

Kifejezések és függvények

Kifejezések (ismétlés)

Ha vannak operátoraink (legalább aritmetikaiak – operator, arithmetic operator), változóink (variable) és literáljaink (literal), akkor ezekből ún. *kifejezéseket* (expression) alkothatunk.

Példák:

1. $2 + 4$
2. $5 * a$
3. $6 * a * (b + 4.0)$
4. $a \% (6 + x)$

Minden *kifejezésnek* van egy **értéke** (value), ami a végrehajtásakor kiszámítható, azaz a fenti esetben:

1. 6
2. amennyiben a értéke éppen 2, mikor végrehajtjuk ($a = 2$), akkor 10 a kifejezés eredménye.
3. $a = 3$ és $b = 2$ esetén: 108.0
4. $a = 16$ és $x = 3$ esetén: 7.

A kifejezéseknek van **típusa** (type) is, mely a szereplő összetevők típusaitól függ, az alábbiak szerint:

Vegyük példának a $6 * a * (b + 4.0)$ műveletet. A számítógép az alábbi lépésekben értékeli ki ezt a kifejezést:

1. kezdjük a zárójellel, azt kell leghamarabb elvégezni.
2. a zárójelben van egy összeadás, $b + 4.0$. b egész, 4.0 double literál. b értéke 2.
 1. a b értékét double-lé konvertálok (nem a változót, csak a benne lévő értéket!)
 2. majd összeadom a 4.0-val.
 3. A részeredmény 6.0.
3. A maradék kifejezés: $6 * a * 6.0$.
4. A szorzást balról jobbra haladva kell elvégezni
5. Az első művelet a $6 * a$ (a értéke 3). Mindkettő int, ezért az eredmény típusa is int.
6. A részeredmény $6 * 3 = 18$.
7. A maradék kifejezés: $18 * 6.0$.
8. A 18 egész érték, a 6.0 double érték.
9. A 18-at double-lé konvertálok. Eredmény: 18.0.
10. Elvégzem a szorzást: 108.0. Típus: double.

11. Készen vagyok.

Ha a kifejezés egy értékadás művelet jobb oldalán helyezkedik el, akkor előbb kiszámolja a kifejezés értékét, majd elvégzi az értékadást.

```
z = 6 * a * (b + 4.0)
```

A fenti lépések elvégzése után marad a következő részkifejezés:

```
z = 108.0;
```

Ez pedig egy értékadás, amellyel megváltoztatja a `z` változó értékét.

Megjegyzés. Az értékadás művelet is kifejezés és értéke is van. a `z = 108.0` értékadás értéke `108.0` (az átadott érték). Ha kiíratjuk azt, hogy `System.out.println(z = 108.0);` akkor az az értékadás értékét írja ki, a képernyőre `108.0` fog kerülni.

Függvények (function)

Annak idején, talán általános vagy középiskolában bizonyosan tanultál már a függvényekről. Pl. *abszolútérték függvény*, ami visszaadja a számot, ha az pozitív, és a -1 -szeresét, ha negatív ($-2 \rightarrow +2$; $+3 \rightarrow +3$). Javában is vannak ilyesmi függvények, pl. az előbb említett abszolútérték használata:

```
w = Math.abs(-2);
```

Tehát maga a kifejezés úgy értendő, hogy vesszük a -2 abszolút-értékét (ez a `Math.abs(-2)`, aminek értéke $+2$), amit átadunk a `w` változónak. A függvények készítésével részletesebben foglalkozunk majd a tanfolyam későbbi részében, addig is egy másik példa:

```
x = Math.ceil(2.3);
```

ami 3 -at ad vissza, mivel a `Math.ceil` a felső egészrészt jelenti (azaz felfelé kerekítést). Az ellentettje a `Math.floor()` (lefelé kerekítést végez).

Hatványozás?

A legtöbb általános iskolában használt műveleti jelet meg fogjuk találni a Java operátorok között. Van $+$, $-$, $*$ (szorzás), $/$ (osztás), viszont nem fogjuk megtalálni a hatványozás műveletet. Van ugyan $^$ operátor, de az nem a hatványozás, hanem az ún. „**kizáró vagy**” művelet (nem te vagy kizáró, hanem a művelet), ami a bitenkénti műveletek egyike, de nem kell tudni, mert nagyon ritkán használják Java-programozásban. A hatványozás egyébként függvénnyel megy:

`Math.pow(2, 4)` pl. a 2^4 -t jelenti.

Kérdés

- Mikor deklarálók egy `d double` változót és azt akarom az értékének adni, hogy $9/5$, akkor `1.0` lesz az eredmény (`double d = 9/5`). Viszont amikor úgy deklarálók két `double` változót, hogy az egyiknek az értéke `9` a másiknak pedig `5`, és ezt egy harmadik `double` változóban elosztom, akkor ott az eredmény már `1,8` (`double a = 9; double b = 5; double c = a/b;`). Erről tudnál egy kis magyarázattal szolgálni?

A műveletek végrehajtási sorrendje miatt először az osztás hajtódik végre, utána az értékadás. Az osztás eredménye $9/5$ esetén `1`, típusa egész. Ezt az egészet utána betesszük a `double` változóba és szélesítéssel `double`-lé konvertálja.

A másik esetben két `double`-t osztunk el, az eredmény ezért `1.8`, típusa `double` lesz. Utána beleírjuk az eredményváltozóba.