Nánási Bence - IHEHBV

Programozás alapjai 3. félév végi nagy házi feladat

Bankrendszer dokumentáció

I. Terv

Nagy házi feladatnak egy kezdetleges bankrendszer implementálását választottam. A projekt végső célja egy olyan program létrehozása, amely képes egy felhasználói felületen (GUI) az alapvető banki ügyintézést ellátni. Ez a szoftver különböző felhasználóknak tud banki fiókot nyitni, ezen a fiókon különböző banki folyamatokat lefolytatni, illetve lehetővé teszi egy adminisztrátornak (az adott banki rendszer üzemeltetője), hogy ezeket a fiókokat karbantartsa, bármelyiket elérje és le tudja kérdezni.

Maga a szoftver külső szemmel látható felületét úgy képzeltem el, hogy egy alap felugró ablakkal nyit, amely egy alapvető beléptetőrendszere lesz a szoftvernek. Itt két beviteli mezőbe: felhasználónév, jelszó lehet bevinni a fiók adatait és így elérni a program különböző további funkcióit. Itt lehet belépni adminisztrátorként is, ami az egyszerűség kedvéért egyelőre az admin – admin felhasználónév – jelszó párral fog működni. Ezután ez az ablak bezáródik és a program az adminisztrátori belépéstől függően két részre esik szét. Attól függően, hogy milyen jogosultságú bejelentkezés történt, egy következő ablak fog megjelenni, ahol a program lényegi része, azaz a fontosabb funkciók fognak megjelenni.

(A specializációban található szövegrészlet)

II. Használati esetek

Amikor a felhasználó megnyitja a programot a bejelntkező felület fogadja, amelyen a felhasználónevét és a jelszavát kérdezi a felhasználótól. Ha a megadott bemenet megfelel az adminok listában tárolt elemek valamelyikével, akkor az alkalmazás bezárja a bejelentkező felületet és megnyitja az adminisztrátori ablakot.

Az adminisztrátori ablakban ki lehet listázni az összes fiók adatait egy táblázatban név, számlaszám és egyenleg formátumban. Fel lehet venni új felhasználót, ahol meg kell adni a szükséges adatokat. A szoftver leellenőrzi a felhasználónevet, hogy volt – e már használva, ha igen, akkor hibát jelez. Új felhasználó felvétele esetén a szoftver automatikusan generál egy bankszámlaszámot a felhasználónak és az IBAN szabvány alapján elmenti azt. (ld. lejjebb) Itt is a szám generálása alatt leellenőrzi, hogy az IBAN már használatban van – e és csakis egyedi bankszámlaszámokat generál. Lehet fiókot törölni is bankszámlaszám alapján, ilyenkor a szoftver leellenőrzi, hogy a megadott számlaszám létezik – e és, ha nem, akkor hibát jelez.

Ha a megadott bemenet megfelel a fiókok listában tárolt elemek valamelyikével, akkor az alkalmazás bezárja a bejelentkező felületet és megnyitja a felhasználói ablakot.

A felhasználói ablakban lehetősége van a fiók kezelőjének megtekinteni a saját adatait és ha esetleges változtatást kíván tenni, akkor szerkesztheti azokat. Ha szeretne pénzt feltölteni azt is megteheti egy külön menüpontban. Itt az alkalmazás leteszteli, hogy a megadott bemenet ténylegesen szám – e, ha nem, akkor hibát jelez. A pénz hozzáadása után frissíti az adatbázist és kilép az ablakból. Hasonlóan jár el az algoritmus akkor, ha a felhasználó pénzt szeretne kivenni a számlájáról. Ha a felhasználó annyi pénzt próbál levenni a számlájáról, amennyi nincs az egyenlegében, akkor a szoftver hibát jelez. Ekkor a kifizetés összegét egy függvény segítségével lebontja a Magyarországon jelenleg használatos bankjegyekbe és a legkedvezőbb módon kifizeti azokat. A banki átutalás során egy bankszámlaszám megadása segítségével

lehet a felhasználó által megszabott mennyiségű pénzt küldeni egy célszámlára. Az utalás összegét jóváírjuk a célszámlára és levonjuk a felhasználó számlájáról, majd ezekről egy megerősítő üzenetet mutatunk a felhasználónak. Ha a felhasználó nem megfelelő bankszámlaszámot ad meg, vagy az összeg helyére nem egy számot ír be, vagy akkora összeget próbál elküldeni, amennyi nincs a számláján, akkor a szoftver hibát jelez.

Mindkét esetben igaz, hogyha a felhasználó a kijelentkezés gombra nyom, akkor a szoftver bezárja az ablakot (admin ablak vagy user ablak) és automatikusan a bejelentkező ablakot jeleníti meg újra.

Hogyha ezek közül a lehetőségek közül egyik sem következik be – azaz a bejelentkezési adatok hibásak – a szoftver hibát jelez.

III. Adatok tárolása

Az adatok tárolását két fájl a bankdata.dat, illetve az admins.dat végzi. Ide a szoftver szerializált formában menti a dokumentumokat. Minden alkalommal, amikor az alkalmazás megnyílik a Bank példányában lévő listákat a bankdata, illetve az admin fájlok jelenlegi tartalmával feltölti (felülírja). Ha változás történik, vagy egy ablakot bezár a felhasználó, akkor a listákban tárolt összes elemet szerializáltan elmenti (frissíti) a két fájlban. A fájlokba mentést és a fájlokból olvasást a Bank osztályon belül található serializeAccounts, serializeAdmins, deserializeAccounts és a deserializeAdmins metódusok végzik.

IV. Osztályok dokumentálása

User Osztály Dokumentáció:

A User osztály egy felhasználót reprezentál, és alapvető információkat tárol egy felhasználóval kapcsolatban.

Adattagok:

- name (Típus: String): A felhasználó teljes nevét tárolja.
- username (Típus: String): A felhasználó bejelentkezési felhasználónevét tárolja.
- password (Típus: String): A felhasználó bejelentkezési jelszavát tárolja.

Metódusok:

- getName(): String: Visszaadja a felhasználó teljes nevét.
- getUsername(): String: Visszaadja a felhasználó bejelentkezési felhasználónevét.
- getPassword(): String: Visszaadja a felhasználó bejelentkezési jelszavát.
- setName(String name): Beállítja a felhasználó teljes nevét.
- setUsername(String username): Beállítja a felhasználó bejelentkezési felhasználónevét.
- setPassword(String password): Beállítja a felhasználó bejelentkezési jelszavát.

Account Osztály Dokumentáció:

Az Account osztály egy banki számlát reprezentál, amely a felhasználói információkat is tartalmaz.

Adattagok:

- IBAN (Típus: String): Az international bank account number-t tárolja a HU (országkód), 42 (ellenőrző szám), 123 (bank kód), és 1111111110000000 (16 számjegyű bankszámlaszám) formátumban.
- money (Típus: double): Az aktuális összeget a számlán tárolja.

Konstruktorok:

- Account(): Az osztály alapértelmezett konstruktora, amely inicializálja a felhasználói adatokat és a számla egyenlegét 0-ra.
- Account(String name, String user, String pw): Konstruktor, amely inicializálja a felhasználói adatokat a paraméterek alapján, és a számla egyenlegét 0-ra.

Metódusok:

- getName(): String: Visszaadja a számlához tartozó felhasználó teljes nevét.
- getUsername(): String: Visszaadja a számlához tartozó felhasználó bejelentkezési felhasználónevét.
- getPassword(): String: Visszaadja a számlához tartozó felhasználó bejelentkezési jelszavát.
- getMoney(): double: Visszaadja a számla egyenlegét.
- getIBAN(): String: Visszaadja a számlához tartozó IBAN-t.
- setMoney(double d): Beállítja a számla egyenlegét az átadott értékre.
- setIBAN(String d): Beállítja a számlához tartozó IBAN-t az átadott értékre.

Admin Osztály Dokumentáció:

Az Admin osztály egy rendszergazdát reprezentál, akinek van joga a különböző felhasználók adatait megtekinteni.

Adattagok:

- adminID (Típus: String): Az adminisztrátor azonosítóját tárolja.

Konstruktor:

- Admin(String name, String user, String pw): Konstruktor, amely inicializálja az adminisztrátor felhasználói adatait a paraméterek alapján.

Metódusok:

- getAdminId(): String: Visszaadja az adminisztrátor azonosítóját.
- setAdminID(String val): Beállítja az adminisztrátor azonosítóját az átadott értékre.
- setAdminID(): Generál és beállít egy új adminisztrátor azonosítót véletlenszerű számok alapján.

Bank Osztály Dokumentáció:

A Bank osztály egy bankot reprezentál, és kezeli az ügyfelek és adminisztrátorok adatait, valamint számos banki műveletet végez.

Adattagok:

- accounts (Típus: ArrayList<Account>): Az ügyfélszámlákat tároló lista.
- admins (Típus: ArrayList<Admin>): Az adminisztrátorokat tároló lista.

Metódusok:

- serializeAdmins(String filename): Adminisztrátorok szerializációja adott fájlnévvel.
- deserializeAdmins(String filename): Adminisztrátorok deszerializációja adott fájlnévvel.
- serializeAccounts(String filename): Ügyfélszámlák szerializációja adott fájlnévvel.
- deserializeAccounts(String filename): Ügyfélszámlák deszerializációja adott fájlnévvel.
- addAccount(Account account): Új számla hozzáadása a listához generált IBAN-nel.
- list(): Az összes számla listázása (tesztelési célokból).
- login(String username, String password): boolean: Bejelentkezés ellenőrzése felhasználónév és jelszó alapján.
- isInAccounts(String iban): boolean: Ellenőrzi, hogy az adott IBAN szerepel-e a számlák között.
- isAdmin(String username, String password): boolean: Ellenőrzi, hogy az adott felhasználó adminisztrátor-e.
- findAccount(String username, String password): Account: Keres egy számlát felhasználónév és jelszó alapján.
- transferMoney(Account from, Account to, double money): Pénzátutalás két számla között.
- findByIBAN(String IBAN): Account: Keres egy számlát IBAN alapján.
- changePassword(String older, String newer): Jelszóváltoztatás az adott régi jelszóval.
- isUsedIban(String iban): boolean: Ellenőrzi, hogy az adott IBAN már használatban van-e.
- generateIban(): String: Generál egy egyedi IBAN-t.
- printBankNotes(double amount): String: Visszaadja az összeget bankjegyekre lebontva.
- addAdmin(Admin admin): Adminisztrátor hozzáadása a listához generált azonosítóval.
- deleteAccount(String accountNumber): Törli a számlát az adott számlaszámon.
- isUsedUsername(String username): boolean: Ellenőrzi, hogy az adott felhasználónév már használatban van-e.
- findAdmin(String username, String password): Admin: Keres egy adminisztrátort felhasználónév és jelszó alapján.
- isUsedAdmin(String username): boolean: Ellenőrzi, hogy az adott adminisztrátori felhasználónév már használatban van-e.

LoginFrame Osztály Dokumentáció:

A LoginFrame osztály egy bejelentkezési ablakot reprezentál, amely lehetővé teszi a felhasználók és adminisztrátorok bejelentkezését.

Adattagok:

- usernameField (Típus: JTextField): A felhasználónév megadására szolgáló szövegmező.
- passwordField (Típus: JPasswordField): A jelszó megadására szolgáló szövegmező.
- loginButton (Típus: JButton): A bejelentkezés gomb.
- bank (Típus: Bank): A bankot reprezentáló osztály példánya.
- account (Típus: Account): Az aktuális felhasználó számláját tároló osztály példánya.

Metódusok:

- LoginFrame(Bank bank): Konstruktor, inicializálja a bejelentkezési ablak komponenseit és beállításait.
- actionPerformed(ActionEvent e): Az eseménykezelő metódus, amely lekezeli a bejelentkezés gombra kattintást.
- A LoginFrame osztály a JFrame osztályból származik, és implementálja az ActionListener interfészt az események kezeléséhez.

ApplicationFrame Osztály Dokumentáció:

Az ApplicationFrame osztály egy alkalmazásablakot reprezentál, amely a bank alkalmazás főképernyőjét tartalmazza. Ezen az ablakon különböző funkciók érhetők el.

Adattagok:

- bank (Típus: Bank): A bankot reprezentáló osztály példánya.
- account (Típus: Account): Az aktuális felhasználó számláját tároló osztály példánya.

Metódusok:

- ApplicationFrame(Bank b, Account account): Konstruktor, inicializálja az alkalmazásablakot és hozzáadja a különböző funkciókat.
- addFeatureButton(String label, ActionListener listener): Segédfüggvény a funkció gomb hozzáadásához.
- Az ApplicationFrame osztály a JFrame osztályból származik. Az ApplicationFrame a különböző funkcióit új osztályok meghívásával oldja meg.

Funkciók:

MyProfileFrame:

- Leírás: A felhasználó adatainak megtekintésére és módosítására szolgáló ablak.

AddMoneyFrame:

- Leírás: Pénz befizetésére szolgáló ablak, ahol a felhasználó új összeget adhat hozzá a számlájához.

MoneyCheckoutFrame:

- Leírás: Pénz kivételére szolgáló ablak, ahol a felhasználó kiválaszthatja a kívánt összeget a számlájáról.

TransferFrame:

- Leírás: Banki átutalásokhoz szolgáló ablak, ahol a felhasználó átutalhat pénzt egy másik számlára.

LoginFrame:

- Leírás: Bejelentkezési ablak, ahol a felhasználók és adminisztrátorok bejelentkezhetnek a rendszerbe. Újból meghívódik, hogyha a kijelentkezés gombra nyom a felhasználó.

AdminFeatureButtonPanel Osztály Dokumentáció:

Az AdminFeatureButtonPanel osztály egy adminisztrációs ablakot reprezentál, amelyen különböző adminisztrációs funkciók érhetők el.

Adattagok:

- bank (Típus: Bank): A bankot reprezentáló osztály példánya.

Metódusok:

- AdminFeatureButtonPanel(Bank b): Konstruktor, inicializálja az adminisztrációs ablakot és hozzáadja a különböző adminisztrációs funkciókat.
- addFeatureButton(String label, ActionListener listener): Segédfüggvény a funkció gomb hozzáadásához.

Az AdminFeatureButtonPanel osztály a JFrame osztályból származik.

Funkciók:

Fiókok listázása:

- Leírás: Az összes bankszámla listázása az adminisztrációs ablakon.

Új fiók hozzáadása:

- Leírás: Új bankszámla létrehozása az adminisztrációs ablakon.

Fiók törlése:

- Leírás: Egy meglévő bankszámla törlése az adminisztrációs ablakon.

Kijelentkezés:

- Leírás: Az adminisztrációs ablakból való kijelentkezés és visszalépés a bejelentkezési ablakra.