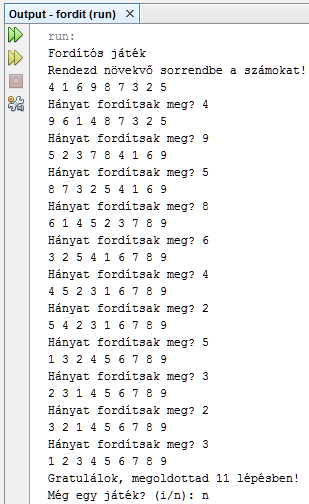
# 04. Tömbök 3.

## Fordítós játék

A tömbökről tanultak felhasználásával most egy egyszerű logikai játékot fogunk készíteni. A játék célja, hogy minél kevesebb lépésben növekvő sorrendbe rendezzük az induláskor összekevert 1…9 számokat. Minden lépésben megfordíthatjuk az első n elem sorrendjét a listában (n = 2…9).

Minta:



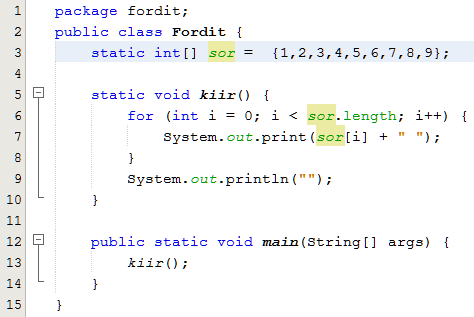
Figyeld meg, hogy a lépések mindig a lista elején fordítják meg az adott számú elem sorrendjét!

Mivel ez a program egy kicsit összetettebb az eddigieknél, először egy egyszerű változatot készítünk, majd azt fokozatosan bővítjük, amíg teljesen készen nem lesz.

## Sor kiírása

A programban szükségünk lesz egy tömbre, amely a számsor elemeit tartalmazza. Mivel több metódusban is fogjuk használni, az osztály elején fogjuk létrehozni.

A program első változata mindössze ennek a tömbnek a tartalmát fogja kiírni egy kiir() metódus segítségével:



Készítsd el és próbáld ki!



## Elemek megfordítása

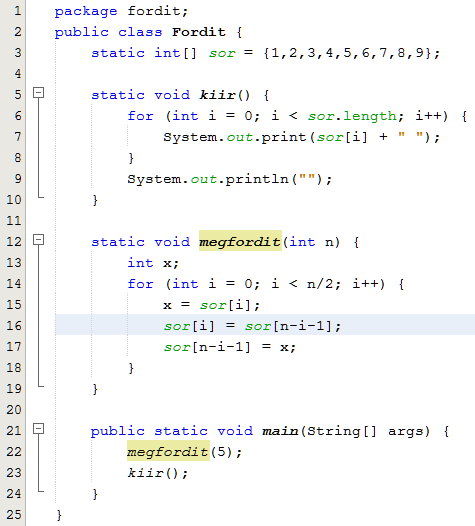
A következő változatban készítünk egy metódust, amely a tömb első n elemét fordítja meg, és ezt is teszteljük a főprogramban.

Hogyan lehet az első n elemet megfordítani? Egy ciklussal végigmegyünk az első n/2 db elemen, és mindig megcseréljük az i-dik és az n–i–1–dik elemet.

A következő táblázat n=5 esetén mutatja a cseréket. i értéke 0, 1 és 2 lesz. n–i–1értéke rendre 4, 3 és 2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i=0 | 5 | 2 | 3 | 4 | 1 | 6 | 7 | 8 | 9 |
| i=1 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 8 | 9 |
| i=2 | 5 | 4 | 3 | 2 | 1 | 6 | 7 | 8 | 9 |

Nézzük a megoldást:



Az elemek cseréje a 14-17. sorban történik. A cseréhez egy x nevű segédváltozót használunk.

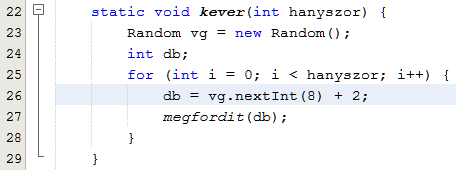
Alakítsd át ilyenre a programot, majd próbáld ki többféle n értékkel is! Az alábbi mintán az első 5 elemet fordítottuk meg.



## Keverés

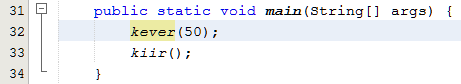
A keverést vissza lehet vezetni a fordításra: Ha egymás után többször megfordítjuk a lista elejének véletlenül kiválasztott hosszúságú részét, a lista összekeveredik. Elméletileg előfordulhat, hogy éppen az eredeti állapot lesz az eredmény, de kellő számú fordítás esetén ennek esélye igen kicsi.

Ebben a változatban készítünk egy kever() metódust, és teszteljük is. A program elején importáld a java.util.Random csomagot, majd írd be a kever() metódust a fordit() metódus után (lehetne elé is, a metódusok sorrendjének nincs jelentősége):



Figyeld meg a véletlen szám előállítását a 25. sorban! Mekkora véletlen számok lehetnek? Írd ide:

A main() metódust alakítd át ilyenre:



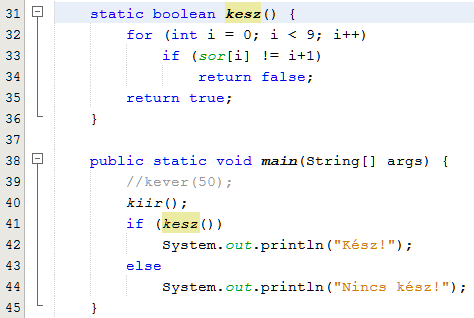
Próbáld ki! Ahányszor futtatod, mindig más eredményt kapsz.



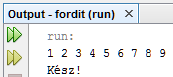
## Kész-e a rendezés?

Szükségünk lesz olyan metódusra is, amely eldönti, hogy sorban vannak‑e a számok. Ehhez egy ciklussal megyünk sorban a számokon. Ha valamelyik nem egyezik meg a ciklusváltozó+1-gyel, akkor nem jó a sorrend. Ha mind megegyezik, akkor sorban vannak a számok.

Írd be a kesz() metódust, és módosítsd a főprogramot így:



Próbáld ki így is (eredmény lent) és úgy is, hogy a 39. sorban törlöd a //-t. Akkor összekeveri a számokat a vizsgálat előtt.

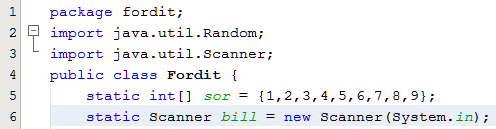


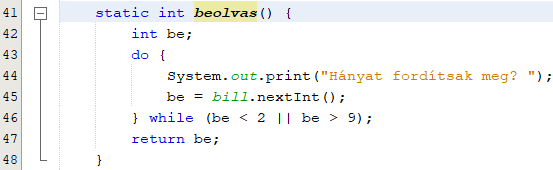
## Beolvasás

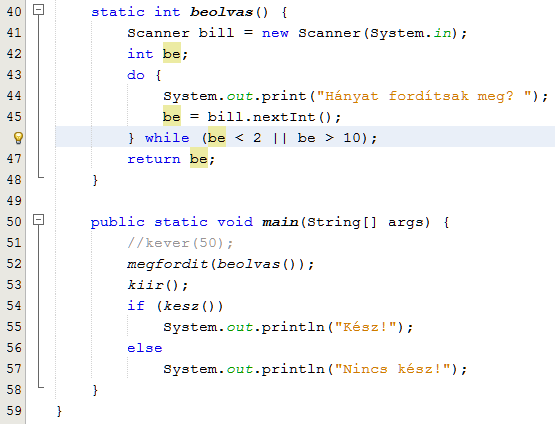
Még egy metódust fogunk használni a programban, amellyel beolvassuk, hogy a játékos hány számot szeretne megfordítani.

A beolvasáshoz importálnunk kell a Scanner osztályt. Mivel ezt a beolvas() metódusban és a főprogramban is használni fogjuk, a Fordit osztály elején létrehozunk belőle egy példányt.

Alakítsd át a program elejét így:

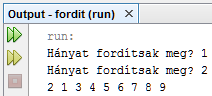


A beolvasást ellenőrizni kell, csak 2 és 9 közötti számokat lehet elfogadni. Írd be a beolvas() metódust a main() metódus elé, és alakítsd át a főprogramot így:  




Figyeld meg az 52. sorban, hogy a beolvasást eredményét közvetlenül átadjuk a megfordit() metódusnak! Így nem kell azt változóban tárolnunk.

Próbáld ki! Figyeld meg, hogy rossz érték beírása után újra kéri azt!



## Főprogram

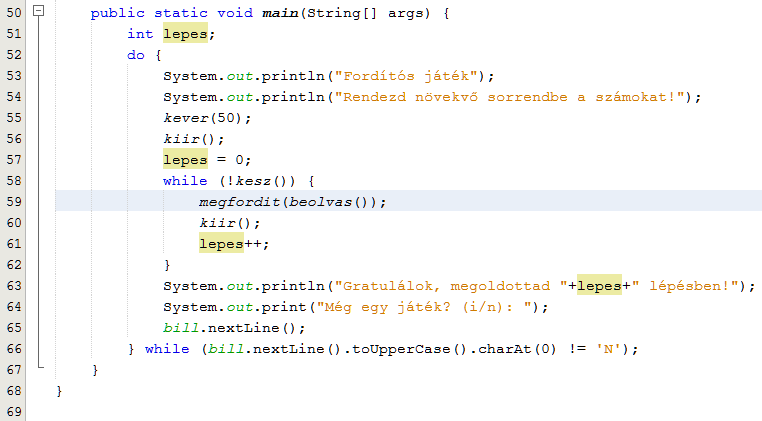
Elkészült minden szükséges metódus, már csak a main() metódust kell átalakítanunk a játékhoz.

Gondoljuk végig, mit kell ennek csinálnia egy játék alatt!

1. Kiírja, hogy mi ez, összekeveri a számokat, azokat is kiírja, és a lépésszámot 0-ra állítja.
2. Amíg nincsenek sorban a számok, megfordít annyit, amennyit beolvasott, újra kiírja őket és megnöveli a lépésszámot.
3. Beolvassa, hogy szeretne-e a játékos még egyszer játszani.

Ezt a három lépést addig kell ismételni, amíg a játékos nem kis- vagy nagy N-nel válaszol a kérdésre.

A megoldás:



Készítsd el és próbáld ki!

## Stratégia

Milyen stratégiával lehet megoldani a feladatot? Mielőtt tovább olvasnál, görgess vissza az szöveg elejére, és nézd meg, hogyan csináltuk! Először próbáld meg önállóan kitalálni a módszert!

A megoldás nagyon egyszerű: Először a lista utolsó elemét tesszük a helyére, majd az utolsó előttit és így tovább a másodikig (az első ekkor már a helyén lesz). Egy elem helyre viteléhez először annyit fordítunk, hogy az elem a lista elejére kerüljön. Ahhoz, hogy a lista elejéről a helyére tegyük a számot, pont annyit kell forgatni, amennyi a szám értéke (pl. 5-ös esetén 5-öt). Így már biztosan el tudjuk végezni a rendezést, de célszerű közben figyelni a rövidítési lehetőségeket.

Rendezd a számokat először csak gépiesen, a fenti módszerrel! A következő játékban már törekedj arra, hogy minél kevesebb lépés kelljen!