Feladat – Kiszh 3 – FONTOS INFÓK

- A feladat során alkalmazd a megtanult objektum-orientáltsági elveket!
- A megadott példakódon ne módosíts, hacsak a feladat nem kéri! Ez alól kivétel a Program.cs elején lévő kommentek (#define). A megoldásnak ezen fájlokkal kell mennie, hiszen az ellenőrzés során a Moodle biztosítja őket.
- Figyelj a kiírás megfelelő formátumára, szóközökre, új sorokra! Az automata javító csak a tényleges kimenetet látja, nem tudja mit akartál.
- Minden pont értékeléséhez szükséges, hogy az adott ponthoz tartozó #define szerepeljen a Program.cs fájl elején (#define PART1 az 1. feladathoz, #define PART4 a 4. feladathoz, stb.).
- Csak olyan kódot tölts fel moodle-be, ami nálad fordul. Ami nálad nem fordul, a Moodle-ben sem fog.
- A fájlokat nem tömörítve kell feltölteni, hanem önmagukban (drag-and-drop-pal egyszerre be lehet húzni az összeset).
- Figyelj rá, hogy a fájlnevek pontosan azok legyenek, amiket a feladat kér!
- A Moodle által használt fordító nem feltétlenül támogatja az újabb nyelvi fejlesztéseket. Amit az órákon tanultunk és használtunk, az mind működik, de az újabb fejlesztések nem biztos.

Feladat – Kiszh 3

- A feladat elkezdéséhez a mellékelt projekt tartalmaz kódokat. A megadott **Program.cs** fájl a tesztelésben segít. Módosítani csak az elején lévő #define-okon lehet, amik a teszteket aktiválják.
- A feladatban egy olyan osztályt kell készíteni, amely tetszőleges típusú adatokat tárol, de korlátozott mennyiségben.

A feladatok:

1. Készíts egy generikus **LimitedStorage** osztályt, amely egy típus paraméterrel rendelkezik, és ilyen típusú értékeket tárol (például listában). Konstruktorában egy egész számot várjon, ami a tárolt elemek maximális száma. Ez a maximális méret később legyen lekérdezhető a *MaxSize* property-n keresztül. **(2 pont)**
2. Az osztálynak legyen egy *Add* metódusa, amely a paraméterben kapott értéket hozzáadja a belső tárolóhoz, de csak akkor, ha még nem érte el a maximális méretet. Az aktuálisan tárolt elemek számát lehessen a *Size* property segítségével lekérdezni. **(2 pont)**
3. Legyen az osztálynak egy *Item* metódusa, amely paraméterben egy indexet vár, és visszatér az adott helyen lévő tárolt elemmel. **(1 pont)**
    a. Amennyiben a kapott index nem érvényes, a metódus térjen vissza a tárolt típus alapértelmezett értékével. **(1 pont)**
4. Legyen az osztálynak egy *Extend* metódusa, amelynek paraméterben át lehet adni egy olyan listát, ami vagy ugyanolyan típusokat tárol, mint az osztály, vagy abból származó típusokat. Tehát, ha az adott **LimitedStorage** objektum **Base** típusú értékeket tárol, akkor az *Extend* metódus paraméterben fogadjon el **List<Base>** típust, továbbá bármilyen másik **List<Derived>** típust is, ahol a **Derived** osztály a **Base**-ből származik. A metódus adja hozzá az elemeket a saját listájához. Ha nem fér el mind, akkor csak annyit adjon hozzá, amennyi elfér. **(2 pont)**
5. Készíts egy Extra osztályt, aminek egyetlen, statikus metódusa lesz. Ez a *LoadStorageList* metódus egy fájlnevet kap, és **double** típusokat tároló **LimitedStorage**-ek listájával tér vissza. A metódus olvassa be az adatokat a fájlból. **(2 pont)**

    A fájl szerkezete: A fájl több **LimitedStorage** adatait tárolja: először van egy sor egy darab egész számmal, ami megmondja a maximális méretet, majd a következő sorban szóközökkel elválasztva lebegőpontos értékek (ezek a tárol számok). A tárolóban lehet kevesebb szám, mint a maximális méret. Például vegyünk egy fájlt az alábbi sorokkal:

    3
    3.4 6.5 7.8
    8
    2.4 1.5 8.2 7.1 5.1
    6
    2.4 4.6 1.5 7.2

    Ebben három tároló van, az első legfeljebb három elemű, és mind a három fel is van töltve az alatta lévő számokkal. A második legfeljebb 8 elemű, de ebből csak 5-öt tárol, a harmadik pedig 6 elemű, amiből ténylegesen 4-et tárol.