HÁZI FELADAT

Programozás alapjai 3.

Programozói dokumentáció

Toronyi Zsombor

S8F7DV

2024. november 25.

Tartalom

BankApp Projekt

Projekt Leírása

A BankApp egy Swing-alapú asztali alkalmazás, amely lehetővé teszi a felhasználók számára bankfiókok kezelését, tranzakciók végrehajtását, és felhasználói profilok adminisztrálását. Az alkalmazás SQLite adatbázist használ az adatok tárolására, és támogatja a felhasználók, számlák és tranzakciók kezelését.

Főbb Funkcionalitások

1. Felhasználói hitelesítés:
   - Regisztráció, bejelentkezés.

2. Számlakezelés:
   - Új számla nyitása, számlák befagyasztása és törlése.

3. Tranzakciók kezelése:
   - Pénz befizetése, kivétele, átutalása másik számlára.

4. Adatok megjelenítése:
   - Tranzakciók megtekintése, egyenleg kijelzése.

Projekt Felépítése

- UserManager.java
- AccountManager.java
- TransactionManager.java
- User.java
- Account.java
- Transaction.java
- MainWindow.java
- LoginWindow.java
- RegistrationWindow.java

Osztályok és Metódusok

1. UserManager

Ez az osztály kezeli a felhasználók adatbázisban történő nyilvántartását.

1. **saveUser(User user)**
   - Ment egy új felhasználót az adatbázisba
   - Paraméter: user - mentendő felhasználó
   - Visszatérési érték: a felhasználó ID-je

2. **userExists(String email)**
   - Ellenőrzi, hogy egy adott email cím már létezik-e az adatbázisban
   - Paraméter: email - keresendő email cím
   - Visszatérési érték: true, ha létezik, különben false

3. **authenticateUser(String email, String password)**
   - Ellenőrzi a felhasználó bejelentkezési adatait
   - Paraméterek: email, password
   - Visszatérési érték: true, ha az adatok helyesek, különben false

4. **loadUser(String email)**
   - Betölti egy adott email címhez tartozó felhasználó adatait
   - Paraméter: email
   - Visszatérési érték: User objektum

5. **deleteUser(String email)**
   - Törli az adott email címhez tartozó felhasználót
   - Paraméter: email
   - Visszatérési érték: true, ha sikeres, különben false

2. AccountManager

Ez az osztály kezeli a felhasználók bankszámláit.

1. **saveAccount(Account account)**
   - Ment egy új bankszámlát az adatbázisba
   - Paraméter: account - mentendő számla

2. **accountExists(int accountNumber)**
   - Ellenőrzi, hogy egy adott számlaszám létezik-e
   - Paraméter: accountNumber
   - Visszatérési érték: true, ha létezik, különben false

3. **loadAccount(int accountNumber)**
   - Betölti egy adott számlaszámhoz tartozó számlát
   - Paraméter: accountNumber
   - Visszatérési érték: Account objektum

4. **loadAccounts(int userID)**
   - Betölti egy adott felhasználó összes számláját
   - Paraméter: userID
   - Visszatérési érték: List<Account>

5. **deleteAccount(int accountNumber)**
   - Töröl egy adott számlaszámhoz tartozó számlát
   - Paraméter: accountNumber
   - Visszatérési érték: true, ha sikeres, különben false

6. **depositMoney(Account account, double amount)**
   - Pénzt helyez el egy számlán
   - Paraméterek: account, amount
   - Visszatérési érték: true, ha sikeres, különben false

7. **withdrawMoney(Account account, double amount)**
   - Pénzt vesz le egy számláról
   - Paraméterek: account, amount
   - Visszatérési érték: true, ha sikeres, különben false

8. **transferMoney(int senderAccount, int receiverAccount, double amount)**
   - Pénzt utal át egy számláról egy másikra
   - Paraméterek: senderAccount, receiverAccount, amount
   - Visszatérési érték: true, ha sikeres, különben false.

9. **freezeAccount(Account account)**
    - Lefagyaszt egy számlát
    - Paraméter: account

10. **unfreezeAccount(Account account)**
    - Feloldja egy számla fagyasztását
    - Paraméter: account

3. TransactionManager

Ez az osztály kezeli a tranzakciókat.

1. **saveTransaction(Transaction transaction)**
    - Elment egy tranzakciót az adatbázisba
    - Paraméter: transaction

2. **loadTransactions(Account account)**
    - Betölti egy adott számla tranzakcióit
    - Paraméter: account
    - Visszatérési érték: List<Transaction>

4. User

Ez az osztály egy felhasználót reprezentál.

5. Account

Ez az osztály egy bankszámlát reprezentál.

6. Transaction

Ez az osztály egy tranzakciót reprezentál.

7. MainWindow

Ez az osztály a főablakot valósítja meg.

8. LoginWindow

Ez az osztály a bejelentkezési ablakot valósítja meg.

9. RegistrationWindow

Ez az osztály a regisztrációs ablakot valósítja meg.

Adatbázis Szerkezete

1. Users tábla:
   - **user_id**: Egyedi azonosító
   - **email**: Felhasználó email-címe
   - **password**: Felhasználó jelszava
   - **datetime**: Regisztráció dátuma

2. Accounts tábla:
   - **account_id**: Egyedi azonosító
   - **user_id**: Hivatkozás a tulajdonosra
   - **account_number**: Egyedi számlaszám
   - **balance**: Számla egyenlege
   - **is_frozen**: Számla státusza (fagyasztott vagy sem)

3. Transactions tábla:
   - **transaction_id**: Egyedi azonosító
   - **sender_account_number**: Küldő számlaszáma
   - **receiver_account_number**: Fogadó számlaszáma
   - **amount**: Tranzakció összege
   - **comment**: Tranzakció megjegyzése
   - **date**: Tranzakció dátuma

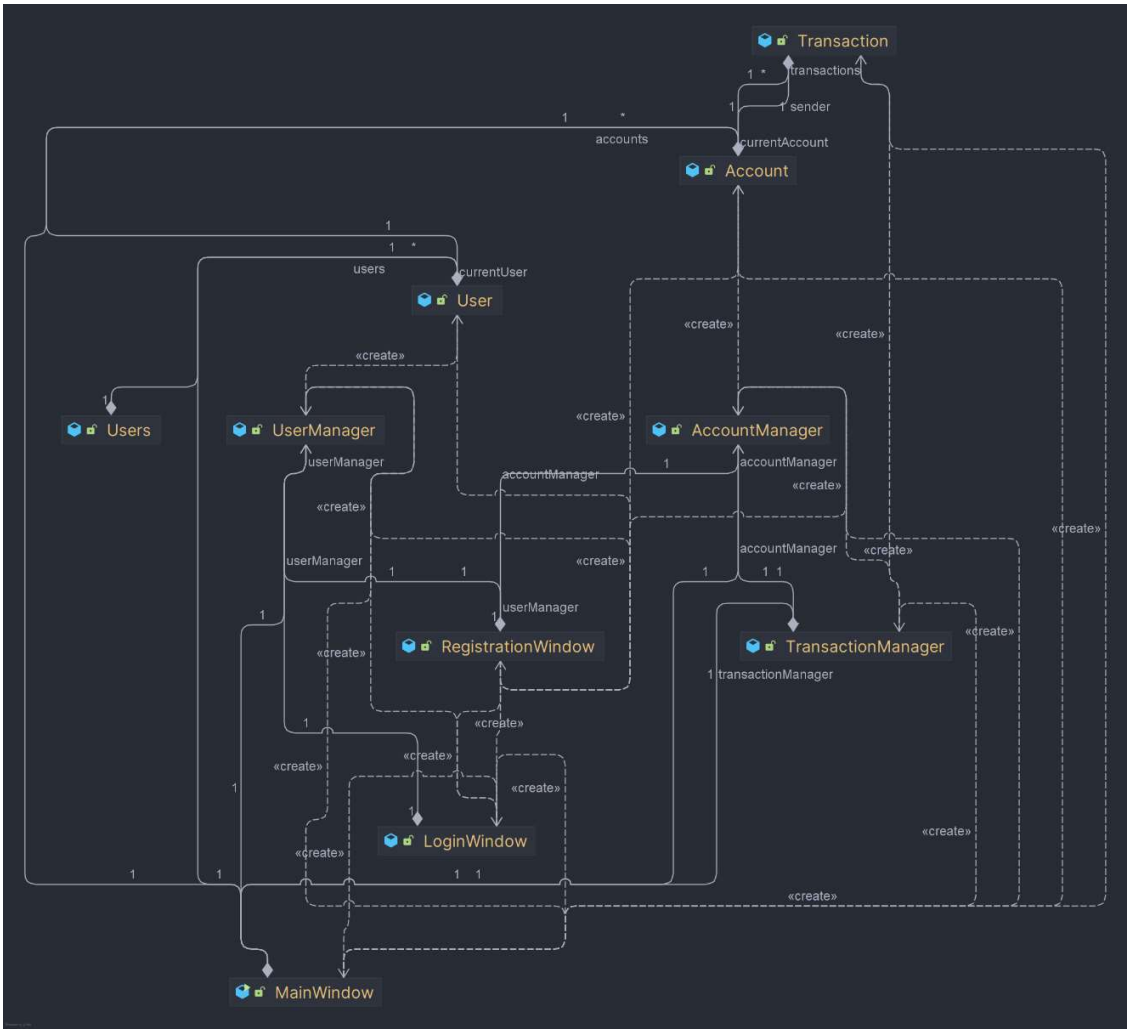
Az adatbáziskezelés megvalósításához az org.xerial:sqlite-jdbc:3.47.0.0 modult használtam.

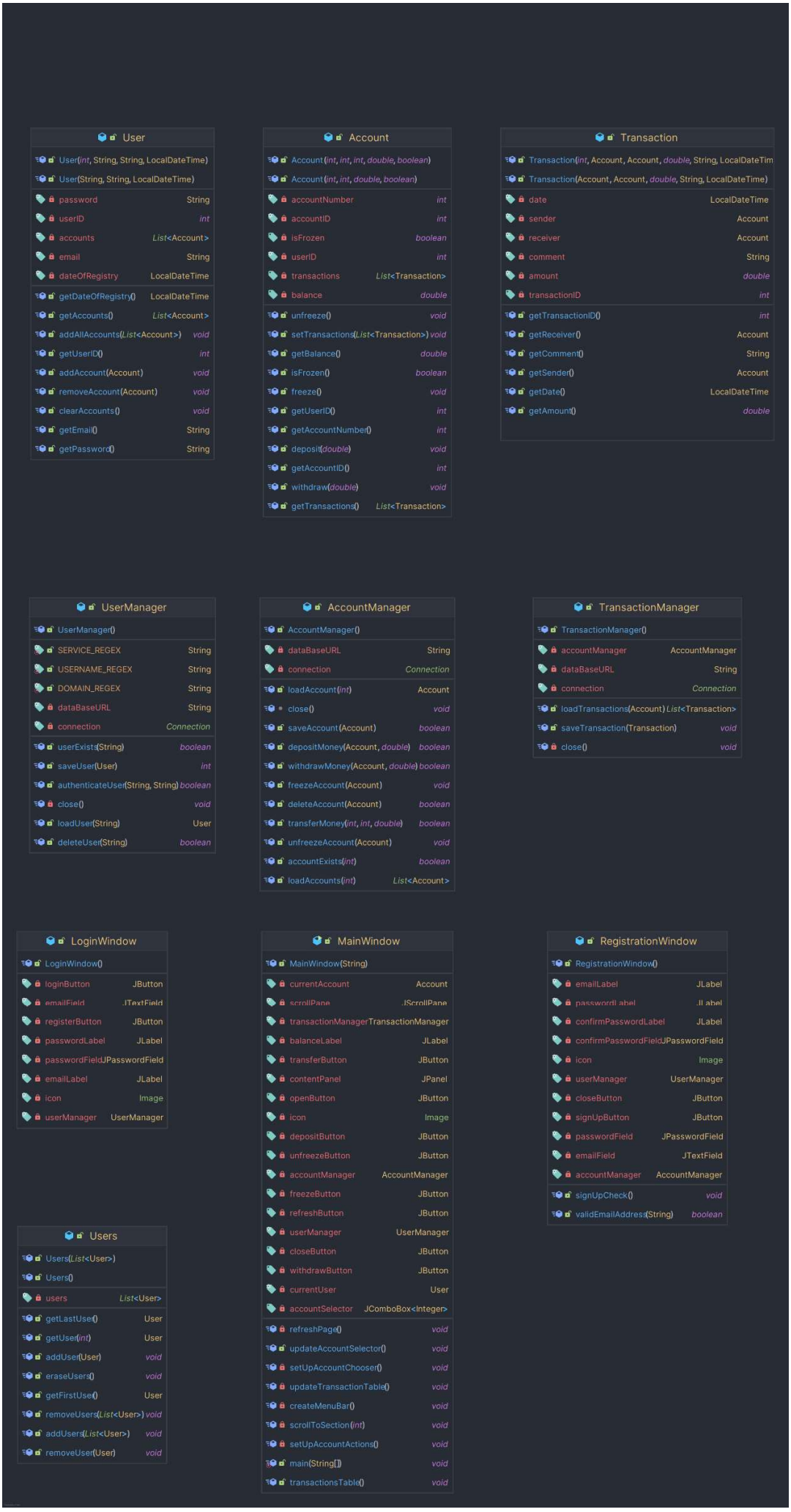
Példák és Működés

1. **Új számla nyitása:**
   - A felhasználó megadja a szükséges adatokat, a rendszer automatikusan generál egy egyedi számlaszámot.
   - Az új számla megjelenik a számlaválasztó mezőben.

2. **Pénz befizetése:**
   - A felhasználó beírja az összeget, majd kattint a "Deposit" gombra.
   - Az egyenleg frissül, a tranzakció mentésre kerül az adatbázisban.

3. **Pénz átvitele:**
   - A felhasználó megadja a cél számlaszámot, az összeget és egy megjegyzést.
   - A tranzakció végrehajtódik, és mindkét számla frissül.




User
User(int, String, String, LocalDateTime)
User(String, String, LocalDateTime)
password String
userID int
accounts List<Account>
email String
dateOfRegistry LocalDateTime
getDateOfRegistry() LocalDateTime
getAccounts() List<Account>
addAllAccounts(List<Account>) void
getUserID() int
addAccount(Account) void
removeAccount(Account) void
clearAccounts() void
getEmail() String
getPassword() String
Account
Account(int, int, int, double, boolean)
Account(int, int, double, boolean)
accountNumber int
accountID int
isFrozen boolean
userID int
transactions List<Transaction>
balance double
unfreeze() void
setTransactions(List<Transaction>) void
getBalance() double
isFrozen() boolean
freeze() void
getUserID() int
getAccountNumber() int
deposit(double) void
getAccountID() int
withdraw(double) void
getTransactions() List<Transaction>
Transaction
Transaction(int, Account, Account, double, String, LocalDateTim
Transaction(Account, Account, double, String, LocalDateTime)
date LocalDateTime
sender Account
receiver Account
comment String
amount double
transactionID int
getTransactionID() int
getReceiver() Account
getComment() String
getSender() Account
getDate() LocalDateTime
getAmount() double
UserManager
UserManager()
SERVICE_REGEX String
USERNAME_REGEX String
DOMAIN_REGEX String
dataBaseURL String
connection Connection
userExists(String) boolean
saveUser(User) int
authenticateUser(String, String) boolean
close() void
loadUser(String) User
deleteUser(String) boolean
AccountManager
AccountManager()
dataBaseURL String
connection Connection
loadAccount(int) Account
close() void
saveAccount(Account) boolean
depositMoney(Account, double) boolean
withdrawMoney(Account, double) boolean
freezeAccount(Account) void
deleteAccount(Account) boolean
transferMoney(int, int, double) boolean
unfreezeAccount(Account) void
accountExists(int) boolean
loadAccounts(int) List<Account>
TransactionManager
TransactionManager()
accountManager AccountManager
dataBaseURL String
connection Connection
loadTransactions(Account) List<Transaction>
saveTransaction(Transaction) void
close() void
LoginWindow
LoginWindow()
loginButton JButton
emailField JTextField
registerButton JButton
passwordLabel JLabel
passwordField JPasswordField
emailLabel JLabel
icon Image
userManager UserManager
MainWindow
MainWindow(String)
currentAccount Account
scrollPane JScrollPane
transactionManager TransactionManager
balanceLabel JLabel
transferButton JButton
contentPanel JPanel
openButton JButton
icon Image
depositButton JButton
unfreezeButton JButton
accountManager AccountManager
freezeButton JButton
refreshButton JButton
userManager UserManager
closeButton JButton
withdrawButton JButton
currentUser User
accountSelector JComboBox<Integer>
refreshPage() void
updateAccountSelector() void
setUpAccountChooser() void
updateTransactionTable() void
createMenuBar() void
scrollToSection(int) void
setUpAccountActions() void
main(String[]) void
transactionsTable() void
RegistrationWindow
RegistrationWindow()
emailLabel JLabel
passwordLabel JLabel
confirmPasswordLabel JLabel
confirmPasswordField JPasswordField
icon Image
userManager UserManager
closeButton JButton
signUpButton JButton
passwordField JPasswordField
emailField JTextField
accountManager AccountManager
signUpCheck() void
validEmailAddress(String) boolean
Users
Users(List<User>)
Users()
users List<User>
getLastUser() User
getUser(int) User
addUser(User) void
eraseUsers() void
getFirstUser() User
removeUsers(List<User>) void
addUsers(List<User>) void
removeUser(User) void