
November 4

# Szoftverfejlesztés gyakorlat – 10. óra

# 2019

OOP – Összetett OOP tervezéses feladat, Öröklődés

Feladatlap

# 1. Feladat – Banki szolgáltatások

Készíts **Tulajdonos** osztályt

- Kívülről írható/olvasható formában tárolja el a tulajdonos nevét
- Biztonsági okokból ne lehessen belőle származtatni

Készíts **BankiSzolgáltatás** osztályt

- A konstruktorban lehessen megadni a tulajdonost, ez a későbbiekben csak olvasható legyen
- Ebből az osztályból ne lehessen közvetlenül példányosítani

Készíts **Szamla** osztályt

- Legyen a BankiSzolgáltatásosztály leszármazottja
- Konstruktorában lehessen megadni a tulajdonost
- Kívülről csak olvasható formában tárolja el az aktuális egyenleget
- Egy Befizet(összeg) metódussal lehessen növelni az egyenleget
- Legyen egy hasonló paraméterű, de nem implementált Kivesz(összeg) metódusa is, aminek a visszatérési értéke egy logikai érték

Készíts **HitelSzámla** osztály

- Legyen a Számlaosztály leszármazottja
- A konstruktorban lehessen megadni a tulajdonos mellett a hitelkeret összegét, a későbbiekben ez csak olvasható legyen
- Valósítsa meg úgy a Kivesz(összeg) metódust, hogy csak a hitelkeret mértékéig engedjen negatív számla egyenleget. Ellenkező esetben ne csökkentse az egyenleget és hamis visszatérési értékkel jelezze, hogy nem sikerült a kivétel

Készíts **MegtakarításiSzámla** osztályt

- Legyen a Számlaosztály leszármazottja
- Kívülről írható/olvasható formában tárolja el a kamat mértékét
- Az osztály egy statikus mezőjében tárolja el az alapértelmezett kamatot. Egy új megtakarítási számla létrehozásakor ez legyen a kamat kezdőértéke
- A Kivesz(összeg) metódus ne engedje 0 alá csökkenni az egyenleget, visszatérési értéke jelezze, hogy sikerült-e a kivét
- Legyen egy Kamatjóváírás() metódusa, ami jóváírja az esedékes kamatot

Készíts **Kártya** osztályt

- Legyen a BankiSzolgáltatásosztály leszármazottja
- A konstruktorban lehessen megadni a tulajdonos mellett a hozzá tartozó mögöttes számlát, illetve a kártya számát
- A kártyaszám legyen kívülről olvasható, a mögöttes számla nem módosítható
- Készítsen egy Vásárlás(összeg)metódust, ami a paraméterként megadott összeggel megpróbálja csökkenteni a mögöttes számla egyenlegét, és visszatérési értéke legyen ennek sikeressége

Egészítsd ki a **Számla** osztályt

- Egészítse ki a Számlaosztály egy ÚjKártya(kártyaszám)metódussal, amely a leendő kártyaszámot várja paraméterként
- A metódus hozzon létre egy új kártyát (az aktuális számlát és annak tulajdonosát adva meg a kártya adataiként) és legyen ez a metódus visszatérési értéke

Készíts **Bank** osztályt

- Tároljon el tetszőleges számú számlát, ezek maximális számát a Bankkonstruktorában lehessen megadni
- Legyen egy Számlanyitás(tulajdonos, hitelkeret)metódusa, amelynek paraméterei egy Tulajdonos objektum és egy hitelkeret összeg. A hitelkeret összegének megfelelően hozzon létre hitel vagy megtakarítási számlát, ezt tárolja el, és ez legyen a metódus visszatérési értéke is
- Legyen egy Összegyenleg(Tulajdonos)metódusa, amely visszaadja a paraméterként átadott tulajdonos számláinak összegyenlegét
- Legyen egy LegnagyobbEgyenlegűSzámla(Tulajdonos)metódusa, amely visszaadja a megadott tulajdonos legnagyobb egyenlegű számláját
- Legyen egy Összhitelkeret()metódusa, amely visszaadja a bank által az összes ügyfélnek adott hitelkeretek összegét

A fenti osztályok implementálását követően hozzon létre példa Tulajdonos és Bank objektumokat, majd próbáld ki a fenti funkciók működését

## 2. Nagykövetség és diplomata szimulátor

A feladatban egy ország nagykövetségét és az ott dolgozó diplomatákat fogjuk ábrázolni. A diplomatákról számon tartjuk nevüket és az általuk beszélt idegen nyelveket. A diplomaták neveit és a beszélt nyelveket String-ként ábrázoljuk.

A feladatban a government.Diplomat és government.Embassy osztályokat fogjuk több lépésben megvalósítani. Az előbbi egy nagykövetségen dolgozó diplomatát, az utóbbi egy nagykövetséget ábrázol.

**government.Diplomat** osztály:

Szerepeljenek benne az alábbi rejtett adattagok.

- Egy String típusú név.
- Egy String tömb, mely a beszélt nyelveket tárolja.

Az osztályhoz készítsen egy rejtett konstruktort, mely paraméterül várja a két adattag kezdeti értékét, és inicializálja az adattagokat.

Valósítson meg egy osztályszintű **make()** nyilvános metódust, mely ugyanazokat a paramétereket várja, mint a konstruktor, és egy Diplomat objektumot ad vissza. A metódus elvégzi az alábbi ellenőrzéseket, majd létrehozza és visszaadja az objektumot, ha minden feltétel teljesül, különben egy null referenciát ad vissza.

- A név nem egy üres szöveg.
- A név pontosan két részből áll, szóközzel elválasztva.
- A két rész egyike sem üres szöveg, mindkettő egy-egy nagy kezdőbetűvel kezdődik. Ennek leellenőrzéséhez használjuk a Character.isUpperCase() metódust.
- Figyeljünk arra, hogy a létrehozott objektum belső állapota ne szivárogjon ki.

Készíts egy **getName()** paraméter nélküli metódust, mely visszaadja a diplomata nevét.

Készíts egy osztályszintű, nyilvános JAMES_BOND adattagot, mely egy diplomatát ábrázol, akinek a neve James Bond, és beszéli a következő nyelveket: ,,English'', ,,German'', ,,French''.

Emellett valósítsuk meg az alábbi nyilvános metódusokat is:

- Egy **speaks()** metódust, mely egy nyelvet vár paraméterül, és logikai értékben adja vissza, hogy a diplomata beszéli-e a nyelvet. Feltesszük, hogy a paraméter nem null.
- Egy **getLanguages()** paraméter nélküli metódust, mely visszaadja a beszélt nyelvek tömbjét. Figyeljünk arra, hogy a metódus ne szivárogtassa ki a belső állapotot.
- Egy **toString()** metódust, mely visszaadja a diplomata szöveges ábrázolását a következő alakban: Diplomat(<név>, <nyelvek>). Például Diplomat(Colin Powell, [English, French]). A nyelvek formátuma tetszőleges, azonban legalább egy szóközzel legyenek elválasztva.

**government.Embassy** osztály:

Vegyük fel az alábbi rejtett adattagokat:

- Egy String típusú országnevet, a nagykövetség országát.
- Diplomat objektumok sorozatát, mely a nagykövetségen dolgozó diplomatákat tárolja.
- Valósítsuk meg az alábbi metódusokat is a government.Embassy osztályban.

Egy konstruktort, mely paraméterül várja a nagykövetség országának nevét és egy Diplomat objektumokból álló tömböt, mely utóbbi a nagykövetségen dolgozó diplomatákat tárolja. A konstruktor beállítja az adattagok kezdeti értékét. Ügyeljünk arra, hogy a létrehozott objektum belső állapota ne szivárogjon ki.

Egy **spokenLanguages()** paraméter nélküli metódust, mely visszaadja a nagykövetségen dolgozó diplomaták által beszélt nyelvek listáját, ismétlés nélkül.

Egy **spokenBy()** metódust, mely egy nyelvet vár paraméterül, és visszaadja azon Diplomat objektumok listáját, melyek beszélik a nyelvet.

Egy **delegation()** metódust, mely nyelvek tömbjét várja paraméterül, és összeállít egy delegációt egy országba, ahol a hivatalos nyelveket a paraméter tárolja. A delegációt úgy kell összeállítani, hogy minden nyelvet beszélje legalább egy ember. Nem kell törekedni a legkisebb méretű delegáció összeállítására, azonban kerüljük el az ismétléseket. A delegációt nevek listájaként adjuk vissza. Feltesszük, hogy minden nyelvet beszél legalább egy diplomata, és a paraméter legalább egy nyelvet tartalmaz.

## 3. Feladat – Kávéfőző

A **kávéfőző** a következő metódusokkal rendelkezzen:

- feltolt(mivel, mennyit) - A kávéfőzőbe maximum 10 adag vizet, és 6 adag kávét lehet tölteni. Vizet bármikor utánatölthetsz, kávét csak akkor, ha nem használt. A kávéfőzőt fel lehessen tölteni vízzel és kávéval a kapacitásnak megfelelően.
- kiurit(mit) - A kávéfőzőből ki lehessen üríteni a használt kávét vagy vizet.
- foz() - A feltöltött kávéfőzővel lehessen kávét főzni. A főzéskor mindig annyi vizet használ fel, ahány adag kávé van benne. A kávéfőző rendelkezzen egy edénnyel, ami főzéskor a kész kávét tárolja. Főzéskor írd ki, hány adag kávét főztél.

Az **edény** a következő metódusokkal rendelkezzen:

- kostol() – megadja, hogy milyen kávé van benne. Ha főzéskor a víz kevesebb, mint amennyi adag kávé van benne, a lefőtt kávé legyen túl erős. Ha a kávé használt volt, akkor a kávé legyen pocsék.
- kiont() – kiönti a kész kávét

Ha az edényt nem üríted ki, akkor a benne lévő jó kávét el tudja rontani az esetlegesen "ráfőzött" rossz, vagyis ha nem ürítetted ki a használt kávét, vagy túl kevés vízzel főzted, valamint a kész kávé mennyisége is legyen több. Ha jó kávéra jót főztél, akkor az edényben is jó marad a kávé, csak a mennyisége legyen több.

A kávéfőzőre rá lehessen nézni, ilyenkor írd ki, hogy mivel és mennyire van feltöltve, használt kávé van-e benne, valamit éppen milyen és mennyi kávé van az edényben.

A vezérlőprogramban metódus hívásokkal üzemeltesd a kávéfőzőt.

```
kv.feltolt("hasznalt_kave",5);
kv.feltolt("kave",5);
kv.feltolt("kave",3);
kv.feltolt("viz",10);
kv.feltolt("viz",2);
kv.foz();
kv.foz();
kv.nez();
edeny.kostol();
edeny.kiont();
kv.nez();
```

Megfelelő hibakezeléssel biztosítsd, hogy ne fordulhassanak elő hibák. A hibákról szöveges üzenetet is írj ki:

- negatív vízmennyiség
- túltöltés (csak akkor tölthess bele, ha belefér)
- üresen főzni
- edényből kétszer kiöntés, stb.