# 14. Keresés

Ismétlő kérdések:

1. Hogyan kereshetünk meg egy tömbben vagy egy listában egy elemet?
2. Mi a különbség a lineáris és a bináris keresés között? Mikor alkalmazható a bináris keresés? Hány lépés kell a két esetben?

Egy kis segítség: Nyisd meg a kapott GIF animációt, és nézd meg a különbséget!

## A feladat

Ebben a leckében a kétféle keresést fogjuk kiróbálni egy olyan tanuló listával, mint az előző leckében, csak most egy kicsit több tanulóval.

Nyisd meg a *tanulok* projektet a NetBeans-ben!

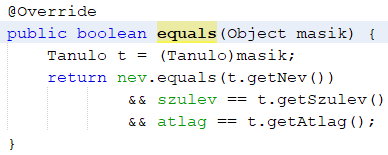
Nézd meg a Tanulo és a Tanulok osztályokat!

## Egyenlőség vizsgálata

Két objektum akkor egyenlő, ha minden mezőjük megegyezik. Ezt nem vizsgálhatjuk a == jelekkel.

A String osztály equals metódusát készen kapjuk, de a többi osztálynál nem ez a helyzet. Ezek az Object osztálytól örökölnek egy equals metódust, amit nekünk felül kell definiálnunk.

Készítsd el először ezt a Tanulo osztályba:

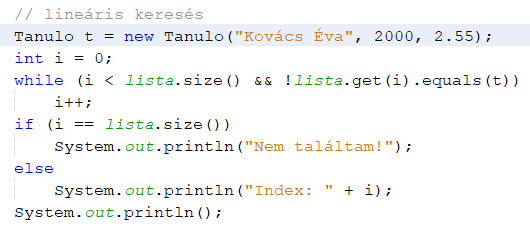


A felülírandó equals metódus egy Object típusú paramétert vár. Ezt először át kell alakítanunk Tanulo típussá. Feltételezzük, hogy Tanulo típusú objektumot kap, ezért ezt nem vizsgáljuk.

Utána mezőnként összehasonlítjuk az értékeket, és ennek eredményét adjuk vissza.

## Lineáris keresés

Most már kipróbálhatjuk a lineáris keresést. Írd ezt a Tanulok osztály main metódusának végére:



Fontos, hogy a ciklusban először azt vizsgáljuk, hogy elértük-e a lista végét, és csak utána próbáljuk meg az összehasonlítást. Fordított sorrendben nem működne!

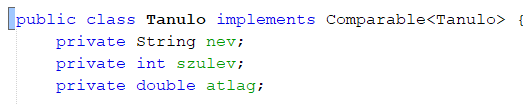
Próbáld ki a programot! Utána írd át valamilyen nem létező adatra (pl. másik év) és próbáld ki így is!

## Comparable interface

A bináris kereséshez először név szerint, azon belül születési év szerint, azon belül pedig átlag szerint fogunk rendezni.

Mivel csak ez a rendezés lesz, és a keresésnél is erre a sorrendre lesz szükség, a Comparable interface-t fogjuk megvalósítani.

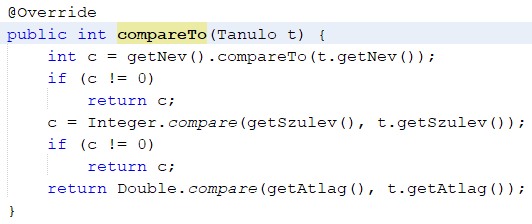
Először módosítsd a Tanulo osztály fejlécét így:



A compareTo metódusnak negatív számot kell visszaadnia, ha a vizsgált objektumnak előrébb kell kerülnie rendezéskor, mint a masik objektumnak, pozitív számot kell visszadnia, ha vizsgált objektumnak kell hátrább kerülnie, és 0-t, ha a két objektum egyenlő.

Ha a két objektum neve megegyezik, akkor össze kell hasonlítani a születési éveket is, és ha azok is megegyeznek, akkor az átlagokat is.

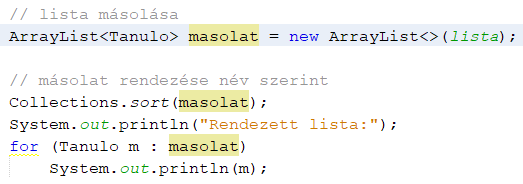
Készítsd el a compareTo metódust így:



Gondold végig a metódus működését!

## Bináris keresés

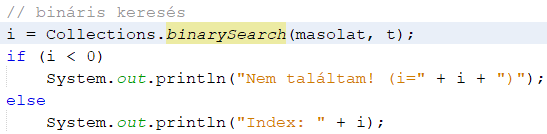
A bináris kereséshez először lemásoljuk a listát, majd a másolatot növekvő sorrendbe rendezzük. Írd ezt a Tanulok osztály main metódusának végére:



Próbáld ki! Ellenőrizd a sorrendet!

A bináris keresését a Collections.binarySearch metódussal végeztethetjük el. Két változata van: az egyikben a listát és a keresett kulcsot kell megadni, a másikban a comparatort is. Mi most az első változatot fogjuk alkalmazni.

Egészítsd ki a *main* metódust a következőkkel:



Próbáld ki létező és nem létező tanulóval is! Milyen értéket ad vissza?

Megoldás: Ha megtalálta az elemet, akkor a listabeli (nemnegatív) indexét adja vissza, egyébként pedig egy negatív számot. A negatív számot úgy kapja, hogy annak az elemnek az indexét, amelyik utána következne a listában, megszorozza -1-gyel, majd kivon belőle 1-et.

## Keresés fájlban

Az eddigiekben csak azt az esetet vizsgáltuk, amikor a lista a memóriában van.

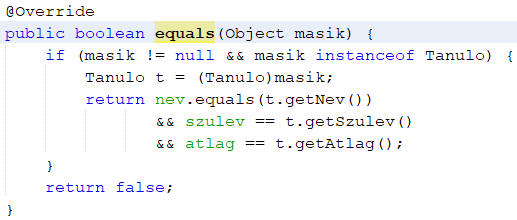
Ha szöveges fájlból kell beolvasni a listát, akkor két eset van:

* Ha nem túl hosszú a lista, akkor beolvassuk a memóriába. A memóriának egy részt elfoglalja, de nagyon gyorsan lehet rendezni, vagy keresni benne.
* A másik esetben sok adat van a fájlban, és nincs elég memória a beolvasásához. Ebben az esetben minden keresésnél soronként kell olvasni a fájlból. Így egyszerre csak egy sor adatai vannak a memóriában, de a keresés nagyon lassú lesz.

(8. tétel)

## Kiegészítés

A lecke elején az equals metódusban nem vizsgáltuk a kapott objektumot, amivel össze kell hasonlítani a tanulót. Egy teljesebb változat így nézne ki:



Próbáld ki!