Függvények

A függvények névvel rendelkező utasításcsoportok, melyeknek információkat adhatunk át, és van egy visszatérési értékük. Mint egy dzsinn:

Hogyan is "használunk" egy dzsinnt?

- megszólítjuk
- megmondjuk, hogy mit akarunk tőle
- megvárjuk a választ

Képzeljük el, hogy a csodalámpából egy selejtes dzsinn bújik elő, aki csak egyetlen dolgot tud csinálni, mondjuk meghatározni egy szám négyzetgyökét. Játsszuk le a beszélgetést:

- Ó, Hatalmas Négyzetgyökszámoló Dzsinn!
- Miért zavartál fel évezredes álmomból, halandó?
- Mondd meg kérlek, mennyi 5 négyzetgyöke?
- 2,23606797749978969640917366873...

Ha a programozási függvényeket nézzük, nekik is van:

- nevük (négyzetgyökszámítás, Javásan: `Math.sqrt()`)
- paraméterük vagy paramétereik, azaz adatok, amit kellenek a működéshez (a szám, aminek a négyzetgyökét vettük)
- visszatérési értékük (a 2,23606...)

Függvény meghívása

Összegezve Javásan: `double gyok = Math.sqrt(5);`

Lásd még: `int x = sc.nextInt();`, ahol az `nextInt()` a függvény, aminek nincs paramétere, az `sc.` pedig azt mondja meg, hogy ez az `sc` nevű `Scanner` objektumhoz tartozik.

Függvény írása

A függvényesdiben az az igazán nagyszerű, ha mi is tudunk készíteni függvényeket. Fontos, hogy ezt a részt az osztályon belülre, de a `main()` függvényen kívülre tegye az ember. Javában a függvények egymás testvérei. A függvények az osztályon belül bármilyen sorrendben lehetnek.

```
public static double negyzetgyok(double param) {
   // megvalósítás
}
```

Nézzük az egyes részeket:

- `public static`: később megmagyarázom
- az első `double`: a függvény visszatérési értékének típusa
- `negyzetgyok`: a függvény neve
- `double param`: a függvénye paramétere. Itt most egy van, ha több lenne, vesszővel kellene elválasztani.

A négyzetgyök kiszámítása első feladatnak egy kicsit meredek lenne, úgyhogy a példa kedvéért legyen egy három szám közül a legnagyobbat kiszámító függvény írása a feladat:

```
public static int max(int a, int b, int c) {
   int m = a;
   if (b > m) {
      m = b;
   }
   if (c > m) {
      m = c;
   }
   return m;
}
```

A `return m;` két dolgot csinál:

- beállítja a függvény visszatérési értékét
- kilép a függvényből

Hívjuk meg az elkészült függvényünket:

```
int f = 5;
int g = 8;
int z = 2;
int mx = max(f, g, z);
System.out.println("A legnagyobb: "+mx);
```

A függvény hívásakor rendre behelyettesítjük az `f`, `g` és `z` változók **értékeit**.

> *A függvény meghívásakor a változó értéke **lemásolódik**, tehát ha megváltoztatjuk a függvényben az értékét, akkor csak a **másolat** értéke változik meg.*

Lefut a függvény, a visszatérési értéke (`return` után megadott `m` értéke) bemásolódik az mx változóba, mintha ezt írnánk le: `int mx = 8;`

> *A függvény paramétereinek neve `a`, `b`, `c` nem kell, hogy megegyezzen a hívásban használt változókkal (`f`, `g`, `z`), de lehetnek azonosak is.*

> *A függvényeken belül létrehozott ún. lokális változókat egy másik függvényből csak úgy érhetjük el, ha az visszatérési értéke a függvénynek.*

Másolás következményei

Ez **hibás**:

```
public static void hibasCsere(int a, int b) {
   int c = a;
   a = b;
   b = c;
}
```

majd a `main()`-ben:

```
int x = 7;
int y = 8;
hibasCsere(x, y);
System.out.println("x="+x+"; y="+y);
```

Próbáld ki! :-)

void

Visszatérési értékként használhatjuk a `void` kulcsszót is, amely azt jelenti, hogy az adott függvénynek nincs visszatérési értéke.

`void` függvények esetében is használható a `return`, de csak így: `return;`, érték nélkül.

Megjegyzések

- két függvénynek lehet azonos neve is, ekkor a paraméterek száma, típusa eltérő kell legyen. (A számítógépnek meg kell tudni különböztetni, hogy melyikre gondoltunk)
- A Javás objektum-orientált terminológiában találkozhatunk a *metódus* fogalommal is. Itt most - az objektum-orientált programozás előtt - egyelőre függvénynek nevezem őket, adózva a struktúrált programozás és a C programozás nagyjainak.
- A `return` után állhat kifejezés is, nem csak változó, pl. `return a+b+c;`

Végezd el a 1-5. feladatokat!