# 12. Listák 1.

## Listák

A tömbök használatánál előre meg kell adni a tömb méretét, az később már nem változhat (legfeljebb átmásolhatjuk az elemeket egy másik tömbbe).

Amikor nem tudjuk előre a tárolandó elemek számát, akkor kényelmesebb egy lista (ArrayList) használata. A lista mérete tetszőlegesen változhat, bármikor hozzáadhatunk új elemet, vagy törölhetünk belőle egy elemet.

Az elemeknek azonos típusúaknak kell lenniük. Ez a típus egy tetszőleges osztály lehet. A primitív típusokat itt is be kell csomagolni (Integer, Double, …)

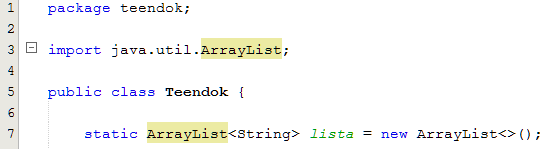


## Teendők program

Mai példánkban egy teendő listát (to-do list) fogunk készíteni.

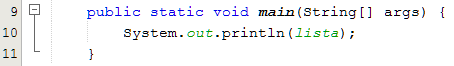
Kezdj egy programot *teendok* néven!

Hozz létre a programban egy lista nevű ArrayList<String> típusú változót, amely egy üres string listát tartalmaz! Ehhez importálni kell az ArrayList osztályt:

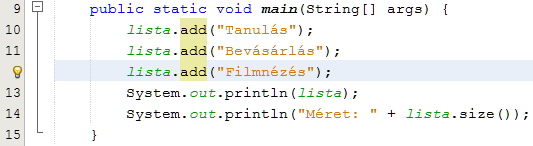


A <> jelek között kell megadni az egyes elemek típusát. A jobb oldalon is megadhattuk volna a <String> típust, de ez nem kötelező.

A főprogramban írasd ki az üres listát!

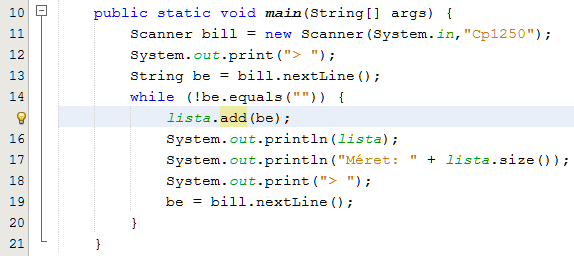


Adj hozzá néhány teendőt a listához! A hozzáadás az add() metódussal történik, a lista méretét pedig a size() metódussal lehet lekérdezni:



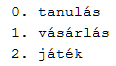
## Beolvasás

Temészetesen nem előre megadott tevékenységeket szeretnénk a listába, hanem be szeretnénk olvastatni az elemeket, ezért alakítsd át így a programot:

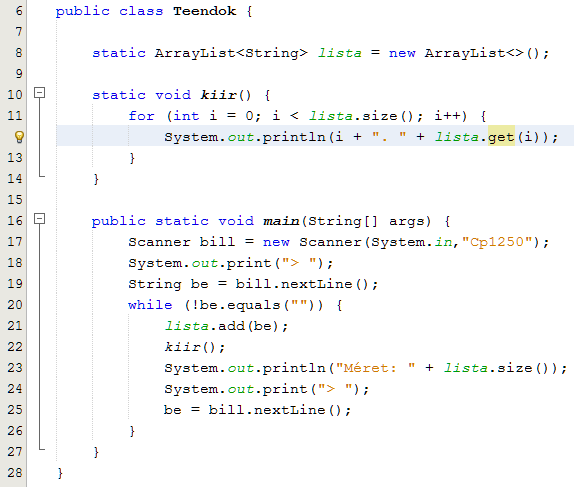


Próbáld ki! Sajnos minden alkalommal újra kell írni a teeendőket, mert még nem tudjuk fájlba menteni őket. Nemsokára megtanuljuk majd azt is.

Szeretnénk javítani a kiíráson úgy, hogy számozva, egymás alá írja ki a teendőket. Például:



Ehhez készítünk egy kiir() metódust, és ezt hívjuk meg a kiíráshoz. Ebben a get() metódust használjuk a lista elemeinek lekérdezéséhez. A lista.get(i) a lista i-dik elemét adja meg.

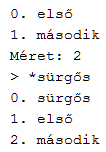


Próbáld ki!

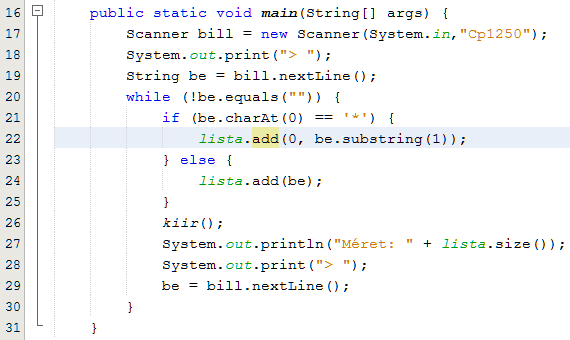
Ha a ciklusban nem lenne szükségünk az i értékére, akkor itt is alkalmazhattuk volna az egyszerűb for-each ciklust, mint a tömböknél.

Sürgős teendőinket nem a lista végéhez szeretnénk fűzni, hanem az elejére szeretnénk beszúrni. Ebben az esetben az add() metódusnak azt a változatát használjuk, amelyben azt is meg lehet adni, melyik elem elé szeretnénk beszúrni. Például a lista.add(0,”sürgős”) a lista 0-dik eleme elé szúrja be a sürgős elemet.

A teendő beírásakor az elejére írt csillaggal jelezzük, hogy a teendő sürgős:



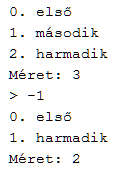
Alakítsd át így a main() metódust:



Próbáld ki!

## Törlés

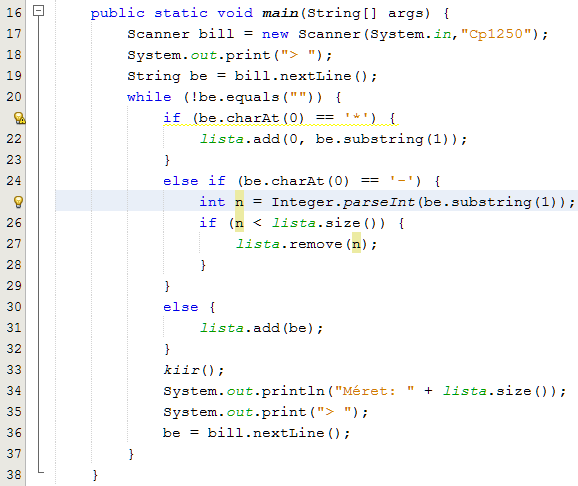
Ha valamit elvégeztünk, szeretnénk törölni a listából. Ezt egy – jellel és a teeendő sorszámával fogjuk megadni. Például:



Figyeld meg, hogy a – jel után közvetlenül írtuk a számot, nem hagytunk ki helyet közöttük.

A törléshez a lista.remove(n) metódust alkalmazzuk, amely a lista n-dik elemét törli.

Ezen kívül a beolvasott string – jel utáni részét át kell alakítani egy egész számmá, mert ez adja meg, hogy hányadik elemet kell törölni. Az átalakítást elvégezhetnénk egy új Scanner objektummal, de most kényelmesebb az Integer.parseInt() metódus használata:



Próbáld ki!

## Feladat

A program nem tájékoztatja a felhasználókat arról, hogyan kell használni. Egészítsd ki úgy, hogy induláskor kiírja a legfontosabb tudnivalókat!

## Összefoglalás

Az alábbi táblázat összefoglalja a legfontosabb listákkal kapcsolatos metódusokat. Melyiket nem használtuk a programban?

|  |  |
| --- | --- |
| **Példa** | **Leírás** |
| lista.add("Mosogatás") | Hozzáadja a Mosogatás elemet a lista végéhez. |
| lista.add(1,"Vasalás") | Beszúrja az 1-es indexű elem elé a Vasalás elemet. |
| lista.set(1,"Főzés") | Átírja az 1-es indexű elemet Főzésre. |
| lista.get(1) | Megadja a lista 1-es indexű elemét. |
| lista.remove(2); | Törli a 2-es indexű elemet. |
| lista.size() | Megadja a lista elemeinek számát. |

## Feladat

Egészítsd ki az alábbi táblázatot!

|  |  |
| --- | --- |
| **Adattípus** | **Elemek száma** |
| Array (tömb) | a.length |
| Arraylist | a. |
| String | a. |