# 13. Rendezés

Ismétlő kérdések:

1. Milyen rendezési módszereket ismersz?
2. Milyen kész metódusok vannak a Javaban a tömbök és a listák rendezésére?

## A probléma

Egy tömb vagy egy lista rendezésekor össze kell hasonlítani az elemeket. Amíg ezek számok vagy stringek, addig ezzel nincs gond.

Ha viszont a tömb vagy lista elemei objektumok, akkor a Java nem tudja hogyan hasonlítsa össze őket.

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

Nézd meg a Tanulo és a Tanulok osztályokat!

A program nem működik. Mi lehet a hiba?

A rendezést többféleképpen fogjuk elvégezni. Ezért először készíts egy Git repository-t a projekt mappájába (Team , Git, Initilize Repository), és végezz egy Commit-ot („Alaphelyzet”)! Az egyes megoldásokat külön ágakon fogjuk elkészíteni.

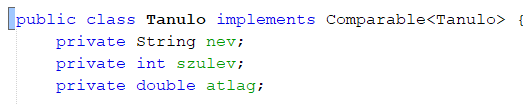
## Comparable interface

Kezdj egy új ágat *comparable* néven, és válts át rá! (Team, Branch/Tag, Create Branch)

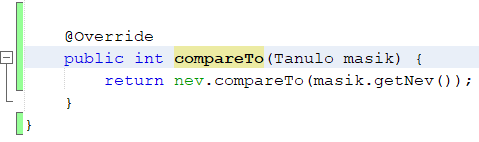
Az első megoldásban úgy módosítjuk a Tanulo osztályt, hogy megvalósítsa a Comparable interface-t. Ezután már lehet rendezni a listát.

A Comparable interface csak egy compareTo metódust tartalmaz, a megvalósításához azt kell felülírni. Ezt most úgy fogjuk megtenni, hogy a nev mező szerint rendezzen.

Először módosítsd a Tanulo osztály fejlécét így:



Ezután készítsd el a compareTo metódust 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.

A Stringek compareTo metódusa is így működik, ezért ebben az esetben csak meg kell hívnunk a megfelelő adattagokkal.

Commit-olj („Comparable”), majd futtasd a programot!

Hogyan lehetne csökkenő sorrendet beállítani?

Hogyan lehetne szulev vagy atlag alapján rendezni?

A szulev mező szerinti növekvő rendezéskor például ez kerülhetne a compareTo metódusba:

return szulev – masik.szulev

## Comparator interface

A Comparable interface használatának korlátja, hogy csak egyféle rendezést lehet megadni.

Ha a programban többféleképpen is kell rendezni, akkor a Comparator interface-t használjuk helyette.

Előkészületképpen válts vissza a master ágra (Team, Branch/Tag, Switch to Branch…)!

Utána készíts egy új ágat *comparator* néven, és válts át rá (Team, Branch/Tag, Create Branch…)!

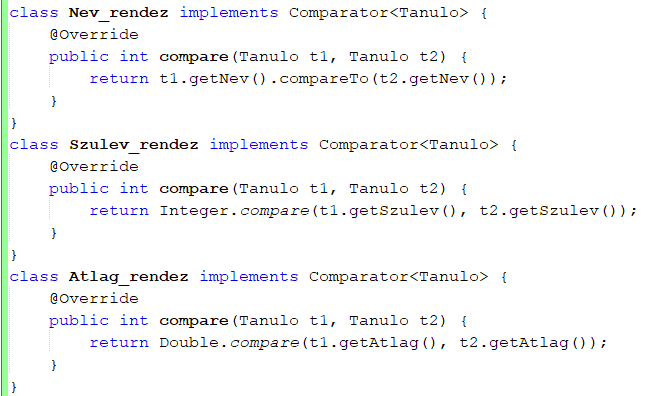
A Comparator interface-t annyi osztállyal valósítjuk meg, ahányféleképpen rendezni szeretnénk.

Ha több másik osztályban is szeretnénk használni ezeket az osztályokat, akkor külön fájlba kell tenni őket publikus osztályként.

Most viszont csak a Tanulok osztályban szeretnénk alkalmazni őket, ezért a Tanulok osztály végére írjuk őket.

Mivel egy fájlban csak egy publikus osztály lehet, ezek az osztályok nem lesznek publikusak, de ez most nem probléma.

Készítsd el az alábbi osztályokat a Tanulok osztály **után**:

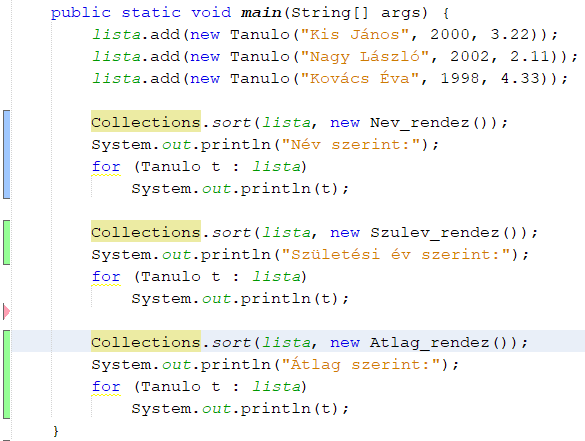


A compare metódusnak negatív számot kell visszaadnia, ha az első objektum kisebb, pozitívat, ha a második, és nullát, ha egyenlőek.

Figyeld meg, hogyan hasonlítjuk össze az egész és a valós számokat!

A Collections.sort metódusnak van olyan változata, amelynél ezeknek az osztályoknak egy-egy példányával adjuk meg a rendezési sorrendet.

Most mind a három rendezést ki szeretnénk próbálni, ezért írd át a main metódust így:



Végezz egy commit-ot („Comparator”), majd próbáld ki a programot!

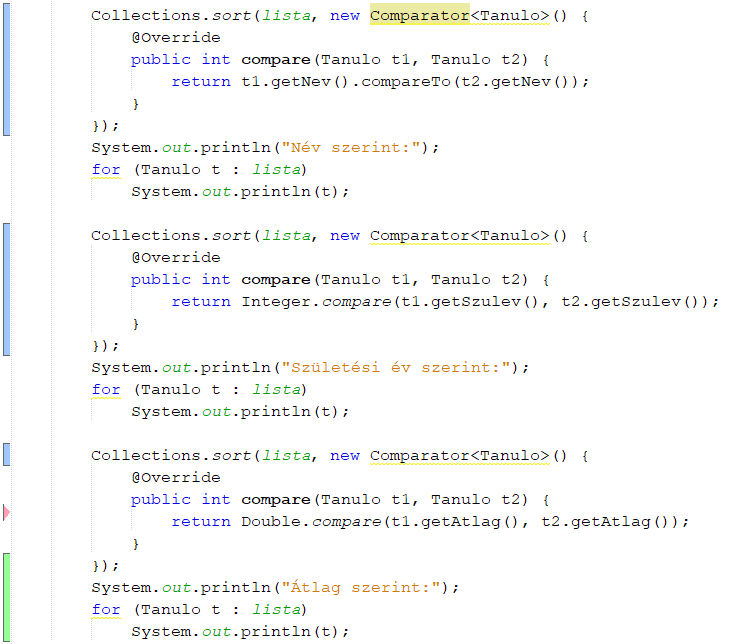
## Névtelen osztályok

Készíts egy újabb ágat *anonymous* néven, amely a *comparator* ágból nyílik.

A Comparator interface-t megvalósító osztályok belső osztályok is lehetnek, vagyis a Tanulok osztályon belülre is írhatjuk őket. (Csak akkor statikusoknak kell lenniük.)

Ha egy ilyen osztályt csak egy helyen használunk, akkor – a JavaScripthez hasonlóan – névtelen, anonymous osztályként is meg lehet adni.

Írd át a main metódusban a rendezéseket névtelen osztályokkal:



Utána töröld a fájl végéről a feleslegessé vált osztályokat, és futtasd a programot!

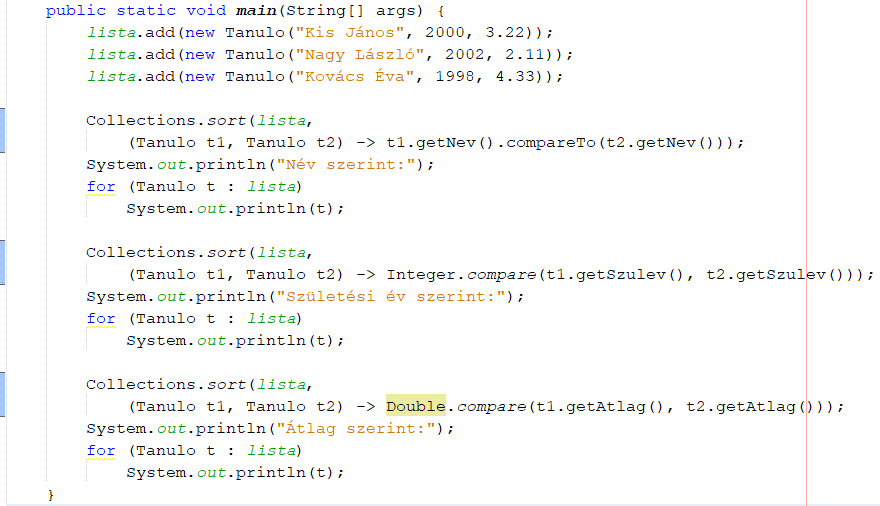
Ha jól működik, készíts egy commitot („Anonymous osztályok”)

## Lambda kifejezések

A Java 8-as változatától kezdve azokat az interface-eket, amelyek csak egy absztrakt metódust tartalmaznak, meg lehet valósítani lambda kifejezésekkel. (Ezeket funkcionális interface-eknek hívják.)

A Comparator interface is ilyen, ezért a névtelen belső osztály helyett is írhatunk lambda kifejezést.

Próbaképpen írd át a main metódus rendezéseit így, majd próbáld ki!



Készíts egy commit-ot az anonymous ágba („Lambda kifejezések”)!

Tovább rövdítheted a kifejezéseket, ha a t1 és a t2 előtt elhagyod a típust (Tanulo), mert ezt nem muszáj megadni. Próbáld ki!

Megjegyzés: A lambda kifejezéseket természetesen a Comparable interface-nél is lehet használni.

## Ágak összefésülése

Most már több változat is van a programból. Az egyes ágakra átváltva újra végignézheted őket.

Ezután válts a *master* ágra, és fésüld össze vele az *anonymous* ágat!

## Kérdés

Hogyan lehetne megfordítani a rendezéseket (hogy csökkenő sorrendbe rendezzen)?