

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



przedmiot
Algorytmy i programowanie 2



Rozliczenie wspólnych wydatków - etap I

Mateusz Orzełowski, Krzysztof Czaplicki, Rafał Kowalczyk, Maja Berej
Numer albumu 324937, 324907, 324919, 324903

WARSZAWA 14 kwietnia 2023

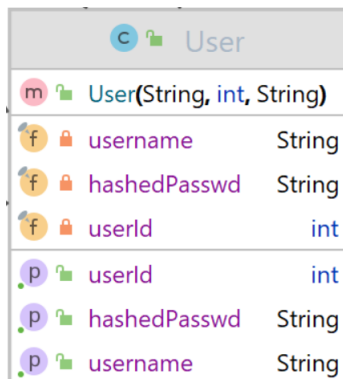
Spis treści

1. Wstęp	3
2. Klasy	3
2.1. User	4
2.2. Transaction	4
2.3. TransactionGraph	5
2.4. TransactionReader	5
2.5. Protocol	6
2.6. Client	7
2.7. Server	7
2.8. Database	8
2.9. App	9
3. Opis aplikacji	10

2. Klasy

2.1. User

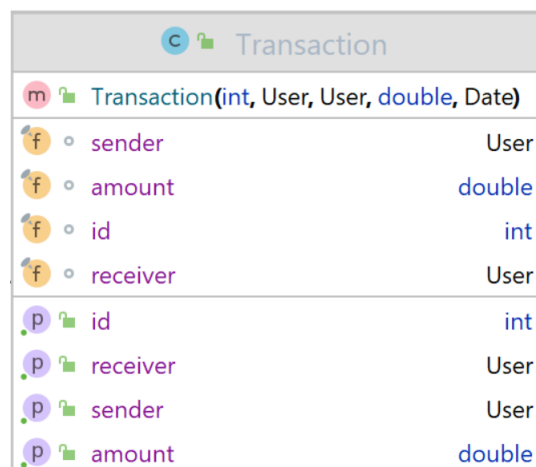
Klasa **User** przechowuje dane użytkownika. Posiada ona funkcje zwracające informacje o użytkowniku.



- **getUsername** - getter username
- **getHashedPass** - getter hashedPasswd
- **getUserId** - getter userId

2.2. Transaction

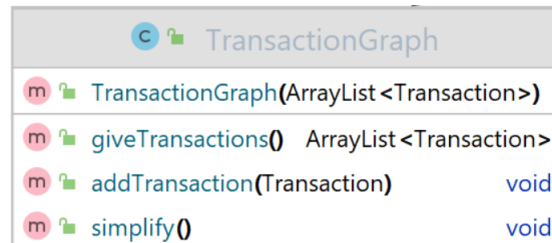
Klasa **Transcation** odpowiada za jedną transakcję. Zawiera informację od kogo i do kogo ma przyjść dana kwota.



- **getSender** - getter sender
- **getReceiver** - getter receiver
- **getAmount** - getter amount
- **getId** - getter id

2.3. TransactionGraph

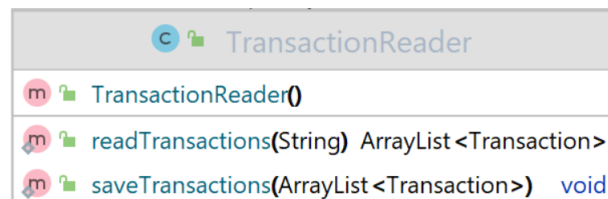
Klasa **TranscationGraph** to struktura danych, która służy do uproszczenia przepływu pieniędzy. Zawiera metodę **addTransaction**, która pozwala na dodanie nowej krawędzi do grafu oraz metodę **simplify**, która analizuje i upraszcza graf.



- **giveTransactions** - zwraca listę transakcji w grafie
- **addTransaction** - przyjmuje obiekt *Transaction*, dodaje transakcję do grafu
- **simplify** - upraszcza graf transakcji

2.4. TransactionReader

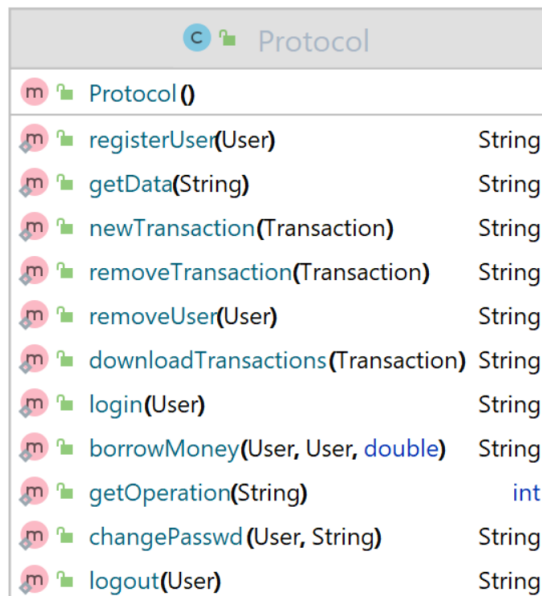
Klasa **TransactionReader** służy do zapisywania oraz odczytywania całej listy transakcji z pliku.



- **readTransactions** - odczytuje i zwraca listę transakcji z pliku
- **saveTransactions** - zapisuje listę transakcji do pliku

2.5. Protocol

Protocol jest to klasa, która pozwala na ustandaryzowanie wymiany informacji między klientem a serwerem. Zawiera zarówno metody kodujące dane informacje jak i metodę interpretującą zakodowane polecenie.

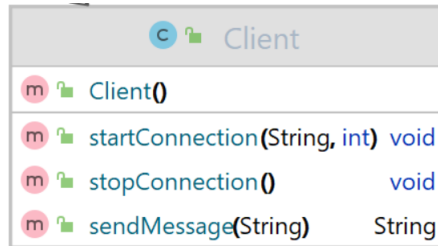


Protocol	
Protocol()	
registerUser(User)	String
getData(String)	String
newTransaction(Transaction)	String
removeTransaction(Transaction)	String
removeUser(User)	String
downloadTransactions(Transaction)	String
login(User)	String
borrowMoney(User, User, double)	String
getOperation(String)	int
changePasswd (User, String)	String
logout(User)	String

- **registerUser** - polecenie do rejestracji użytkownika
- **getData** - metoda interpretująca otrzymanego Stringa, zakodowanego według określonego schematu, zwraca uzyskane dane
- **newTransaction** - polecenie dodania nowej transakcji
- **removeTransaction** - polecenie usunięcia transakcji
- **removeUser** - polecenie usunięcia użytkownika
- **downloadTransactions** - polecenie pobrania transakcji
- **login** - polecenie logowania użytkownika
- **borrowMoney** - polecenie sprawdzenia czy możliwa jest operacja pożyczania pieniędzy
- **getOperation** - metoda interpretująca otrzymanego Stringa, zakodowanego według określonego schematu, zwraca rodzaj rozpoznanej operacji
- **changePasswd** - polecenie zmieniające hasło
- **logout** - polecenie wylogowania

2.6. Client

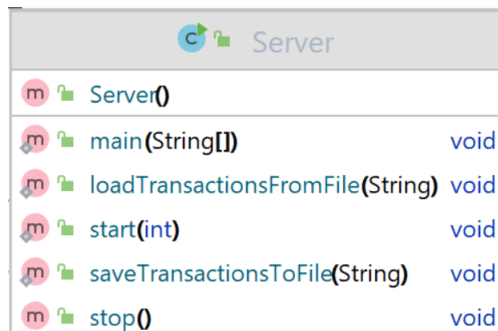
Klasa **Client** odpowiada za obsługę połączenia z serwerem przy użyciu socket'ów.



- **startConnection** - metoda rozpoczynająca połączenie z serwerem, przyjmuje adres ip jako String oraz int jako port
- **stopConnection** - metoda kończąca połączenie z serwerem
- **sendMessage** - metoda wysyłająca wiadomość do serwera, przyjmuje wiadomość typu String, wiadomość jest poleceniem zakodowanym przy pomocy klasy Protocol

2.7. Server

Klasa **Server** stanowi aplikację serwera. Spaja wszystkie klasy, które są potrzebne do komunikacji i obsługi klienta.



- **main** - główna metoda programu
- **loadTransactionFromFile** - metoda odczytująca transakcje z pliku
- **start** - metoda startująca serwer
- **saveTransactionsToFile** - metoda zapisująca transakcje do pliku
- **stop** - metoda stopująca serwer

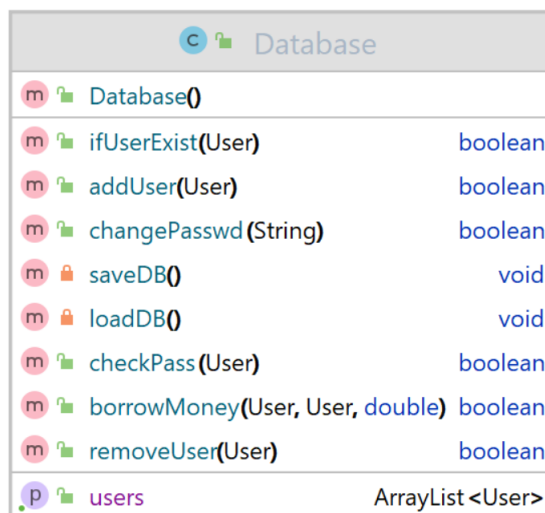
Klasa **EchoClientHandler** to klasa wewnętrzna klasy **Server**. Jest tworzona w momencie utworzenia nowego wątku, gdy nowy klient połączy się z serwerem. Nieustannie nasłuchuje połączenie z klientem.



- **run** - główna metoda obsługująca klienta

2.8. Database

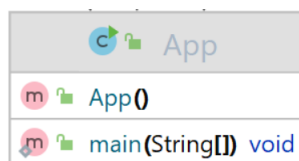
Klasa **Database** obsługuje bazę danych, w której znajdują się dane dotyczące użytkowników. Pozwala na przeprowadzenie podstawowych operacji na zapisanych rekordach.



- **ifUserExist** - metoda zwracająca `boolean`, który mówi nam czy dany użytkownik istnieje
- **addUser** - metoda dodająca użytkownika do bazy danych
- **changePasswd** - metoda zmieniająca hasło danego użytkownika
- **saveDB** - prywatna metoda zapisująca dane do pliku
- **loadDB** - prywatna metoda ładująca dane z pliku
- **checkPass** - metoda sprawdzająca czy dane do logowania są poprawne
- **borrowMoney** - metoda zwracająca `boolean`, czy dana operacja jest możliwa, czy tacy użytkownicy istnieją
- **removeUser** - metoda usuwająca użytkownika z bazy danych

2.9. App

Klasa **App** implementuje aplikację działającą na komputerze użytkownika oraz interfejs graficzny.



- **main** - główna metoda programu działającego po stronie klienta

3. Opis aplikacji

Aplikacja powinna umożliwiać użytkownikowi:

- logowanie się
- rejestrację
- zmianę hasła
- usunięcie konta
- dodanie transakcji
- usunięcie transakcji
- pobranie wszystkich transakcji.