
Szolgáltatásorientált Programozás gy.

II. Zárthelyi Dolgozat

# HERBA(N)K ATM Szimuláció

Ebben a zárthelyi dolgozatban egy központi banki szerver és a hozzá kapcsolódó ATM kliensek működését kell leprogramoznia. Minden ügyfélnek van egy bankszámlája, amelyhez bankkártyák tartoznak. Létezik **VIP**, illetve **Normál** számlacsomag. A rendszernek képesnek kell lennie tranzakciók kezelésére, és ha a kapcsolat megszakad, az adatokat a központi szerveren kell tárolni. A tranzakciók kezelése során ügyelni kell a számlaegyenleg helyességére!
Az kék színnel jelzett feladatok nem kötelezőek, plusz pontnak számítanak!

# Objektum mezők

- **Bankkártya:** kártyaszám (szöveg), PIN kód (egész), tulajdonos számlaszáma (szöveg)
- **Bankszámla:** számlaszám (szöveg), tulajdonos neve (szöveg), egyenleg (egész), típus (VIP/Normál)
- **BankKliens:** Tartalmazza a szükséges banki utasításokat, és kezeli a beérkező kéréseket.

# Feladatok (40 pont)

**1. feladat.** A szerverrel a kliensek a 127.0.0.1 IP címen és 1210 porton keresztül kommunikálnak. Az IP címet és portot tárolja el az alkalmazás konfigurációs fájljában. Ebből nyerje ki és alkalmazza a szerver és kliens applikációban! **(4 pont)**

**2. feladat.** Bankkártya osztály **(8 pont)**

A. Készítsen egy ”bankkártya generáló” konstruktort!

B. A bankkártyák generálásánál a következőt vegye figyelembe:

*a)* A kártyaszám **16 számjegyből** álló **véletlenszerű** számsor, és **mindig 1209-el** kezdődik. (Minta: 1209 1234 5678 9101)

*b)* A PIN kód **4 számból** álló **véletlenszerű** számsor, és **nem egyezhet már meglévő kártyák PIN kódjával**. (Minta: 6543)

C. Készítsen egy statikus metódust, amely visszaad egy bankszámlaszám és PIN kód alapján egy bankkártya objektumot!

D. Készítsen egy statikus metódust, amely egy XML fájlból kiolvassa az összes bankkártya objektumot!

E. Készítsen egy statikus metódust, amely egy XML fájlba kiírja az összes bankkártya objektumot!

A **debitcards.xml** fájl felépítése a következő:

```
<debitcards>
 <card number="1209458291023847" pin="1942" owner="HUF-00008812"/>
 <card ...>
 ...
</debitcards>
```

**3. feladat.** Bankszámla osztály **(10 pont)**

A. Készítsen egy "bankszámla generáló" kontstruktort!

B. **Normál** számla esetén 10 000 Ft, **VIP** esetén 25 000 Ft regisztrációs bónusz van.

C. A számla egyenlege nem eshet 0 forint alá!

D. Amikor a számla létrejön, generáljon egy bankkártyát, a számla tulajdonos számlaszáma pedig a következő legyen:

*a)* Mindig **HUF** szöveggel kezdődik.

*b)* Kötőjellel elválasztva 8 számjegyből álló számsor: az **első 4 számjegy** a legnagyobb létrehozott számlaszámot követő szám, a **második 4 számjegy** véletlenszerű számjegy.
(Példa: HUF-**0159**7863 követő következő számlaszám: HUF-**0160**9832)

E. Készítsen egy statikus, bool-al visszatérő függvényt, amely regisztrál egy új számlát használatra! Számla készítéskor szükséges megadni a tulajdonos nevét, illetve melyik csomagot választja.

F. Készítsen egy statikus, bankszámla objektummal visszatérő függvényt, amely bankkártyaszám és PIN kód alapján be tud engedni egy számlatulajdonost a rendszerbe. Amennyiben a kombináció nem egyezik vagy nem található ilyen bankkártya, adjon vissza **null** értéket!

G. Készítsen egy statikus metódust, amely egy XML fájlból kiolvassa az összes bankszámla objektumot! Az XML felépítését megtalálja lentebb!

H. Készítsen egy statikus metódust, amely egy XML fájlba kiírja az összes bankszámla objektumot! Az XML felépítését megtalálja lentebb!

A **accounts.xml** fájl felépítése a következő:

```
<accounts>
 <account number="HUF-00008812" owner="Herbák Marcell" balance="6767" type="VIP"/>
 <account ...>
 ...
</accounts>
```

**4. feladat.** BankKliens osztály **(12 pont)**

A. Használjon dedikált szálak helyett **ThreadPool-t**!

B. Készítsen egy metódust, amely elküldi a kliensnek a paraméterben megadott szöveget!

C. Készítsen egy metódust, amely bezárja ezeket a streameket és a kliens kapcsolatot, és eltávolítja a klienst a listából!

D. A parancsok paraméterei szóköz karakterrel vannak elválasztva.

E. A (+) karakterrel jelölt parancsok csak hozzáférés után elérhetőek. Készítse el a következő parancsokat:

- **QUIT**: Bezárja a végrehajtó kliens kapcsolatát.
- **ACCESS**: Kapcsolódás az szerverhez a bankkártyaszám és PIN alapján. Amennyiben sikeres a belépés, tárolja el a bankszámla objektumot.
- **OPEN**: Létrehoz egy új bankszámlát! Létrehozás után belép a szerverre! Létrehozás után írja ki a bankkártya számát és a PIN kódot!
- **LOGOUT**: Kijelentkezés a számláról. (+)
- **DETAILS**: Kiírja a paraméterben megadott számlának a tulajdonosát, számlaszámát, egyenlegét, számlatípusát és bankkártyaszámát. Ha paraméterben nincs megadva akkor a bejelentkezett felhasználóét adja vissza! (+)
- **TRANSACTION**: Paraméternek meg kell adni, hogy feltölteni (**DEPOSIT**) vagy levenni szeretne pénzt (**WITHDRAW**), illetve a mennyiséget! Ha a számla 0 Ft alá esne, jelezzen a kliensnek és szakítsa meg a tranzakciót! (+)
- **TRANSFER**: Paraméternek meg kell adni egy számlaszámot és az utalni kívánt összeget! **Normál** csomag esetén az utalás díja az utalt összeg 5%-a, **VIP** esetén ingyenes. Ha a számlatulajdonos éppen használja a programot, aki az utalást megkapja, jelezzen számára, hogy utalása érkezett! (+)
- Minden esetben jelezzen mindenről a szerver a kliens számára! Ismeretlen parancs vagy helytelen parancs paraméterek esetén írja ki a kliensnek a hiba okát!

**5. feladat.** A kliens (*TcpClient*) tudjon csatlakozni a szerverhez *TcpListener*-en keresztül, és tudjon kéréseket küldeni annak! Ha megszakad a kapcsolat egy klienssel, a szerver konzolján jelezzen! **(6 pont)**

Amennyiben a szerver forráskódja nem tud hiba nélkül lefordulni és elindulni, a dolgozat eredménye elégtelen! Futás közben fellépő hibák és kivételek az érintett feladatok pontját 0-ra csökkentik!