Órai feladat – Hasító táblázat

Mivel a leadott feladatokat lehetséges, hogy automatizált módon fogjuk ellenőrizni, ezért kérünk mindenkit, hogy a lenti (ékezetek nélküli) elnevezéseket tartsa meg. Szükség esetén további mezőket fel lehet venni, bár ezekre általában nincs szükség. Ugyanígy kérünk mindenkit, hogy próbálja meg önállóan megoldani a feladatot, mivel csak így fog bármit tanulni belőle. Szükség esetén persze a laborvezetőket nyugodtan meg lehet keresni, akik segíteni fognak.

Bevezetés

A feladat egy egyszerű banki rendszer elkészítése. A bankban az ügyfelek fiókjai egy hasító táblában vannak tárolva. A tároláshoz használt kulcs a számlaszám, melynek segítségével egy fiók megkeresését gyorsan el lehet végezni. Nap végén a napi utalásokról kapunk egy naplófájlt, ami az aznapi összes utalást tartalmazza felsorolásszerűen. Feladatunk az utalás napló feldolgozása, utalások végrehajtása, hogy végül megkapjuk a zárás utáni egyenlegeket.

Osztályleírások

Az egyes osztályokból a *Console* osztály metódusait meghívni **tilos**, azt csak a *Program* osztályon belül tegye!

class BankHashSet<K,T>

Hasító tábla osztály, a kulcsütközések kezelését láncolt altáblák segítségével kell megoldani.

class BankHashSetItem ♦ Beágyazott privát adatosztály, mely K és T típusra tartalmaz publikus referenciát „key” és „content” néven.

Leírás ♦

- Legyen generikus, kívülről lehessen meghatározni a kulcs és tartalom típusát is.
- Rendelkezzen egy belső publikus „HashCallback” delagate típussal, ami a következő függvény szignatúrát írja elő: **int HashCallback(K key, int size)**.
- Készítsen egy *HashCallback* szignatúrának megfelelő „DefaultHashing” statikus [1] privát függvényt. Használja a „key” paraméter „GetHashCode” metódusát. Az így kapott számot a „size”-zal történő maradékképzéssel hasítsa a megfelelő tartományba. Ne legyenek negatív értékek.
- „_contents” Tömb, *BankHashSetItem* listák tömbje, privát adattag.
- „_size” *egész szám* típusú privát adattag.
- „HashFunction” *HashCallback* függvény típusú privát adattag.

[1] Azért statikus, mert paraméterként lesz használva a lenti konstruktorban, tehát fordítási időben szükségünk van erre a metódusra.

- Legyen egy konstruktor ami beállítja az előbb felsorolt három adattagot. „_contents” tömb mérete értelemszerűen a „_size”-nak megfelelő legyen. Ne feledkezzen meg a „_contents” alapértelmezett listáinak létrehozásairól. Kezelje azt az esetet, amennyiben a konstruktor „hashFunction” paraméterét valaki „null”-al hívja meg, ez esetben a „HashFunction” legyen a „DefaultHashing” függvény.
- Legyen egy másik, paraméter nélküli konstruktor is, ami az előző konstruktort hívja meg fix paraméterekkel. „HashFunction”-nek a „DefaultHashing” legyen beállítva. A „_size” legyen 100.

Publikus metódusok ♦

- **void Insert(K key, T content)** Új elem beszúrása a hasító táblába. Paraméterként kapott értékekből készítsen *BankHashSetItem*-et, a „HashFunction” segítségével állítsa elő a tároláshoz szükséges indexet, majd a megfelelő listába illessze be az újonnan készített elemet.
- **T Find(K key)** Kulcs alapján keresse ki és térjen vissza az adott helyen tárolt tartalommal. Amennyiben nem található a keresett kulcsú elem, dobjon kivételt megfelelő szöveggel.
- **T this[K key]** Valósítsa meg az indexelő operátort.[2] Szögletes zárójel és kulcs index segítségével lehessen az indexelt tartalmat írni, olvasni.

class BankAccount

Leírás ♦

- „AccountNumber” *szöveges* típusú, kívülről csak olvasható tulajdonság.
- „Money” *64bit-es lebegőpontos* típusú, csak olvasható tulajdonság.
- Készítsen konstruktort ami beállítja ezeket.

Publikus metódusok ♦

- **void Deposit(double amount)** A megadott összeggel növeli a pénzt.
- **void Withdraw(double amount)** A megadott összeggel csökkenti a pénzt.

class Bank

Leírás ♦

- **BankHashSet<string, BankAccount> _accounts** privát adattag.
- Legyen egy konstruktora, ami a „BankHashSet” konstruktorával egyező paramétereket várjon. A bejövő paraméterek alapján hozzon létre egy „BankHashSet” példányt és állítsa be a „_accounts” adattagot.[3]
- Legyen egy paraméter nélküli konstruktor is, mely a hasító tábla paraméter nélküli konstruktorát meghívja.

[2] A tömböket számokkal szoktuk indexelni, ehhez használjuk a szögletes zárójel operátort. Jelen esetbe a „K” (kulcs) típus lesz, amivel indexelünk. Majd „T” típusú példányt adunk vissza. Ennél bővebb segítséget a hasító táblákhoz tartozó laborvideókban, vagy a hivatkozott oldalon talál: LINK

[3] Nem szerencsés osztályok konstruktorait egymástól függővé tenni. Akit zavar megteheti, hogy létrehoz egy adatosztályt, ami az összes konstruktor paramétert összefogja, így az esetleges változtatásokra is felkészíthetjük a programunkat, így csak egy helyen kell adott esetben módosítást végezni.

Publikus metódusok ♦

- **void Transaction(string from, string to, double amount)** Végezze le a két fiók közötti pénzmozgatási tranzakciót. A „from" kulcsú fiók utal pénzt a „to" kulcsú fióknak az „amount" paraméter mértékében.
- **void RegisterAccount(string AccountNumber, double deposit)** Beszúrja a fiókot a hasítótáblázatba a megadott kezdeti összeggel.

Tesztelés

Olvassa be és dolgozza fel a „bank_log.txt" fájlt. A fájl első felében soronként az összes fiók azonosító száma és kezdő összege „ , " vesszővel elválasztva található. Ezután egy üres sor, végül szintén soronként az egyes fiókok közötti tranzakciók vannak felsorolva. Az egyes tranzakciók formátuma szintén „ , ' vesszővel tagolt elemekből áll, a következő sorrendben értelmezendő: „kitől", „kinek", „mennyit".

- Tesztelés során törekedjen a egyes feladatokat logikusan külön statikus metódusokba szervezni, ne minden a *Main*-be ömlesztve legyen. (pl. külön fájl feldolgozás metódust, stb)
- Hozzon létre egy példányt a „Bank" osztályból.
- Vegye fel a fiókokat, amik a fáj elején találhatóak.
- Végezze el az összes felsorolt tranzakciót.
- Írja ki, hogy a tranzakciók után mennyi pénz van az egyes fiókokban. Ehhez oldja meg, hogy valahogy lelehessen kérni az összes fiókot a „Bank"-ból (és annak hasító táblájából).

Opcionális

- Egészítse ki a bankot egy eseménnyel, mely a tranzakciók bekövetkezésekor meghívja a feliratkozott metódusokat. Ennek a segítségével kilehessen írni, hogy honnan, hova, mennyi pénz mozgott, illetve a számlákon jelenleg mennyi pénz van.
- Próbáljon ki különböző hasítási módszereket, nézze meg melyikkel, mennyi ütközés történik.
- Próbáljon ki különböző hasító tábla méreteket (2 hatványait, 10 hatványait, nagy prímszámokat), nézze meg melyikkel ,mennyi ütközés történik.