# 4. Metódusok

Először tisztázzunk két fogalmat:

**Objektum**: Az objektum a valódi világ egy modellezett eleme, amely bizonyos tulajdonságokkal rendelkezik, és valamilyen tevékenységeket tud végezni.

**Osztály**: Megírt kódok, amelyek az objektumok tervrajzai, benne a tulajdonságokkal és a tevékenységekkel. Az objektumokat ezekből hozzuk létre, ezeknek a példányai.

Ilyen lehet például egy Kutya osztály, a kutyák tulajdonságaival (név, fajta, szín, magasság, stb.), valamit a rájuk jellemző tevékenységekkel (evés, ivás, ugatás).

A Kutya osztály példányai Rex, Kántor, Bodri, ...

## Metódusok hívása

A metódusok valamilyen tevékenységet végző programrészek, amelyek még egy értéket is előállíthatnak.

A metódus akkor futnak le, ha a program egy pontjáról meghívjuk őket. Ehhez általában három dolgot kell megadnunk:

* az objektumot vagy osztályt, amely a metódust tartalmazza,
* a **metódus** nevét,
* és a **paramétereket**, ha vannak.

Az objektumot vagy az osztályt akkor kötelező megadni, ha nem az az objektum vagy osztály, amelyből a metódust meghívjuk.

Minden esetben kötelező megadni a metódus nevét. A metódusok neve adott osztályon belül egyedi. Valójában a metódus neve, a bemenő adatok típusai és száma együttesen kell hogy egyedi legyen. Az ezzel kapcsolatos úgynevezett túlterhelésről majd később lesz szó.

Ha a metódusnak szükséges, akkor a feladattal kapcsolatos adatokat, a paramétereket is meg kell adni. Természetesen lehet olyan metódus is, amely nem használ bemenő adatokat.

Lássunk pár példát olyan metódusokra, melyeknél az osztályt is meg kell adni:

Math.min(23,b);

Két bemenő adat, egy szám és egy számot tartalmazó változó közül a kisebb értéket adja eredményül.

Math.sqrt(18);

Kiszámítja 18 négyzetgyökét.

System.out.println("Ez egy rövid mondat.");

Kiír egy szöveget a képernyőre.

A Java programnyelv metódusait két csoportba sorolhatjuk:

1. olyan metódusok, melyek valamilyen értéket állítanak elő,
2. olyan metódusok, melyek valamilyen tevékenységet hajtanak végre.

Az első csoport nagyjából olyan, mint egy táblázatkezelő program függvényei. Ezek a metódusok valamilyen bemenő adatok segítségével számítást végeznek, és adott típusú (szám, szöveg, karakter, stb.) eredményt állítanak elő. Ez az eredmény a metódus visszatérési értéke. A bemenő adatok, amelyeket paramétereknek nevezünk, valamilyen változók vagy literálok.

A második csoportba tartozó metódusok valamilyen tevékenységet hajtanak végre, ide tartozik például a képernyőre való kiíratás.

Ez a lecke elsődlegesen az előre megírt metódusokról szól, a saját magad által megírt metódusok egy másik lecke része lesz. A kész metódusokat tehát formailag a következőképp lehet használni:

Osztalynev.metodus();

Osztalynev.metodus(paraméter);

Osztalynev.metodus(paraméter1, paraméter2, …);

Ahogy láthatod, először hivatkozni kell arra az osztályra, amelynek a metódusát használni szeretnéd, azután a . operátor használatával hivatkozunk az osztály adott nevű metódusára. Ez az a két dolog, ami minden esetben kötelező. A harmadik dolog, hogy megadd a metódusnak azokat az adatokat amelyeket használnia kell. Vagy azért, hogy egy eredményt állítson elő (1. típus), vagy azért, hogy annak segítségével hajtsa végre a kért tevékenységet.

Minek kell az osztálynév, ha a metódusoknak neve van? Azért, mert létezhet több azonos nevű metódus, (sőt sokszor ez a helyzet), ezért meg kell mondani, hogy ez a metódus melyik osztályba tartozik.

## A Math osztály

A Math osztály a matematikával kapcsolatos metódusokat tartalmaz. A metódusok többsége sokféle számot is elfogad, beleértve egész és valós értékeket is, azok összes altípusával együtt. Sőt, ezeket keverve is működik.

A Math osztályt nem kell, sőt nem is lehet példányosítani. A metódusokat az osztály nevével kell meghívni.

Néhány fontosabb metódus a Math osztályból:

Math.min(a,b); // a két változó közül a kisebb értéket adja eredményül

Math.round(a); // matematikai szabály szerint kerekíti a változó értékét

Math.abs(a); // az adott változó értékének abszolút értékét adja vissza

Math.sqrt(a); // az adott változó négyzetgyökét adja vissza

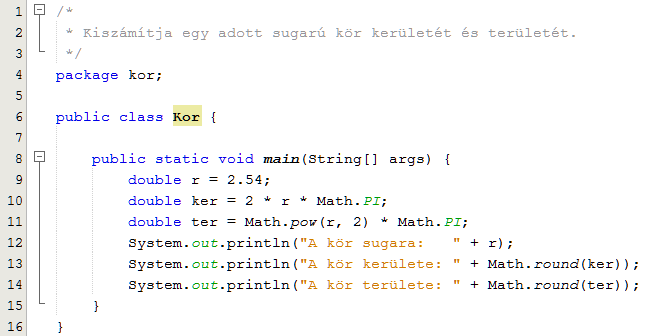
Math.pow(a,b); // az a számot a b-edik hatványra emeli

Math.PI; // Pi értékét adja vissza

Az utolsó kakukktojás. Ez nem metódus, hanem állandó, vagyis olyan változó, amelynek fix értéke van. Ezért nem teszünk utána zárójeleket.

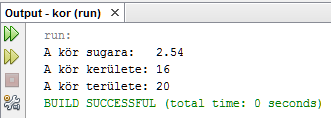
## Kör kerülete és területe

Készítsünk egy programot, amely kiszámítja egy adott sugarú kör kerületét és területét! (K = 2r, T = r2)



A kör sugara most 2,54. A 10. és 11. sorban a pi értékét a Math.PI állandó segítségével határoztuk meg. A 11. sorban a hatványozáshoz a Math.pow() metódust használtuk. A 13-14. sorban a kerekítést a Math.round() metódussal végeztük.

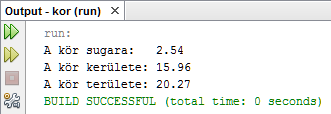
Kezd egy új Java projektet kor néven, és készítsd el a programot! Futtasd a programot! A kimenet ilyen lesz:



Szeretnénk az eredményeket egy kicsit pontosabban, mondjuk két tizedessel kiíratni. A Math.round() metódusban azonban ezt nem lehet megadni. A megoldás az, hogy az eredményt megszorozzuk százzal, kerekítünk, majd osztunk százzal. Arra kell figyelni, hogy hogy az osztásnál 100.0-val osszunk, hogy az eredmény double típusú legyen. Alakítsd át a 13-14-es sorokat így:



Futtasd a programot, és nézd meg az eredményt:



Feladat: Alakítsd át a programot úgy, hogy az eredmények három tizedes jeggyel jelenjenek meg!

Próbáld ki más sugár értékekkel is a programot!

Elég kényelmetlen megoldás, hogy a sugár értékét csak a program kódjában tudjuk módosítani. Legközelebb megtanuljuk, hogyan lehet a billentyűzetről beolvasni a felhasználó által megadott adatokat, és ezekkel számolni a programokban. Így programjaink különböző értékekkel is használhatók lesznek.

## Kérdések

Mi lesz a következő kifejezések értéke, ha x egy double típusú változó, és értéke 6,23?  
a. Math.min(x,5);  
b. Math.round(x\*10)/10.0;  
c. Math.abs(x);  
d. Math.sqrt(x);  
e. Math.pow(x,3);