

## Precedencia (precedence)

Ha többféle operátor van egy kifejezésben, akkor nem mindegy, hogy milyen sorrendben hajtjuk őket végre.

Pl.

$2 + 3 * 5$ , ha először a  $3*5$ -öt hajtjuk végre (ahogy matematikailag helyes), akkor  $2 + 15 = 17$ -et kapunk.

Ha a  $2+3$ -at hajtjuk végre előbb (matematikailag helytelenül), akkor  $5 * 5 = 25$  lesz az eredmény.

Bizonyára emlékszünk matekóráról, hogy először a zárójelben lévő műveleteket hajtjuk végre, utána a hatványozást, utána a szorzást és osztást, legvégül az összeadást és kivonást.

A matematikai operátorok végrehajtási sorrendje (ún. precedenciája (mi előz meg mit)):

- zárójel
- hatványozás
- szorzás, osztás
- összeadás, kivonás

A Javában is vannak operátorok és **szintek**, csak sokkal több, mint a matekban.

Egy másik fontos szempont, hogy merről merre hajtsuk végre a műveleteket: **balról jobbra vagy jobbról balra**? Vegyük pl. a  $120 / 6 / 2$  kifejezést. Ha előbb a  $120 / 6$ -ot hajtjuk végre (ahogy matematikailag helyes), akkor  $20 / 2 = 10$  az eredmény. Ha fordítva (előbb a  $6 / 2$ -t, akkor  $40$  az eredmény).

Ezért a Java precedencia-táblázata két dolgot határoz meg:

- **melyik operátort kell előbb végrehajtani?** (előbb szorzás, utána összeadás)
- **ha két azonos szinten lévő operátor is van egy kifejezésben, melyik jön előbb** (két összeadás közül előbb a bal oldali jön)

Miért fontos ismerni a precedencia-táblázatot?

1. el tudjuk dönteni egy kifejezés helyes eredményét
2. fölösleges zárójelezéstől (és áttekinthetatlenségtől) kíméljük meg magunkat és a programunkat.

Végül a Java precedencia-táblázata (megtalálod a Google-be beírva, hogy „java operator precedence”)

| Operator  | Description   | Level | Associativity |
|---|---|-------|---------------|
| [ ]<br>.<br>( )<br>++<br>--                     | access array element<br>access object member<br>invoke a method<br>post-increment<br>post-decrement | 1     | left to right |
| ++<br>--<br>+<br>-<br>!<br>~                    | pre-increment<br>pre-decrement<br>unary plus<br>unary minus<br>logical NOT<br>bitwise NOT           | 2     | right to left |
| ()<br>new                                       | cast<br>object creation   | 3     | right to left |
| *<br>/<br>%                                     | multiplicative  | 4     | left to right |
| + -<br>+  | additive<br>string concatenation  | 5     | left to right |
| << >><br>>>>                                    | shift   | 6     | left to right |
| < <=<br>> >=<br><br>instanceof                  | relational<br>type comparison   | 7     | left to right |
| ==<br>!=  | equality  | 8     | left to right |
| &   | bitwise AND   | 9     | left to right |
| ^   | bitwise XOR   | 10    | left to right |
|   | bitwise OR  | 11    | left to right |
| &&  | conditional AND   | 12    | left to right |
|   | conditional OR  | 13    | left to right |
| ?:  | conditional   | 14    | right to left |
| = += -=<br>*= /= %=<br>&= ^=  =<br><<= >>= >>>= | assignment  | 15    | right to left |

1. ábra <http://introcs.cs.princeton.edu/java/11precedence/>