# 03. Tömbök 2.

## Tömbök mérete

A múltkori programban beírtuk a for ciklusba a tömb méretét. Ez azonban nem túl jó megoldás, mert ha módosítunk a tömb méretén, a for ciklusok határait is át kell írnunk.

Sokkal elegánsabb, ha lekérdezzük a tömb méretét, és így a programok módosítása is egyszerűbb.

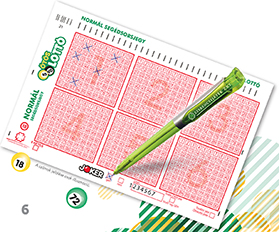
A Javaban minden tömbnek van egy length nevű változója, amely a tömb méretét, vagyis elemeinek számát tartalmazza.

Például:

int t[] = new t[10];  
for (int i = 0; i < t.length; i++)  
 System.out.println(t[i]);

Ez nagyon hasonlít a stringek length() metódusára, de itt nem teszünk zárójeleket utána, mert ez nem metódus!

## Lottószámok



Szeretnénk az ötös lottón játszani, de nincs ötletünk, hogy milyen számokat jelöljünk be. Most egy olyan programot fogunk írni, amely segít nekünk ebben. Ahányszor lefuttatjuk, mindig kiír öt darab 1 és 90 közötti véletlen számot, növekvő sorrendbe rendezve.

Első nekifutásra megpróbálhatnánk úgy megoldani a feladatot, hogy egy ciklussal előállítunk 5 darab 1 és 90 közötti véletlen egész számot, és kiírjuk őket. Így azonban két probléma is adódna:

1. Előfordulhatna, hogy kétszer (vagy többször) ugyanazt a számot kapjuk.
2. Utána növekvő sorrendbe kellene rendezni őket.

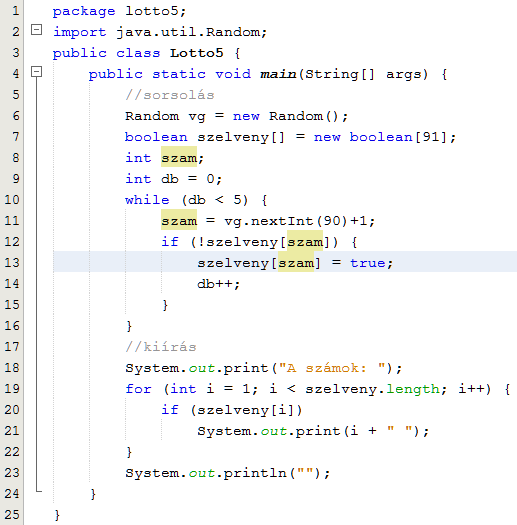
Ezért most más módszert választunk. A lottószelvényt egy tömbbel modellezzük, amelynek elemei boolean típusúak. Minden számnak egy tömbelem felel meg, amely azt jelzi, hogy a szám be van-e jelölve.

Induláskor a tömb minden eleme hamis (false) lesz, ami azt jelenti, hogy még egyik sincs bejelölve. (Ez a Java-ban automatikus így van.)

Utána előállítunk 1 és 90 közötti véletlen számokat. Ha a tömbben a kisorsolt indexű elem még hamis, akkor igazra (true) állítjuk. Ezzel bejelöljük. Ha már be volt jelölve, újabb számmal próbálkozunk. Így biztosan nem lesz kétszer ugyanaz a szám. Mindezt addig folytatjuk, amíg be nem jelöljük mind az 5 számot.

Végül sorban végigmegyünk a tömb elemein, és amelyik be van jelölve, azt kiíratjuk.

Nézzük a programot:



A szelveny tömbnek eggyel több (91) eleme van, mert a nulladik elemet most sem használjuk. A kiírást is az 1-es elemnél kezdjük.

Készítsd el és próbáld ki a programot!



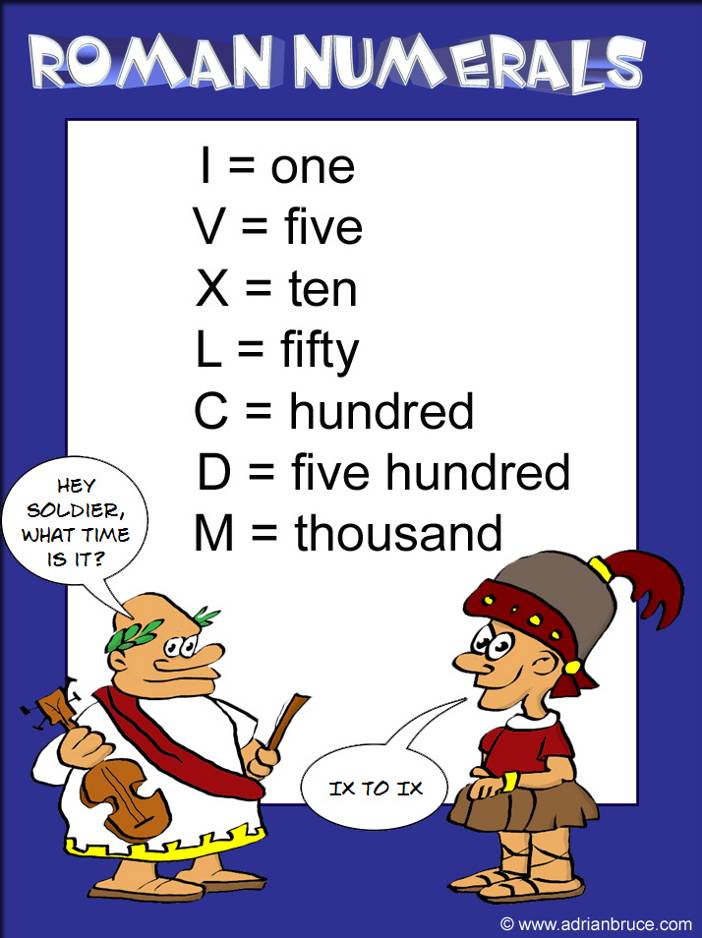
## Tömb kezdőértékeinek megadása

A tömb létrehozásakor rögtön megadhatjuk a tömb elemeit is a következő formában:

int[] szamok = {1,3,5,7,9};

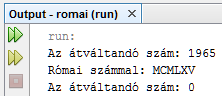
Ezzel létrehozunk egy öt egész számból álló tömböt, amely a felsorolt elemekből áll, és a tömb címét elhelyeztük a szamok változóban. Más típusoknál is lehet alkalmazni. Ezt fogjuk használni a következő példában is.

## Római számok

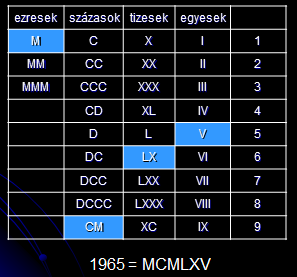


Készítsünk programot, amely a beírt 1 és 3999 közötti egész számot átváltja római számra! A programból a tartományon kívüli számmal (pl. 0-val) lehessen kilépni!

Minta:



Az átváltás elvégzését 1965 esetén az alábbi táblázat szemlélteti:



Vesszük az első számjegyet: 1-es. Az ezresek oszlopból vesszük az első sort: M

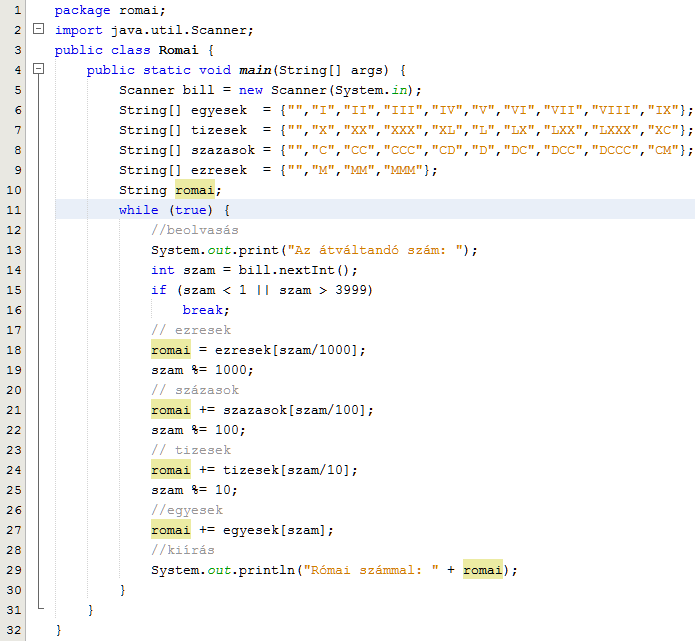
Vesszük a második számjegyet: 9-es. A százasok oszlopban a 9-es sor tartalma: CM

Vesszük a harmadik számjegyet: 6-os. A tizesek oszlopban a 6-os sor tartalma? LX

Vesszük az utolsó számjegyet: 5-ös. az egyesek oszlopban az 5-ös sorban V van.

Összeolvasva: MCMLXV

Számítsd át 1848-at római számra!   
Írd ide az eredményt:

Nézzük a programot:  


A fenti táblázat elemeit tömbökben tároljuk (6-9. sor). A számjegyeket az egész osztás és a maradékképzés segítségével határozzuk meg (18-28. sor). (Hasonlít a korábban készített címletezéshez.)

Gondold végig a program működését, készítsd el, majd próbáld ki! Ellenőrizd 1848 átváltását! Próbáld ki érvénytelen évszámal is!