Karakter feldolgozó és elemző feladatsor

Bevezetés

Ebben a feladatban egy fantázia világ szereplőit leíró adathalmazt kell feldolgoznod.
A bemenet egy karaktereket tartalmazó szövegfájl, melyben minden sor egy karakter adatait tartalmazza, mezők **|** jellel elválasztva.
A cél a sorok ellenőrzése, érvényes karakterek objektumorientált kezelése, majd a feldolgozott adatok alapján különféle lekérdezések és statisztikák készítése.

1. Feladat – A bemeneti adatfájl szerkezete

A karaktereket egy szövegfájl tartalmazza, minden sor egy karaktert ír le a következő mezőkkel:

Mező index	Megnevezés	Típus	Megszorítás
0	ID	string	Egyedi azonosító (pl. CH001)
1	Név	string	A karakter neve
2	Szint	int	0–100 között lehet, különben a sor hibás
3	Karakter osztály	enum	Warrior, Mage, Rogue, Cleric, Ranger
4	Életerő	int	Nem lehet negatív
5	Mana	int	Lehet 0 vagy pozitív; -1 hibás
6	Megítélés	enum	Good, Neutral, Evil
7	Régió	string	Nem lehet üres
8	Arany	int	0–1.000.000 között

Feladat:
Olvasd be a fájlt soronként, bontsd fel a mezőket a **|** jelek mentén, majd minden mezőt validálj.
Ha bármelyik mező hibás, a sort ki kell hagyni, és nem szabad példányosítani belőle objektumot.

2. Feladat – A Karakter osztály kibővítése egy számított „abszurd” születési dátummal

A Karakter osztály alapmezői mellett egy új automatikusan kiszámolt adatmezőt kell kezelni:

Születési dátum (DateTime)

Ez a dátum nem valós, hanem a karakter többi tulajdonságából származtatott „abszurd”, determinisztikus érték. Ennek megvalósításához hozzunk létre egy `AbsurdDatumSzamitas` nevű paraméter nélküli `DateTime`-al visszatérő vügvényt az osztályon belül.

A `AbsurdDatumSzamitas` fügvény számítás menete:

Alap dátum meghatározása éves tartományból:
Egy determinisztikus „random” év kiválasztása a **[1800–2000]** tartományból. Ehez használjuk a `DetRandom` osztály Next() metódusát. Ennek első paramétere egy a karakter adataiból képzett string melyet a következőképpen kell előálítani:
Definiálunk az osztályon belül egy `GetDetSeed` nevű fügvényt melynek egy szöveg paramétere van. A visszatérési értéke pedig szöveg, tartalma **|** jelekkel elválaszva az osztály minden mezőjének értéke (ID, Név, Szint, Karakter_osztály, Életerő, Mana, Megitélés, Régió, Arany) + **|** jel után fűzve a paraméter.
Ekkor az alap dátum meghatározható DetRandom.Next eredménye lesz az alap dátum év része, a hónap és nap pedig legyen 1. hónap 1. napja. Ehez használjuk az imént megírt **GetDetSeed** fügvényt melynek paraméter stringje az intervallum szövege lesz: **“1800 2001”** formában.

Ezzel létrehoztuk az alap születési dátumot, majd ezt módosítani kell a következők szerint:
- Adjunk hozzá a következő képlettel kapott eredményt napokban: Szint * 17 + Életerő * 3 - Mana *2 + (Megitélés*50)
- Adjunk hozzá a következő képlettel kapott eredményt hónapokban: Életerő * Mana * (Karakter_osztály *1)

Megjegyzés:
Az enumok használatánál a számszerű értékével kell dolgozni.
A hónapok számításához használt képlet eredményét normalizálni kell úgy, hogy 50-nél le lehessen nagyobb.

3. Feladat – A születési dátum módosítása (ModositottDatum)

A már kiszámolt „abszurd” dátumot tovább kell módosítani . Ennek megvalósításához hozzunk létre egy fügvényt `ModositottDatum` melynek van egy bemeneti szám, illetve egy kimeneti szám paramétere, A visztérési típusa pedig Dátum.

A tartomány értékét az **`eltolasNap`** = Arany + bemeneti paraméter mod 30 Képlettel számoljuk, majd ezáltal a tényleges tartomány [-**`eltolasNap`**, **`eltolasNap`**] fog alakulni ahol **`eltolasNap`** az előző képlet eredménye.
A fügvény kimeneti paramétere ennek a számításnak lesz az eredménye vagyis a **`eltolasNap`** lesz.

Ezután használjuk a `DetRandom.Next` fügvényt melynek hash stringje itt a `GetDetSeed()` fügvényhívással képezzük (paraméter nélkül)

Ezután a kapott “Random” értéket összeadjuk az régi születési idővel majd visszaadjuk az így kapott dátumot.

Az elkészített fügvényt a karakterek beolvasását végző ciklusban a karakter objektum létrehozása után kell meghívni, a kimenő paraméterét pedig egy változóban gyűjteni mely összeget megadjuk bemenő paraméterben.

5. Feladat – A karakterek Dictionary-ben való tárolása

A kész Karakter obejktumokat **Dictionary<string, Karakter>** struktúrában tároljuk ahol a kulcs a karakter ID mezője.

6. Feladat – Lekérdezések és statisztikák (LINQ)

A feladathoz több, elkülönülten kezelendő lekérdezés tartozik.
Mindegyikhez elvárt a LINQ lambda használatával kell megvalósítani!

A) - Régiónkénti átlagos szint

Csoportosítani kell a karaktereket a Régió mező alapján, és számítani egy-egy csoport átlagos szintjét.

B) - Osztályonként a leggazdagabb karakter

Külön csoportonként (Warrior, Mage stb.) ki kell választani azt, aki a legtöbb arannyal rendelkezik.
Megjegyzés: Használjunk GroupBy-t és OrderByDescending-t a rendezéshez és csoportosításhoz.
a GroupBy helyes használata,

C) - Morális megítélés szerinti darabszám

Számoljuk meg, hány karakter tartozik Good / Neutral / Evil kategóriába.

D) - TOP 5 legerősebb karakter egyedi erő pont alapján

Az erőt a következő képlettel számoljuk: Életerő * 1.5 + Mana * 1.2 + Szint *5
A képlet eredménye szerint adjuk vissza az 5 legerősebb karaktert!

E) - A legfiatalabb 3 karakter

A legkésőbbi (legfiatalabb) születési dátum alapján visszaadni a 3 legfiatalabb karaktert.

7. Feladat – Új fájl generálása a feldolgozott karakterekből

Először a karakter osztályhoz definiáljuk a `ToFileLine` paraméter nélküli fügvényt. A fügvény célja hogy visszadja az osztályt olyan formában mint ahogy azt beolvastuk. Vagyis minden mezője **|** jelekkel van elválasztva, és a végén ott van a számított dátum is.

Írjuk ki az összes Karaktert egy **kimenő_karakterek.txt** nevő szöveges állományba.