8. heti feladatok

1. órai feladat Egy amatőr sportoló szereti az állóképességi sportokat, ezért fut, úszik, kerékpározik és túrázik. Egyik évben eldöntötte, hogy összeírja, mennyi pénzt költ ezekre a sportokra. Ennek érdekében havi bontásban csv állományokban rögzítette, hogy milyen sportokhoz kapcsolódó kiadásai voltak.

A csv állományokat úgy nevezte el, hogy `yyyy_mm.csv` mintát követett a nevük, és az egy évre vonatkozó fájlokat egy könyvtárban tárolta. Amelyik hónapban nem volt semmilyen kiadása, ahhoz a hónaphoz nem hozott létre csv állományt.

A csv állományok fejléc sort nem tartalmaznak. Egy soruk pedig a `sportág,szöveges leírás,dátum,kiadás összege` formátumot követi. Sportágként a `Cycling`, `Hiking`, `Running` és `Swimming` szavak szerepelhetnek. A szöveges leírás arról szól, hogy miért adott ki pénzt, pl.: uszoda belépő, kerékpár hajtókar, futócipő. (Mivel nem magyar anyanyelvű a sportoló, ezért angolul rögzített mindent, így legalább a karakterkódolási dolgokkal se kell foglalkoznia.) A dátum `YYYY.MM.DD` formátumban jelenik meg, aminek előnye, hogy a **DateOnly** osztály parszere így könnyedén tudja értelmezni. A kiadás összege egész számként van eltárolva.

A feladat megoldásához három osztályt már előkészítettünk, melyeknek kezdeti kódját le tudják tölteni.

- A **TrainingCost** osztály képes egy vásárlási bejegyzés adatait eltárolni. Az osztálynak van egy parszere is, mely egy megfelelő formátumú stringet képes **TrainingCost** típusúvá átalakítani. A parszer tesztelése is megtörtént.
- A **MonthlyCosts** osztály tárolja el az egy havi költségeket a publikusan elérhető `TrainingCosts` auto-property-ben. Az osztály rendelkezik egy `MonthlyCosts LoadFrom(string filename)` szignatúrájú statikus metódussal, amely képes egy adott nevű csv fájlból beolvasni a havi kiadásokat. Ennek a metódusnak a tesztelése is megtörtént már.
- A **YearlyCosts** osztály tárolja el az egy évre vonatkozó adatokat. Ehhez egy olyan publikus auto-propertyként megvalósított 12 elemű tömböt használ, melynek minden eleme **MonthlyCosts** típusú. Ha egy adott hónapban nem volt költés (azaz nem létezett csv állomány), akkor null értéket tárol az adott elemnél a tömb. Az osztálynak van egy `YearlyCosts LoadFrom(string folderName)` szignatúrájú statikus metódusa, amely egy könyvtárban lévő csv állományokat képes feldolgozni. A metódus tesztelése már megtörtént.

Az Ön feladata az alábbi „lekérdezések" implementálása. Az egyes feladatoknál érdemes átgondolni, hogy melyik programozási tétel, vagy mely programozási tételek összeépítésének használata lehet célravezető. Az egyes feladatrészek implementálása után írjon teszteseteket is, melyekkel le tudja fedni az össze előálló határesetet. A teszteléshez szabadon készíthet további csv állományokat is. Ügyeljen arra, hogy az újabb funkcionalitások bevezetése közben a korábban már létező tesztek sikeres futása ne sérüljön.

1. A **MonthlyCosts** osztályban megvalósítandó funkcionalitások:
   1.1. Adja össze, hogy mennyi volt a teljes költés egy hónapban.
   1.2. Határozza meg, hogy egy adott feltétel teljesülése[1] esetén mennyi volt a teljes költés egy adott hónapban. (Feltételre példa: csak az úszásra és kerékpározásra fordított költések érdekelnek.)
   1.3. Határozza meg, hogy volt-e egy adott feltételnek megfelelő költés egy hónapban.

[1] Az egyes feltételeket érdemes általánosan, predikátum függvények használatával implementálni.

1.4. Határozza meg, hogy az adott hónap minden kiadása megfelelt-e egy adott feltételnek.
1.5. Határozza meg, hogy volt-e legalább k darab, egy adott feltételnek megfelelő költés egy hónapban.
1.6. Határozza meg, hogy melyik volt a k-adik adott feltételnek megfelelő költés egy hónapban, ha volt egyáltalán ilyen.
1.7. Számolja össze, hogy hány darab olyan költés volt egy hónapban, ami egy adott feltételnek megfelelt.
1.8. Adja meg, hogy melyik volt a legnagyobb kiadással járó költés egy hónapban. (Ha több is van, akkor csak az elsőt adja vissza.) Ha nem volt egyetlen költés se a hónapban, akkor dobjon el egy **ZeroLengthArrayException** kivétel objektumot.
1.9. Adja meg, hogy melyek voltak a legnagyobb kiadással járó költések egy hónapban. (Ha több is van, akkor az összesnek az indexét adja vissza.)
1.10. Határozza meg, hogy melyik volt a legnagyobb kiadással járó költés egy hónapban azok között, melyek egy adott feltételnek megfelelnek.
1.11. Határozza meg, hogy ha a költségeket úgy számolja, hogy kerékpározás és futás esetén 2-vel osztja őket, akkor melyik volt a legnagyobb költés az adott hónapban.
1.12. Gyűjtse ki az összes olyan költés indexét, amely egy adott feltételnek megfelel.
1.13. Gyűjtse ki az összes olyan kiadás minden adatát, amely egy adott feltételnek megfelel.
1.14. Alakítsa át a költéseket tároló tömböt úgy, hogy az elejére kerüljenek azok a költések, amelyek egy feltételnek megfelelnek, a végére pedig azok, amik nem.

2. A **YearlyCosts** osztályban megvalósítandó funkcionalitások:

2.1. Határozza meg, hogy melyik hónapban fordított a legtöbb pénzt sportolásra.
2.2. Határozza meg, hogy melyik hónapban fordított a legtöbb pénzt egy adott sportágra.
2.3. Határozza meg azokat a kiadásokat, amelyek sportága és leírása azonos két megadott hónap esetén.
2.4. Gyűjtse ki, hogy mely sportágra hányszor fordított pénzt az év során.

Bármilyen egyéb lekérdezést is megvalósíthat, ami még eszébe jut.

2. gyakorló feladat Valósítson meg egy kezdetleges leckekönyvhöz kapcsolódó funkcionalitásokat az alábbiak szerint:

- Hozzon létre egy **SubjectEntry** nevű osztályt, mely ebben a feladatban a tantárgyat és a hozzá kapcsolódó féléves eredményt fogja tárolni az alábbi felépítés szerint:
  - A tantárgy neve.
  - Az eredmény megszerzésének jellege enumként (`Banned`, `MidYearMark`, `Exam`).
  - Egy egész érték ami 1-5 értéket vehet fel.
  - Az egyes értékek kívülről csak olvasható formában legyenek elérhetők és kezdeti értéket a fájl konstruktorából kapjanak.
  - Legyen egy statikus `SubjectEntry Parse(string str)` metódusa, amely egy strukturált stringből képes létrehozni egy **SubjectEntry** példányt.
- Hozzon létre egy **Semester** nevű osztályt, mely egy félév bejegyzéseit fogja tárolni az alábbiak szerint:
  - Az adott szemeszter szöveges azonosítója, mint például („2024/25/2”).
  - Az adott félévben elért tantárgyi bejegyzések (**SubjectEntry**) tömbje
  - Az egyes értékek kívülről csak olvasható formában legyenek elérhetők és kezdeti értéket a fájl konstruktorából kapjanak.

  - Legyen egy statikus `Semester Parse(string str)` metódusa, amely egy strukturált stringből képes létrehozni egy **Semester** példányt.

- Hozzon létre egy **Student** nevű osztályt, mely egy hallgató leckekönyvét fogja tárolni az alábbiak szerint:
  - A hallgató neve.
  - A hallgató neptun kódja.
  - Az eddigi félévei (**Semester**) egy tömbben tárolva.
  - Az egyes értékek kívülről csak olvasható formában legyenek elérhetők és kezdeti értéket a fájl konstruktorából kapjanak.
  - Legyen egy statikus `Student Parse(string str)` metódusa, amely egy strukturált stringből képes létrehozni egy **Student** példányt.

Példa egy strukturált stringekből összeálló fájlra

```
Gipsz Jakab;BATM4N;2022/23/1@SZTF1:Banned:1&DIMAT:Exam:4&FZK:MidYearMark:3#2022/23/2@SZTF1:Exam:1#2024/25/1@SZTF1:Exam:5

Kiss Pál;K1SSPL;2024/25/2@SZFA:Exam:5#2024/25/2@ALGA:Exam:4

...
```

- Implementálja az alábbi lekérdezéseket:
  1. Határozza meg, hogy egy paraméterként átadott tantárgynál mi volt a maximális osztályzat amit elértek egy adott szemeszterben.
  2. Határozza meg, hogy egy paraméterként átadott tantárgynál mi volt a maximális osztályzat amit elértek bármely szemeszterben.
  3. Határozza meg, hogy egy paraméterként megkapott hallgató egy paraméterként megkapott tárgyat megpróbált-e már sikertelenül 6 alkalommal?
  4. Egy hallgatónál egy félévben válogassa szét a sikeres és sikertelen tárgyakat ugyan azon tömbön belül.
  5. Határozza meg azokat a hallgatókat akik ugyan abban a félévben ugyan azt a tárgyat végezték el sikeresen.