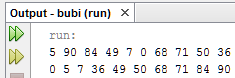
# 07. Rendezés

## Feladat

Készítsünk programot, amely létrehoz egy 10 elemű, 0 és 99 közötti véletlen számokat tartalmazó tömböt! Először írja ki a számokat az eredeti sorrendben, utána rendezze őket növekvő sorrendbe, végül írja ki őket úgy is!

Minta:



## Buborékos rendezés

A legegyszerűbb rendezési módszer a buborékos rendezés. Ennél először az utolsó elemet tesszük a helyére, utána az utolsó előttit, és így tovább. Legutoljára a második és vele együtt az első elem kerül a helyére.

Az utolsó elem úgy kerül a helyére, hogy az első elemtől indulva minden elemet összehasonlítunk az utána következővel, és ha szükséges, megcseréljük őket.

Az utolsó előtti elemet ugyanígy tesszük a helyére, de ennél már csak az utolsó előtti elemig kell összehasonlítani, mert az utolsó már biztosan a helyén van.

Így sorban helyére tesszük a többi elemet is.

Utoljára a második elemet úgy tesszük a helyére, hogy összehasonlítjuk az elsőt a másodikkal, és szükség esetén megcseréljük őket.

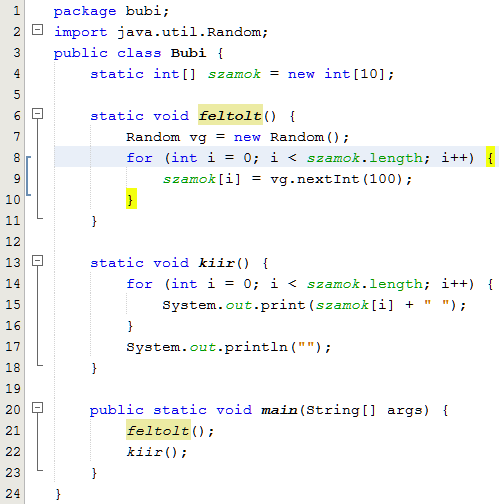
Kövesd végig a rendezést az alábbi ábrán! A színek jelentése:

* kék: összehasonlítás, nem kell csere
* piros: összehasonlítás, kell csere
* szürke: már a helyén van



## A feladat megoldása

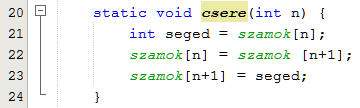
A program első változatában létrehozzuk a tömböt, feltöltjük véletlen értékekkel, majd kiíratjuk az elemeit:



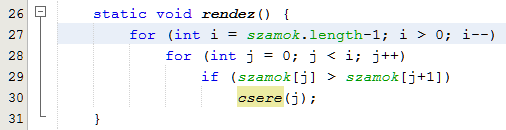
Próbáld ki!



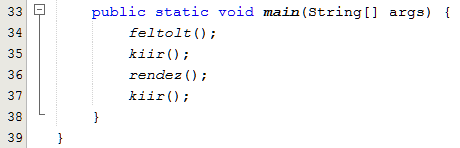
A rendezés elvégzéséhez szükség lesz egy csere() metódusra, amely megcseréli az n-dik és az n+1-dik elemeket. Írd a kiir() metódus után:



Most következik a buborékos rendezés:



És végül meg kell hívni a rendezést a főprogramból:



Próbáld ki a teljes programot!

## Rendezés csökkenő sorrendben

A csökkenő sorrendben történő rendezéshez csak egy apró módosításra van szükség. Végezd el ezt, és próbáld ki!

## Stringek rendezése

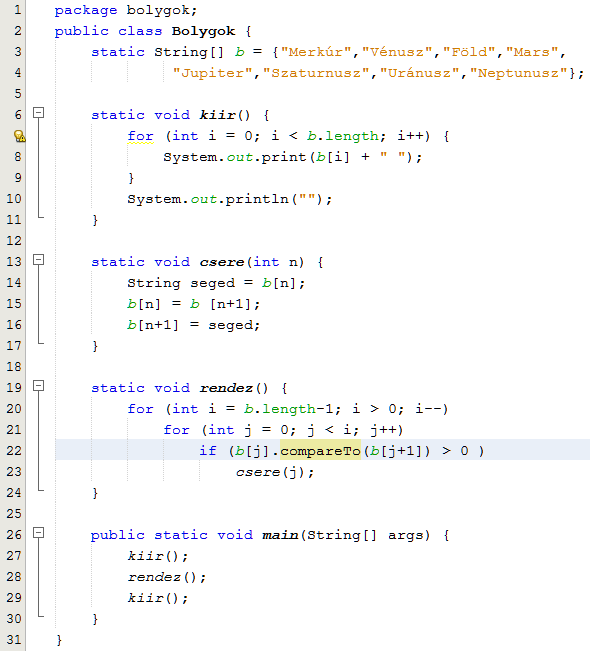
Stringeket nem hasonlíthatunk össze a kisebb és nagyobb jelekkel. A stringek összehasonlításához a compareTo() metódust használhatjuk. Például:

String a = "alfa";  
String b = "béta";  
if (a.compareTo(b) < 0) System.out.println("a kisebb, mint b");

A compareTo() metódus a pont előtti stringet hasonlítja össze a zárójelben lévővel (a-t b-vel). Ha az első előrébb van a rendezésnél, akkor az eredmény negatív, ha hátrább, akkor pozitív. Ha a két string megegyezik, akkor az eredmény 0.

A fenti példában az if utasítás feltétele igaz lesz.

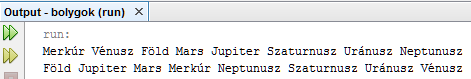
A stringek rendezését a *bolygok* nevű projekttel fogjuk kipróbálni. Ezt már előkészítettük, csak nyisd meg, és hasonlítsd össze a *bubi* programmal!



Nézzük a különbségeket:

* Most nem számokból, hanem stringekből álló tömböt rendezünk, amelyet néhány bolygó nevével töltünk fel.
* A rendezésnél nem > jellel, hanem a compareTo() metódussal hasonlítjuk össze a tömb két elemét.

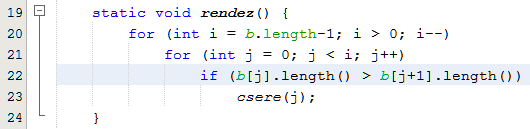
Próbáld ki a programot!



## Rendezés más szempont szerint

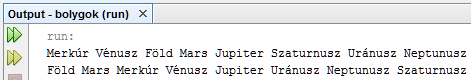
Előfordul, hogy nem ABC rendbe, hanem valamilyen más szempont alapján szeretnénk rendezni. Legyen most az a feladat, hogy a bolygók nevének hossza szerint rendezzünk! A legrövidebb nevű kerüljön előre, a leghosszabb nevű pedig a végére!

Ehhez csak a 22. sort kell átírni a rendez() metódusban:



Ebben a feltételben tetszőleges összehasonlítást megadhatunk, és a program e szerint fog rendezni.

Próbáld ki!



Megjegyzés: A buborékos rendezés a legegyszerűbb rendezési módszer, de egyben a legkevésbé hatékony is. Más szavakkal: sok (ezer vagy millió) adat esetén elég lassú. A mi feladatainkban ez nem probléma, de komoly programokban másik módszert kell alkalmazni.

## Feladat

A következő videó a buborékos rendezést szemlélteti:  
<https://www.youtube.com/watch?v=nY3As6NPmeQ>

Nézd meg!

Keress, és nézz meg olyan videókat, amelyek táncokkal szemléltetik a különböző rendezési módszereket!