

Elágazás (if)

Mire jók a logikai kifejezések?

A program végrehajtása az európai könyvolvasási sorrend szerinti módon zajlik... amíg bele nem nyúlunk. Az if utasítás segítségével pont ilyen "belenyúlást" tudunk végrehajtani.

Az if utasítással meg tudunk jelölni egy részt a programban, amit vagy végrehajtunk, vagy nem. Attól függően, hogy a logikai kifejezés igaz-e vagy hamis. (Ez az egyik felhasználási módja. A másik ld. később).

```
if (logikai kifejezés) {
    utasítások
}
```

Ha a logikai kifejezés pillanatnyilag pont igaz, akkor az utasítások végrehajtódnak, ellenkező esetben nem.

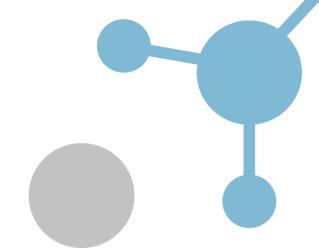
Példa:

```
if (5 <= age && age <= 10) {
    System.out.println("Az életkor 5 és 10 közötti");
}</pre>
```

A másik lehetőség, hogy választunk két kódrészlet között, ha a logikai kifejezés igaz, akkor az egyik, ha hamis, a másik ágat (branch) hajtjuk végre.

```
if (5 <= age && age <= 10) {
    System.out.println("Az életkor 5 és 10 közötti");
} else {
    System.out.println("Az életkor nem 5 és 10 közötti");
}</pre>
```







Az if-es szerkezetek korlátlanul egymásba ágyazhatók:

```
if (5 <= age && age <= 10) {
    System.out.println("Az életkor 5 és 10 közötti");
} else {
    if (age < 5) {
        System.out.println("Az életkor 5-nél kisebb");
    } else {
        System.out.println("Az életkor 10-nél nagyobb");
    }
}</pre>
```

Minél több ilyen ágat ágyazunk egymásba, annál nehezebben érthető lesz a szerkezet. Pedig tulajdonképpen egymást kizáró feltételeket fogalmazunk meg: Egy szám vagy 5-nél kisebb, vagy 5 és 10 közötti, vagy 10-nél nagyobb. És ezek közül az egyik biztosan, de nem lehet egynél több kategória tagja. A feltételek részekre osztják az összes lehetséges esetet, azaz "partícionálják".

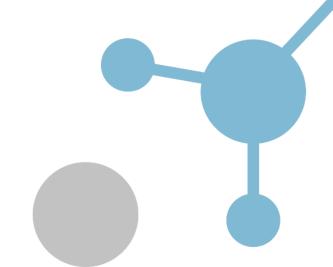
A partícionálás kifejezés egy adott dolog teljes és átfedésmentes felosztását jelenti. Azaz anyuka a tortát "partícionálja" ②. Ahonnan még ismerős lehet: ugyanígy hívják azt is, amikor a háttértárat különböző meghajtókra osztja az ember (hogy legyen egy meghajtó, amin a Windows van, meg egy másik, amin az adatok – hogy ha a Windows megsérül, nyugodtan lehessen formázni és újratelepíteni anélkül, hogy az adatok elvesznének.

Elárulom, hogy az if ágaiban, ha egyetlen utasítás kap helyet, akkor a kapcsos zárójelek (" $\{$ " és " $\}$ ") elhagyhatók.

Nézzük a fenti kódot, és hagyjuk el piros, vastag kapcsos zárójelet:

```
if (5 <= age && age <= 10) {
    System.out.println("Az életkor 5 és 10 közötti");
} else
    if (age < 5) {
        System.out.println("Az életkor 5-nél kisebb");
    } else {
        System.out.println("Az életkor 10-nél nagyobb");
    }</pre>
```







Rendezzük át a kapott kódot:

```
if (5 <= age && age <= 10) {
    System.out.println("Az életkor 5 és 10 közötti");
} else if (age < 5) {
    System.out.println("Az életkor 5-nél kisebb");
} else {
    System.out.println("Az életkor 10-nél nagyobb");
}</pre>
```

Sokkal áttekinthetőbb, nem?

Bár általában javasolni szoktam, hogy minden kapcsos zárójelet tegyünk ki, ebben az esetben tegyünk kivételt.

Általánosságban azért szoktam ilyesmiket javasolni, hogy "szép, tiszta" legyen a programunk, ami azt jelenti, hogy ha megmutatjuk egy másik programozónak (pl. a mentorodnak), akkor azonnal értsen belőle mindent. Ezért vesszük a konvenciókat, irányelveket. Van egyébként erről egy könyv is, Robert C. Martin: Tiszta kód (Clean Code).

A kapcsos zárójelekről

Ebben és a következő fejezetekben több olyan utasítással ismerkedünk majd meg, amiben a kapcsos zárójelek használata általában szükséges, de mindenképpen ajánlott.

A kapcsos zárójelek ún. *utasításblokk*okat határolnak. Olyannyira határolnak, hogy az utasításblokkon belül létrehozott változók a blokk végén **megszűnnek**. Mint itt a width (szélesség) változó:

```
if (length < 100) {
   int width = length / 2;
   System.out.println(width); // width létezik
}
// width már nem létezik</pre>
```

Képzeljük el az egészet úgy, hogy a kapcsos zárójelek **ketrecbe zárják** a benne lévő utasításokat és ez a ketrec a belül létrehozott változókat nem engedi ki: [©]



