Feladat

- A feladat elkezdéséhez a mellékelt projekt tartalmaz **main** függvényt. A teszteléshez a **main** függvényben a nem használt kódokat ki lehet kommentezni, de módosítani azokat nem szabad. A feladathoz csak azokat a függvényeket készítsd el, amiket a feladat kér, illetve ami ezen felül feltétlenül szükséges a helyes működéshez.
- Ügyelj arra, hogy minden lefoglalt memória kerüljön megfelelően felszabadításra.
- A nem forduló kód 0 pont.
- A feladatban egy banki ügyfelet és a hozzá tartozó különböző számlákat kezelünk.

- Adott a SzamlaSeged osztály, amelynek a szamjegyek statikus metódusa visszaadja a 2x8-as formátumú számlaszám 16 számjegyét egy tömbben, a megoldás során ezt majd használni kell.

1) Készíts egy **Tranzakcio** osztályt, amely egy banki tranzakció adatait tárolja: ellenoldali számlaszám (szöveg), tranzakció összege (egész) és, hogy jóváírás-e a tranzakció (logikai érték, igaz esetén jóváírás, hamis esetben terhelés). A konstruktor várja az adatokat, és legyen hozzájuk getter (*getSzamlaszam*, *getOsszeg*, *getJovairas*). **(3 pont)**
    a) Készíts az osztályhoz egy * unáris operátort, ami logikai értékkel tér vissza: igaz ha a tranzakció érvényes, hamis ha érvénytelen. Egy tranzakció érvénytelen, ha az összeg nulla vagy negatív, illetve ha a számlaszám nem CDV helyes (CDV helyesség ellenőrzése a feladat után van leírva)**(2 pont+4pont)**
    b) Készíts egy **<<** operátort az osztályhoz, amellyel az osztály tartalmát tetszőleges kimeneti folyamra meg lehet jeleníteni. **(1 pont)**

2) Készíts egy **Szamla** osztályt, ami egy számlát reprezentál. Legyen egy számlaszám adattagja (szöveg), illetve egy egyenleg adattagja (egész). Az adattagokat a konstuktor állítsa be, getterek nem szükségesek. **(3 pont)**
    a Legyen az osztálynak egy tisztán virtuális *vegrehajt* metódusa, ami egy Tranzakció objektumot vár paraméterként és logikai értékkel tér vissza (sikerült-e a végrehajtás vagy nem). A működést majd a származtatott osztályok valósítják meg. **(1 pont)**
    b Legyen az osztálynak egy virtuális *kiir* függvénye, ami megjeleníti a számla adatait a konzolon. **(1 pont)**
3) Származtass egy **FolyoSzamla** osztályt a **Szamla**-ból, amely új adatként tárolja a számla megnevezését (szöveg). A konstruktor várja az összes adatot, getter függvények nem szükségesek. **(2 pont)**
    a A *vegrehajt* függvény először ellenőrizze, hogy érvényes-e a tranzakció. Jóváírás esetén növeld a számla egyenlegét a tranzakció összegével. Terhelésnél pedig csak akkor csökkentsd az egyenleget a tranzakció összegével ha nem lesz negatív az egyenleg utána. Sikeres végrehajtásnál igazzal térjen vissza a függvény, egyébként hamissal. **(3 pont)**

b Fejtsd ki az ősosztály *kiir* függvényét úgy, hogy írja ki a számla összes adatát az ősosztály metódusát is felhasználva. **(1 pont)**

4) Származtass egy **HitelSzamla** osztályt a **Szamla**-ból, amely új adatként tárolja a maximális hitelkeret értékét. Az összes adatot a konstruktor várja, getterek nem szükségesek. **(2 pont)**
   a A *vegrehajt* függvény akkor hajtsa végre a tranzakciót ha az egyrészt érvényes. Másrészt jóváírás esetén növelje az egyenleget a tranzakció összegével, terhelés esetén pedig csak akkor csökkentse az egyenleget, ha az egyenleghez a hitelkeretet is hozzáadva, a levonás után nem lesz negatív az egyenleg. **(3 pont)**
   b Fejtsd ki az ősosztály *kiir* függvényét úgy, hogy írja ki a számla összes adatát az ősosztály metódusát is felhasználva. **(1 pont)**
5) Készíts egy **Ugyfel** osztályt, amely egy ügyfelet reprezentál és tetszőleges mennyiségű és típusú számlát kezel. Tároljuk még az ügyfél nevét (szöveg), amit a konstruktor kap meg és állít be, getter nem szükséges. A számlákat úgy kell tárolni, hogy mindig pontosan annyi hely legyen a tömbben, amennyi kell. Hozd létre az ennek megfelelő adattagokat. **(4 pont)**
   a Készíts egy kiir függvényt, ami kiírja az ügyfél nevét és a számlák adatait a kiir függvényük segítségével. **(2 pont)**
   b Az osztályhoz a **+=** operátorral lehessen egy újabb számlát hozzáadni, a *main*-ben látható módon. Ellenőrizni kell azt, hogy az ügyféknek legfeljebb 1 db hitelszámlája lehet, folyószámlája viszont bárennyi.**(4 pont)**
   c Az osztály destruktora szabadítsa fel a tárolt számlákat. **(2 pont)**
   d Készíts egy **alkalmaz** függvényt, ami egy tranzakciót kap paraméterként, a tárolt számlák közül az elsőn végrehajtja, amelyen végrehajtható a tranzakció és visszatér annak a számlának a címével. **(3 pont)**
   e Készíts az osztályhoz egy **[ ]** operátort, ami megkap egy tömb indexet, és visszatér az adott indexen tárolt számla címével. Érvénytelen index esetén null mutatóval térjen vissza. **(2 pont)**

CDV ellenőrzés

- A számjegyeket balról jobbra haladva egyenként meg kell szorozni a 9,7,3,1,9,7,3,1,... számokkal (tehát balról az első számjegyet kilenccel, a másodikat héttel, stb. kell megszorozni(,

- a szorzatokat össze kell adni,

- majd az eredmény utolsó számjegyét ki kell vonni 10-ból. A kapott szám a CDV. (Ha az összeadás eredményének az utolsó száma 0, akkor a CDV is nulla).

- A számlaszámok esetén, a 8. és az utolsó karakteren is található CDV. Ez tehát azt jelenti, hogy a számlaszámoknál külön számítást kell végezni az első hét jegyre és külön a 9-15. pozíciókra ha a számlaszám 16 hosszú.