true
        System.out.println("a >= 10 → " + (a >= 10)); // true
        System.out.println("b <= 15 → " + (b <= 15)); // false
    }
}
```

نکته بسیار مهم: عدم پشتیبانی از مقایسه‌ی زنجیره‌ای

برخلاف زبان‌هایی مثل پایتون، در جاوا نمی‌توان چند مقایسه را به صورت زنجیره‌ای نوشت. عبارت زیر غلط است و منجر به خطای نتیجه نادرست خواهد شد:

```
// wrong, this doesn't work in java
if (3 < x < 10) {
    // ...
}
```

در جاوا، باید این گونه نوشته شود:

```
if (x > 3 && x < 10) {
    // correct!
}
```

چرا؟

- عبارت  $x < 3$  ابتدا  $x < 3$  را بررسی می‌کند، که خروجی آن `true` یا `false` است.
- سپس تلاش می‌شود این مقدار بولی با 10 مقایسه شود که منطقی نیست و باعث خطا یا نتیجه‌ی غیرمنتظره می‌شود.
- به جای آن باید از عملگرهای منطقی مانند `&&` برای ترکیب چند شرط استفاده کرد.

---

### کاربرد با انواع مختلف

عملگرهای مقایسه‌ای فقط به نوع `int` محدود نیستند. می‌توان از آن‌ها برای مقایسه‌ی انواع دیگر عددی، (`double`, `float`, `long`, ...) و حتی کاراکترها (`char`) استفاده کرد:

```
public class Main {
    public static void main(String[] args) {
        double x = 5.5;
        System.out.println(x >= 5.0); // true

        char c = 'A';
        System.out.println(c < 'Z'); // true
    }
}
```

## عملگرهای منطقی (Logical Operators)

در زبان برنامه‌نویسی جاوا برای ترکیب چند شرط استفاده می‌شوند. نتیجه‌ی نهایی این ترکیب‌ها نیز همیشه مقداری از نوع بولی (true, false) است.

در جدول زیر، عملگرهای منطقی اصلی جاوا را همراه با کاربرد، مثال و توضیح آورده‌ام:

جدول عملگرهای منطقی در جاوا

عملگر	نام	معنا	مثال	نتیجه
&&	AND	و (هر دو شرط باید درست باشند)	$x > 5 \&\& x < 10$	فقط اگر هر دو شرط درست باشند true
	OR	یا (حداقل یکی از شرط‌ها درست باشد)	$x > 5    x < 10$	اگر حداقل یکی از شرط‌ها درست باشد true
!	NOT	نقیض (معکوس کردن مقدار بولی)	$!(x > 5)$	اگر شرط درست باشد، خروجی false می‌شود

مثال کامل:

```
public class Main {
    public static void main(String[] args) {
        int x = 8;

        // AND
        System.out.println(x > 5 && x < 10); // true

        // OR
        System.out.println(x < 5 || x > 10); // false

        // NOT
        System.out.println(!(x == 8)); // false
    }
}
```

نکات مهم:

۱. && فقط زمانی خروجی true می‌دهد که هر دو شرط درست باشند.

۲. || اگر حتی یک شرط درست باشد، خروجی true خواهد بود.

۳. ! مقدار بولی را معکوس می‌کند !true می‌شود false و برعکس.

۴. ترکیب این عملگرها برای ساختن شرط‌های پیچیده‌تر بسیار کاربردی است.

---

مثال ترکیبی پیچیده‌تر:

```
int age = 22;
boolean hasID = true;

if (age >= 18 && hasID) {
    System.out.println("Access granted.");
} else {
    System.out.println("Access denied.");
}
```

در اینجا شرط بررسی می‌کند آیا هر دو مورد (بالای ۱۸ سال بودن و داشتن کارت شناسایی) برقرار است یا نه. (با دستورات

شرطی در بخش بعدی آشنا می‌شویم.