# 24. Osztályok 5.

## Kéktúra

Ebben a leckében folytatjuk a Kéktúrával kapcsolatos feladat megoldását.

Emlékeztető:

A *kektura*.*csv* állomány első sorában a túra kezdetének tengerszint feletti magassága található méterben megadva. A további sorok a túra egy-egy szakaszához tartozó adatokat tárolják. Az adatokat pontosvessző választja el egymástól. Például:

Hidegkuti major;Leteres a Sztupahoz;3,903;153;53;n

A sorokban lévő adatok rendre a következők:

* A túraszakasz kiindulópontjának a neve.   
  Például: „Hidegkuti major”
* A túraszakasz végpontjának a neve.   
  Például: „Leteres a Sztupahoz”
* A túraszakasz hossza [km]. Például: „3,903”
* A túraszakaszon mért emelkedések összege [m].   
  Például: „153”
* A túraszakaszon mért lejtések összege [m]: Például: „53”
* A szakasz végpontja pecsételőhely-e: „i” vagy „n”

Nézd meg a *kektura.csv* fájl tartalmát!

A következőkben meghatározzuk:

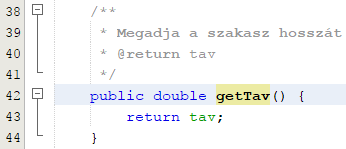
* a túra teljes hosszát,
* a három leghosszabb szakasz hosszait, és
* a legrövidebb szakasz hosszát.

Közben megismerkedünk a lambda kifejezésekkel és megtanuljuk sorba rendezni az objektumokat.

## Teljes hossz kiszámítása

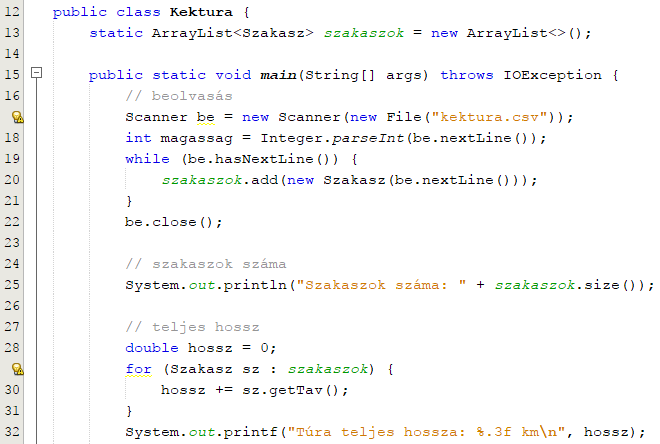
Következő feladatunk a túra teljes hosszának meghatározása. Ehhez végig kell menni az összes szakaszon, és össze kell adni az egyes távokat.

Az első lépés, hogy a Szakasz osztályban készítünk egy publikus metódust, amellyel le lehet kérdezni egy szakasz hosszát. Egészítsd ki a Szakasz osztályt így:



Ez a metódus annyira egyszerű, hogy nem fogjuk tesztelni.

A főprogramban létrehozzuk a listát, beolvassuk az adatokat, kiírjuk a szakaszok számát, majd kiszámítjuk a teljes hosszt:



Figyeld meg az egyszerűsített for ciklust és a printf() használatát!

Próbáld ki! Mennyi a teljes hossz? Írd ide:

## Lambda kifejezések

A lambda kifejezések a Java 8-as változatától kezdve használhatók, és sokszor lényegesen lerövidítik a szükséges kódot.

Egy lambda kifejezés tulajdonképpen egy egyszerűsített metódus.

Vegyük például a következő metódust:

int plusz(int a, int b) {  
 return a + b;  
}

Ez lambda kifejezéssel így írható:

(a, b) -> a + b

Figyeld meg, hogy:

* ennek a metódusnak nincs neve,
* a paramétereket zárójelek között kell megadni, utána nyíllal,
* a visszaadott értéknek és a paramétereknek nincs megadva a típusa.

A lambda kifejezéseket akkor fogjuk használni, amikor paraméterként kell megadni egy metódust, így nem lesz szükség a névre.

A típusokat a fordító fogja kitalálni a helyzetnek megfelelően (nem csak egészekkel működik!)

a és b helyett tetszőleges neveket használhatunk.

## Objektumok sorba rendezése

Már tanultuk, hogy egy lista objektumait a Collections.sort() metódussal tudjuk sorba rendezni. (Tömbök esetén pedig az Arrays.sort() metódust használjuk.)

Integer, Double vagy String típusú objektumok esetén egyértelmű, hogyan kell rendezni.

Más objektumok, például Szakasz típusú objektumok esetén viszont nem tudja a fordító, hogyan hasonlítsa össze őket. Ilyenkor a sort metódusnak megadunk egy lambda kifejezést, amely meghatározza az összehasonlítást.

A lambda kifejezésben a és b két objektum, az eredmény pedig:

* <0, ha a<b,
* 0, ha a=b,
* >0, ha a>b.

A stringek CompareTo() metódusa is pontosan így működik. Ezért két szöveges adattag összehasonlítását így adjuk meg (itt feltételezzük, hogy a getEleje() metódus az eleje adattag értékét adja meg):

(a, b) -> a.getEleje().compareTo(b.getEleje())

Ez alapján a sort() metódussal a szakaszok induló állomása szerint növekvő sorrendbe rendeznénk. Hogyan lehetne csökkenő sorrendet megadni?

Egész számok esetén alkalmazhatjuk az Integer.compare metódust. Ha például a szintvaltozas metódus azt adja meg, hogy mennyi az emelkedés és a lejtés különbsége az adott szakaszon, akkor így rendezhetünk e szerint:

(a, b) -> Integer.compare(a.szintvaltozas(), b.szintvaltozas())

Valós számok esetén használhatjuk a Double.compare metódust. Ha például a távok szerint növekvő sorrendbe szeretnénk rendezni, akkor ezt a lambda kifejezést adjuk meg a sort() metódusnak:

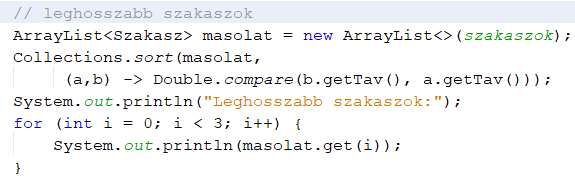
(a, b) -> Double.compare(a.getTav(), b.getTav())

Hogyan lehet csökkenő sorrendet megadni?

## Leghosszabb szakaszok

Most az a feladat, hogy írassuk ki a három leghosszabb szakasz adatait. Ezt a következő lépésekkel végezzük el:

1. Készítünk egy másolatot a listáról, mert később az eredeti sorrendben is szükségünk lesz rá.
2. A másolatot csökkenő sorrendbe rendezzük.
3. Kiíratjuk a csökkenő sorrendbe rendezett lista első három elemét.

Egészítsd ki a főprogramot az alábbiak szerint:  


Figyeld meg, hogyan készítettünk másolatot a listáról, és hogyan alkalmaztuk a lambda kifejezést!

Próbáld ki a programot!

A következő leckében befejezzük a feladat megoldását.

## Önálló feladat

Írasd ki a legrövidebb szakasz adatait is!

