| Dátum: | 2018. március 23. | **Szoftvertervezés és -fejlesztés II.** | |
|---|---|---|---|
| Név: | ……..... | Neptun kód: | |

A feladatra adott C# megoldást egy `neptunkod_nev` nevű könyvtárban helyezze el. A teljes könyvtárat tömörítse egyetlen zip fájlba, majd ezt töltse fel az oktató utasításai szerint.

A, Feladat.

Készítsen egy **IÚjraírható** nevű interfészt, ami az alábbiakat írja elő:

- *UtolsóÍrásDátuma* nevű, dátumot visszaadó csak olvasható tulajdonságot.
- *Újraírás(adat : szöveg)* nevű metódust.

Készítsen egy **DigitálisAdattároló** nevű osztályt az alábbiak szerint:

- Kívülről csak olvasható módon tárolja egy szöveg tömbben az adatokat.
- Csak egy konstruktora legyen, ami beállítja az adattároló tömb méretét.
- Rendelkezzen egy Írás(adat : szöveg) metódussal, ami a tömbben eltárolja a paraméterként kapott szöveget, ha megtelt a tároló, akkor egy eseményhívás történjen, ami tartalmaz egy referenciát az eseményt hívó objektumra, ha a megadott méretnél több írás érkezik, akkor dobjon egy **NincsÜresHely** kivételt, ami tartalmaz egy referenciát a kivételt dobó objektumra, és a maximum méretet.
- Legyen egy *SzabadHely* metódusa, ami visszaadja, hogy hány üres hely van még a tömbben.
- Az osztály valósítsa meg az IComparable interfészt, ami a szabad helyek száma szerint hasonlítson össze.

Készítsen egy **ÚjraírhatóAdattároló** nevű osztályt, amely a **DigitálisAdattároló** leszármazottja.

- Egészítse ki egy csak olvasható *lezárt* nevű mezővel, ha ez a mező igaz, akkor elveszíti az újraírható tulajdonságát.
- Írja felül az Írás metódust úgy, hogy a funkcionalitás ne változzon, de az eseményt ne hívja.
- Legyen egy *Lezárás* metódusa, ami lezárja az adattárolót és meghívja az eseményt, ha az adattároló üres a metódus hívásakor, akkor dobjon egy **ÜresAzAdattároló** kivételt, ami tartalmazza a maximum méretet.
- Valósítsa meg az **IÚjraírható** interfészt az alábbiak szerint.
    - Az *UtolsóÍrásDátuma* legyen az utolsó íráskor vagy újraíráskor aktuális dátum és idő.
    - Az *Újraírás* metódus, ha nincs üres hely, akkor a legrégebben beírt adatot írja felül.

Készítsen egy **ÚjraírhatóFeladat** nevű osztályt az alábbiak szerint:

- Konstruktorban beállítható a maximum pontszám és a feladat szövege illetve a maximum újraírhatóság száma.
- Legyen egy Válasz(válasz : szöveg) metódusa, amivel válasz adható a feladatra, a választ tárolja is el.
- Valósítsa meg az **IÚjraírható** interfészt az alábbiak szerint:
    - Az *UtolsóÍrásDátuma* legyen a legutolsó válaszadás dátuma.
    - Az *Újraírás* metódus írja felül a már megadott választ.

Minden osztálynak legyen egy *ToString()* metódusa is, ami visszaadja a lényeges adatokat.
Hozza létre a főprogramot:

- Hozzon létre egy tömböt és töltse fel tetszőleges *DigitálisAdattároló* és *ÚjraírhatóAdattároló* objektumokkal. Rendezze a tömb elemeit.
- Írjon mindegyikbe adatokat (kezelje a felmerülő kivételeket).
- Legalább egy Adattárolót töltsön fel teljesen és írja, ki ha kész az írás.
- Hozzon létre egy *IÚjraírható* tömböt másolja bele az *ÚjraírhatóAdattároló* példányokat, és tegyen bele néhány *ÚjraírhatóFeladat* objektumot is.
- A tömb minden elemét írja újra többször is, és írja ki, ha egy elemet nem lehet többször újraírni.

*A feladat megoldása közben vegye figyelembe a tanult OOP alapelveket: egységbezárás, láthatóság (ős és leszármazott között is), polimorfizmus stb.!*

**Rendelkezésre álló idő : 120 perc**