

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 13 kwietnia 2023

Spis treści

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

2. Klasy

2.1. User

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

User		
m	User(String, int, String)	
f	username	String
f	hashedPasswd	String
f	userId	int
p	userId	int
p	hashedPasswd	String
p	username	String

2.2. Transaction

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

Transaction		
m	Transaction(int, User, User, double, Date)	
f	sender	User
f	amount	double
f	id	int
f	receiver	User
p	id	int
p	receiver	User
p	sender	User
p	amount	double

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.

TransactionGraph		
m	TransactionGraph(ArrayList<Transaction>)	
m	giveTransactions()	ArrayList<Transaction>
m	addTransaction(Transaction)	void
m	simplify()	void

2.4. TransactionReader

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

TransactionReader	
m	TransactionReader()
m	readTransactions(String) ArrayList<Transaction>
m	saveTransactions(ArrayList<Transaction>) void

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

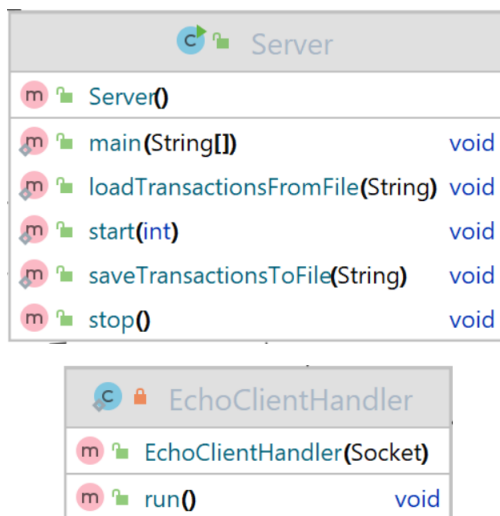
2.6. Client

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

Client	
m	Client()
m	startConnection(String, int) void
m	stopConnection() void
m	sendMessage(String) String

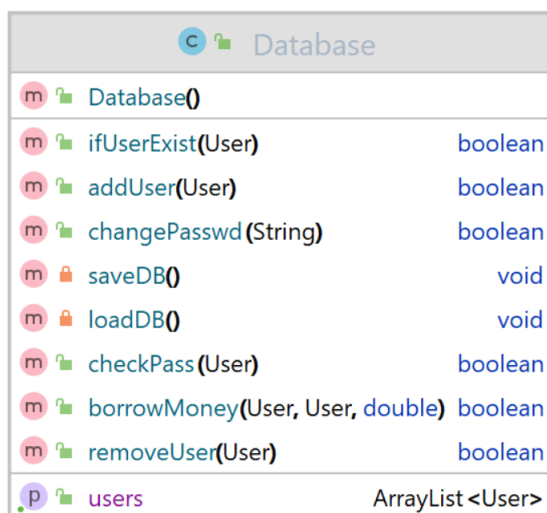
2.7. Server

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



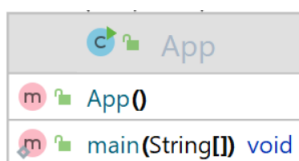
2.8. Database

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



2.9. App

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



3. Opis aplikacji

Aplikacja powinna umożliwiać użytkownikowi:

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