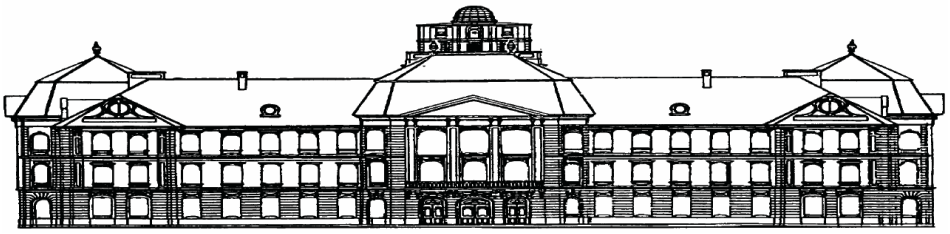

Eszterházy Károly Katolikus Egyetem

Magasszintű programozási nyelvek II.

2022/2023 tavaszi félév

1. zárthelyi dolgozat (levelező tagozat)

Általános információk

Hozzon létre egy *Console Application (.NET Framework)* projectet a **C:\temp** mappába, melynek neve az Ön **neptunkódja**! A kijelölt feladatokat ebben implementálja C# nyelven, majd munkája végeztével ugyanilyen néven tömörítse a programot és töltse föl a megadott címre!

A kicsomagolás után nem beazonosítható dolgozatok automatikusan elégtelenek, ezek újraírására nincs lehetőség!

A dolgozat írása közben csak és kizárólag a **C:\temp**, valamint a **Letöltések** mappák lehetnek megnyitva a projecten kívül!

A dolgozat megírása során semmilyen segédanyagot, órai feladatot nem használhat!

A dolgozat megírása során semmilyen kommunikációs csatorna (levelezőrendszer, chat, stb. . .) nem lehet nyitva és nem léphet kapcsolatba senkivel.

Bármilyen a fentiekre utaló magatartás esetén azonnal elégtelen a zh eredménye. Ennek újra írására nincsen lehetőség!

Törekedjen részmegoldásokra! Ha valamit nem tud megoldani (pl egy property-t nem tud teljes mértékben kidolgozni), hagyja ott annak szignatúráját, vagy hozza olyan állapotra, hogy tudjon vele tovább dolgozni!

A dolgozat megírására 2 órája van, figyeljen az időbeosztásra!

Feladatkiírás

A zárthelyi dolgozatban egy fiktív banknál lekötött bankbetétek tárolását és azok lekérdezéseit menedzselő alkalmazást kell fejlesztenie.

Az osztályok amelyeket implementálnia kell majd az alábbiak szerint épülnek föl.

- Bankbetét (ősosztály)
  - Befektetési jegy (a bankbetét gyermekosztálya)
- Bank (konténerosztály a bankbetétek nyilvántartására)

Bankbetét

Hozzon létre osztályt **Bankbetet** néven, és implementálja benne az alábbiakat!

Mezők és property-k

- **Azonosító**: szöveges adat, az osztályon kívül nem módosítható. Értéke nem lehet null vagy üres string. Csak számokat, az angol ABC nagybetűit és '/' karaktert tartalmazhat, de ez nem lehet az utolsó karakter. Az első karakter csak angol nagybetű lehet!
- **Kezdőtőke**: egész típusú adat, kívülről nem módosítható. Értéke az $[50000; 100000000]$ intervallumban érvényes. Értéke csak egyszer adható meg!
- **Kamatláb**: valós típusú adat, kívülről nem módosítható. Értéke a $[0{,}01; 0{,}99]$ intervallumban érvényes.
- **Lekötés ideje** (években mérve): egész értékű adat, kívülről nem módosítható. Értéke az $[1; 20]$ intervallumban érvényes. Ennél az adatnál írja meg a klasszikus property metódusokat a nyelvi szinten támogatott property-k helyett! A getter esetén alkalmazzon késői kötést!
- **Hosszútávú lekötés-e**: logikai típusú számított property. Értéke igaz, ha a lekötés ideje hosszabb, mint 10 év!

Konstruktorok

- Készítsen konstruktort, mely az összes adatot (azonosító, kezdőtőke, kamatláb, lekötés ideje) bekéri és eltárolja.
- Készítsen konstruktort, mely minden adatot bekér a lekötés idején kívül! Ezt az előző konstruktor meghívásával automatikusan 5 évre állítja!

Metódusok

- Készítsen egész értékkel visszatérő metódust **Hozam** néven, mely paraméterben vár egy egész számot, ami a lekötéstől számított éveket jelenti. Alkalmazzon késői kötést!

  Amennyiben a paraméterben kapott évek száma nagyobb, vagy egyenlő, mint a lekötés ideje, úgy a klasszikus kamatos kamat számítással kapjuk a metódus által visszaadott eredményt:

  $$kezdotoke \cdot \left(1 + \frac{kamatlab}{100}\right)^{ev}$$

  Ellenkező esetben a fent számított értékből büntetésként le kell vonni a fennmaradó évek és a kezdőtőke 100-adrészének a szorzatát!
- Írja felül a **ToString()** metódust úgy, hogy minden tárolt és számított adat ízlésesen megjelenjen!
- Írja felül az **Equals()** metódust! Két bankbetét akkor egyezik meg, ha az azonosítójuk megegyezik!

Befektetési jegy

Hozzon létre osztályt **BefektetesiJegy** néven! Jelölje meg ősként a **Bankbetet** osztályt és implementálja benne az alábbiakat!

Mezők és property-k

- **Részvény alap**: szöveges adat, az osztályon kívül nem módosítható. Értéke nem lehet null vagy üres string.
- **Minimum hozam**: valós típusú adat. Értéke a [0,01; 0,5] intervallumban érvényes.
- **Maximum hozam**: valós típusú adat. Értéke a [*minimum hozam*; 0,99] intervallumban érvényes.
- **Kockázat**: felsorolás típusú adat. Lehetséges értékei: **alacsony, mersekelt, magas** Elég csak a property szignatúráját megírni!

Konstruktorok

- Készítsen konstruktort, mely a következő adatokat: **azonosító, kezdőtőke, részvény alap, minimum hozam, maximum hozam, kockázat**. Az ős konstruktorának hívásával beállítja a kamatláb értékét 0,01-re. Minden az ebben az osztályban definiált adatot eltárol.
- Készítsen konstruktort, mely minden olyan adatot bekér, mint az előző konstruktor, kivéve a kockázatot, melynek értékét az előző konstruktor hívásával **mersekelt**-re állít.

Metódusok

- Írja felül a **Bankbetet** osztályban definiált, a **lekötés ideje** mezőhöz definiált **get**tert! A metódus csak dobjon egy kivételt, melynek üzenete az, hogy befektetési jegy esetén a lekötés ideje nem értelmezhető információ!
- Írja felül a **Bankbetet** osztályban definiált **Hozam** metódust az alábbiak szerint!
  - Befektetési jegy esetén nincs büntetés, akármilyen korán vesszük ki az összeget!
  - A kamatláb helyett minden évben generáljon egy véletlen számot a minimum- és maximum hozam értékei közé. *Segítség: Ebben az esetben a hatványozás helyett egy ciklust kell alkalmaznia, ami annyiszor fut le, ahány év után vesszük ki az összeget!*
- Írja felül a **ToString()** metódust úgy, hogy minden tárolt és számított adat ízlésesen megjelenjen!

Bank

Hozzon létre osztályt **Bank** néven, és implementálja benne az alábbiakat!

Mezők és property-k

- **Bankbetétek**: kívülről nem hozzáférhető bankbetéteket tartalmazó lista. Ne írjon hozzá közvetlenül property-t!
- **Bank neve**: szöveges adat, nem kell ellenőriznie semmit! Elég a property szignatúráját megírni!

Metódusok

- **Bankbetét hozzáadása**: metódus, mely paraméterben vár egy bankbetét objektumot. Ha az már szerepel az adatbázisunkban (a lista), akkor dobjon kivételt, ellenkező esetben mentse el a listába!

Lekérdezések

- **Bankbetétek magas kezdőtőkével**: csak olvasható property, mely összegyűjti egy listába és visszaadja az 5 *millió* forintnál magasabb kezdőtőkével rendelkező bankbetéteket!
- **Befektetési jegyek adott hozam fölött**: metódus, mely összegyűjti egy listába és visszaadja azon befektetési jegyeket, melyek a paraméterben kapott valós értéktől nagyobb minimális hozammal rendelkeznek!
- **Indexelő**: Megkeresi és visszaadja az indexként kapott azonosítóval rendelkező bankbetétet!
- **(Extra feladat plusz pontért) Legmagasabb minimális hozammal rendelkező magas kockázatú befektetési jegy**: csak olvasható property, mely megkeresi és visszaadja azt a befektetési jegyet, melynek a legmagasabb a minimális hozama a magas kockázatúak között!

Főprogram

A főprogram megírásához felhasználhatja az alábbi linken található forráskódot:

`https://pastebin.com/98v89xsJ`

- Hozzon létre egy **Bank** példányt, és állítsa be a nevét a **"<Vezetéknév> <Keresztnév> bankja"** értékre! (Természetesen itt mindenki írja be a saját nevét a helyőrzők helyére!)
- Olvassa be a mellékelt **bankbetetek.csv** pontosvesszővel tagolt fájlt, és annak adatai alapján töltse fel a bankot bankbetét-, illetve befektetési jegy típusú objektumokkal! Az input fájlban az adatok a mezők ebben a dokumentumban definiált sorrendje szerint vannak elhelyezve!
- Jelenítse meg az összes a konténerosztályban implementált lekérdezés eredményét!

Szöveg konvertálása enummá:

```
MyEnum myEnum = (MyEnum)Enum.Parse(typeof(MyEnum), "text");
```