SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Zadanie 1 - MyBlockchain

Adrián Vančo
ID: 103171
Predmet: Digitálne meny a blockchain

Zadanie

Cieľom zadania je vytvoriť vlastnú blockchainovú sieť, ktorá bude zvládať základné dopyty, zaradí transakcie do blokov, jednotlivé bloky do reťaze blokov a dosiahne konsenzus medzi účastníkmi.
Zadanie je rozdelené do 3 fáz.

Fáza 1 – Jednoduchý coin

Za úlohu je implementovať logiku pre centrálnu autoritu na spracovávanie transakcií a vytvorenie ledgeru. V každom bloku centrálna autorita dostane zoznam transakcií, ktoré skontroluje a zverejní zoznam platných transakcií. Môže sa stať že jedna transakcia sa môže odkazovať na inú v rovnakom bloku. Tiež sa môže stať, že v zozname obdržaných transakcií nachádza viac ako jedna transakcia míňajúca rovnaký výstup (UTXO- unspent transaction output), toto by bol samozrejme double-spending a platná je iba jedna transakcia míňajúca UTXO. Ostatné transakcie, ktoré tiež míňali toto UTXO sú neplatné. Znamená to, že transakcie nemôžu byť kontrolované izolovane, je to problém ako si vybrať podmnožinu transakcií, ktoré sú spoločne platné.
V realite ak nastane double-spending sa preferuje vybrať transakciu, ktorá ma najväčšiu odmenu (fee), avšak z nej potom nemusíme nadobudnúť väčšiu celkovú odmenu.

Úlohou bolo do súboru HandleTxs.java implementovať triedu HandleTxs s týmito náležitosťami:

Konštruktor : public HandleTxs(UTXOPool utxoPool);
Funkcie : public boolean txIsValid(Transaction tx);
public Transaction[] handler(Transaction[] possibleTxs);

public HandleTxs(UTXOPool utxoPool);

vytvorí kópiu obdržaného utxoPoolu a uchová si referenciu.

public boolean txIsValid(Transaction tx);

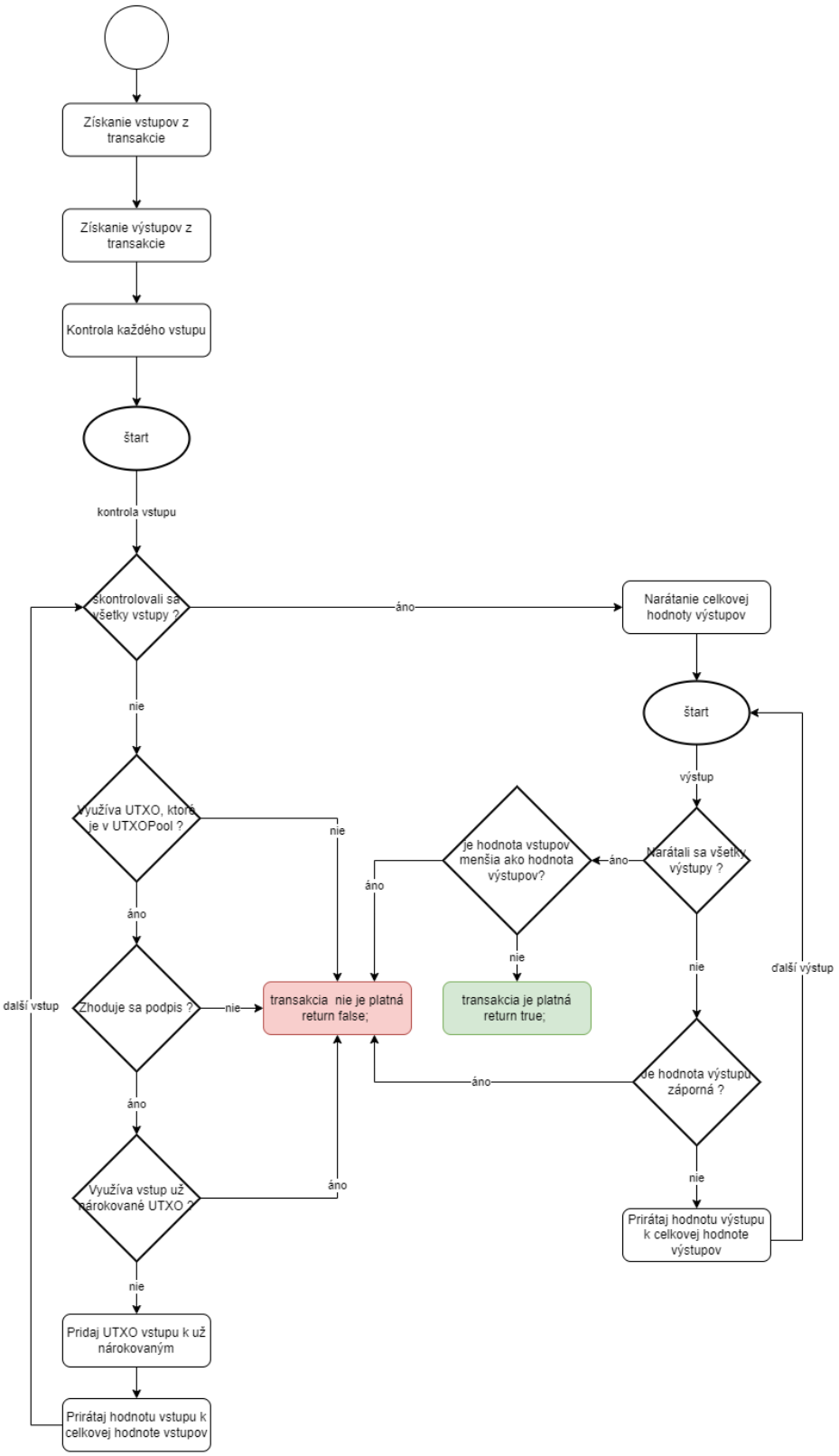
Overí platnosť transakcie. Vráti true ak je transakcia platná false ak nie je.
Transakcia je platná ak:

(1) sú všetky výstupy nárokované v aktuálnom UTXO pool
(2) podpisy na každom vstupe sú platné,
(3) žiadne UTXO nie je nárokované viackrát,
(4) všetky výstupné hodnoty s sú nezáporné a
(5) súčet vstupných hodnôt je väčší alebo rovný súčtu jej výstupných hodnôt

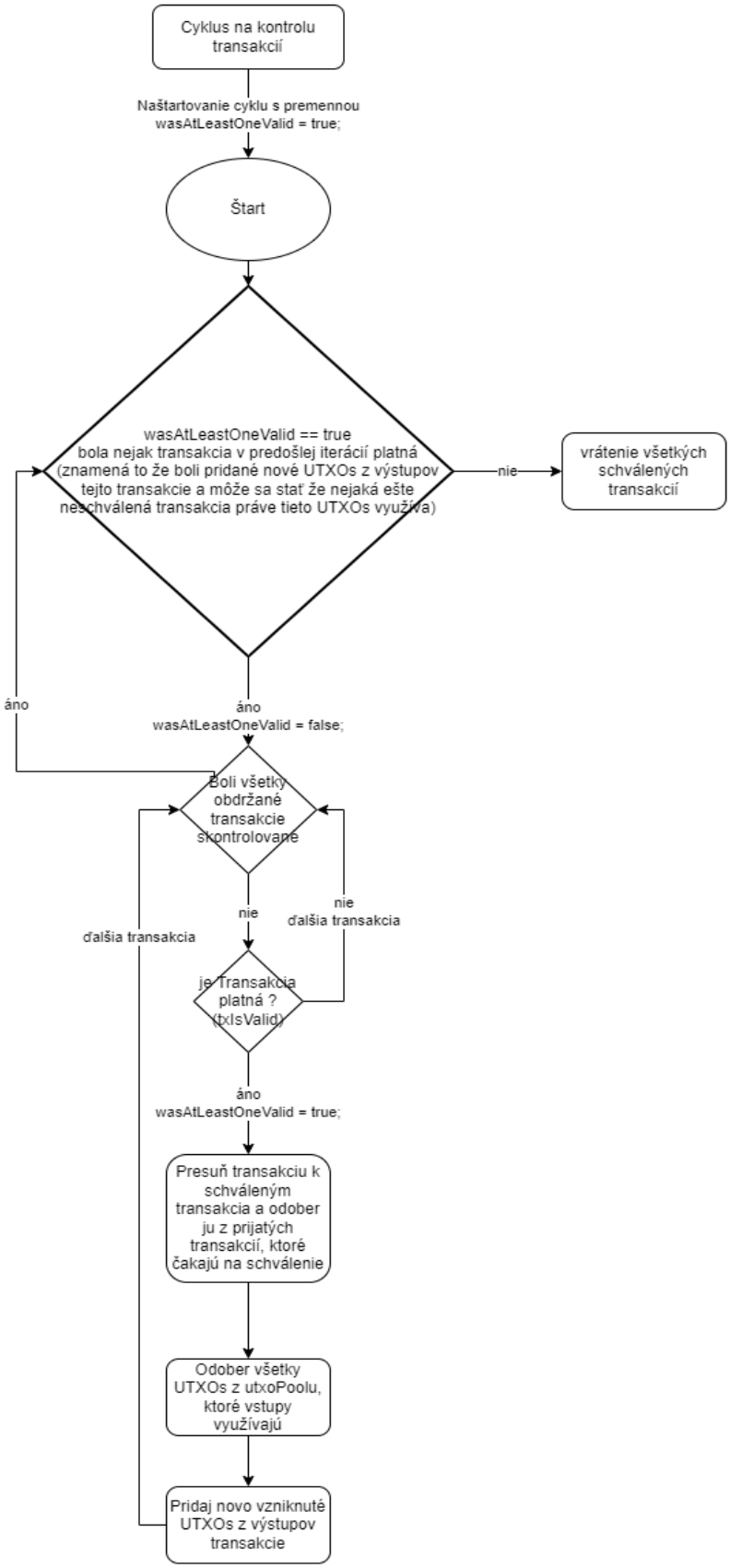
public Transaction[] handler(Transaction[] possibleTxs);

Overí všetky obdržané transakcie aktualizuje priradený utxoPool a vráti len platné transakcie.

Blokový návrh fungovania txIsValid(Transaction tx);



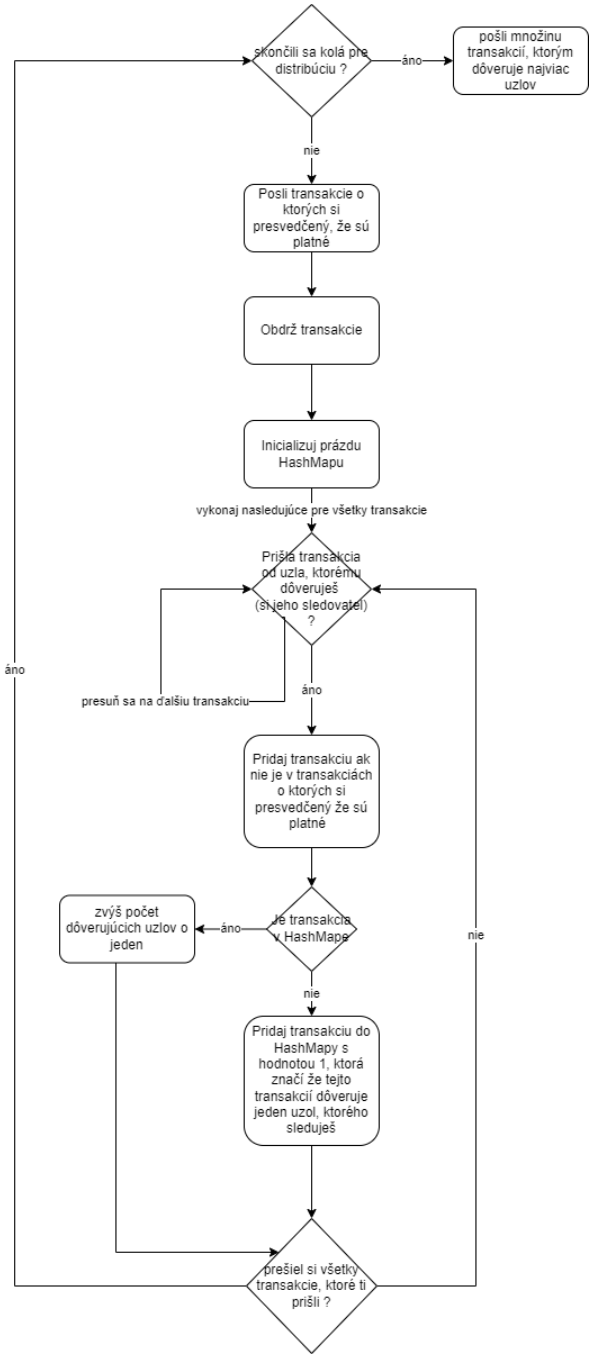
Blokový návrh fungovania handler(Transaction[] pissibleTxs);



Fáza 2 – Dôvera v konsenzus

V tejto fáze je potrebná implementácia fungovania v súbore TrustedNode.java tak aby všetky TrustedNode mali po skončení distribučných kôl rovnaké zoznamy transakcií teda dosiahli konsenzus. V tejto časti sa využíva algoritmus distribuovaného konsenzu s grafom vzťahov ,,dôvery“ medzi uzlami. To je alternatívna metóda odolávania sybil útokom a dosiahnutia konsenzu.

Popis riešenia:



Fáza 3 – Blockchain

V tejto fáze je potrebná implementácia funkcie **UTXOPoolGet()**, ktorá vráti aktuálny UTXOPool do súboru HandleTxs.java z fáze 1. A skopírovanie súboru do tejto fázy. Následne doplniť funkcionalitu do súboru BlockChain.java o nasledujúce:

Konštruktor: public BlockChain(Block genesisBlock);

Funckcie: public Block getBlockAtMaxHeight();
public UTXOPool getUTXOPoolAtMaxHeight();
public TransactionPool getTransactionPool();
public boolean blockAdd(Block block);
public void transactionAdd(Transaction tx);

public BlockChain(Block genesisBlock);

Konštruktor, ktorý pridá genesisBlock do blockchainu tým že vytvorí BlockNode pre genesisBlock a do konštruktora pre BlockNode dá genesisBlock, null pretože nemá žiadneho rodiča a nový utxoPool, ktorý má v sebe UTXO coinbase transakcie genesisBlocku. Následne ešte nastaví referenciu premennej na vytvorený BlockNode, ktorá slúži na získanie najvyššieho bloku, pridá ho do HashMapy, ktorú používam na vyhľadávanie blokov v blockchaine a ešte sa nastaví premenná poslednej výšky.

Použité premmenné:

private int lastHeight; udržiavanie koncovej výšky
private BlockNode oldestMaxHeightBlock; udržiavanie najvyšieho najstaršieho bloku
private TransactionPool txPool; transakčný pool blockchainu
private HashMap<ByteArrayWrapper,BlockNode> H; na rýchle vyhladanie bloku v blockchaine

public Block getBlockAtMaxHeight();
vráti blok na maximálnej výške

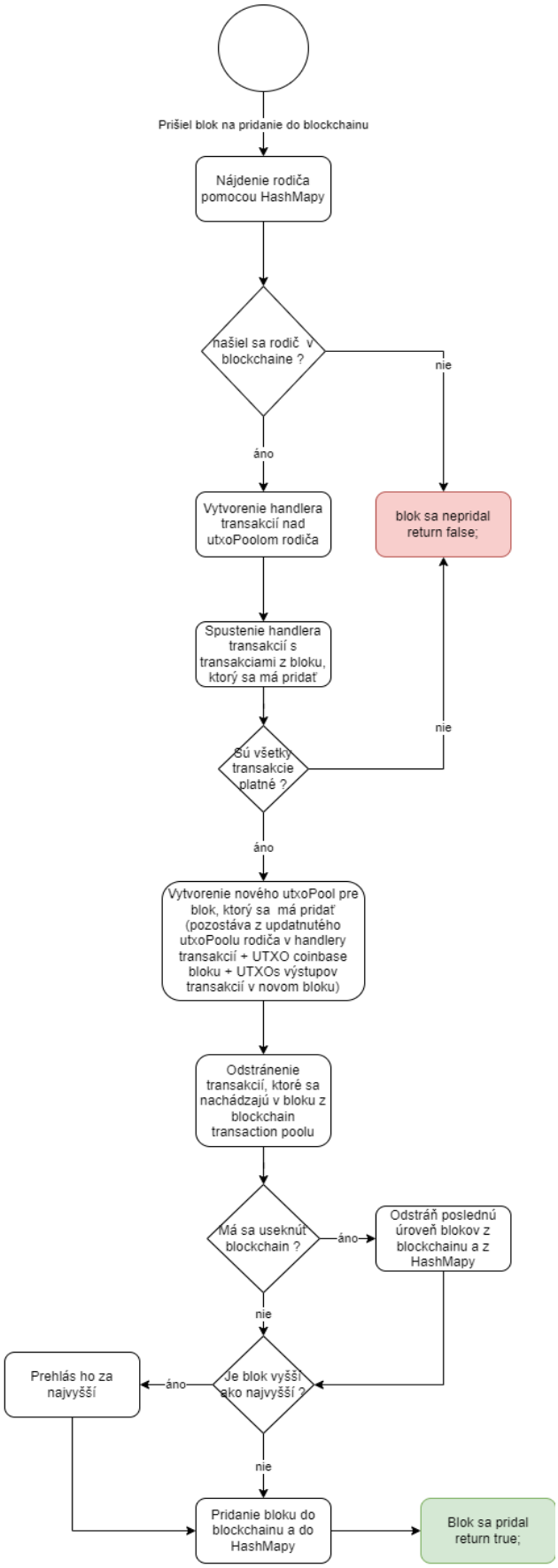
public UTXOPool getUTXOPoolAtMaxHeight();
vráti utxoPool na maximálnej výške

public TransactionPool getTransactionPool();
vráti kópiu transakčného poolu blockchainu

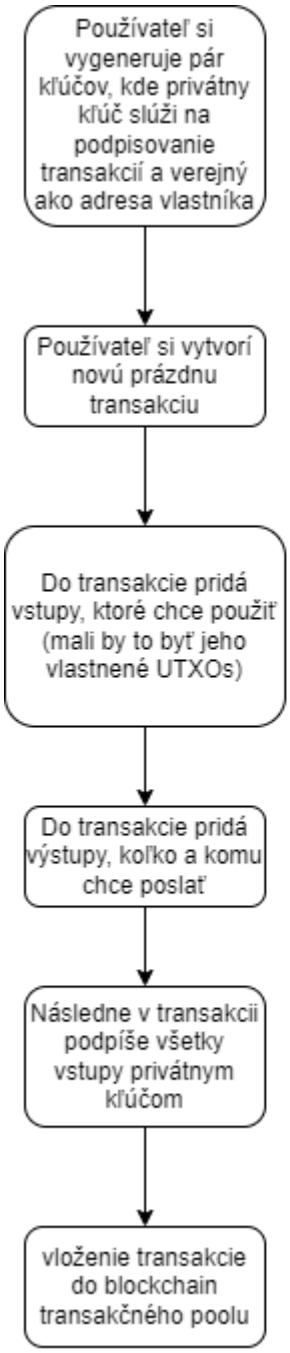
public boolean blockAdd(Block block);
pridá blok ak sa našiel rodič, transakcie v bloku sú platné. Ak sa pridal vráti true, inak false;

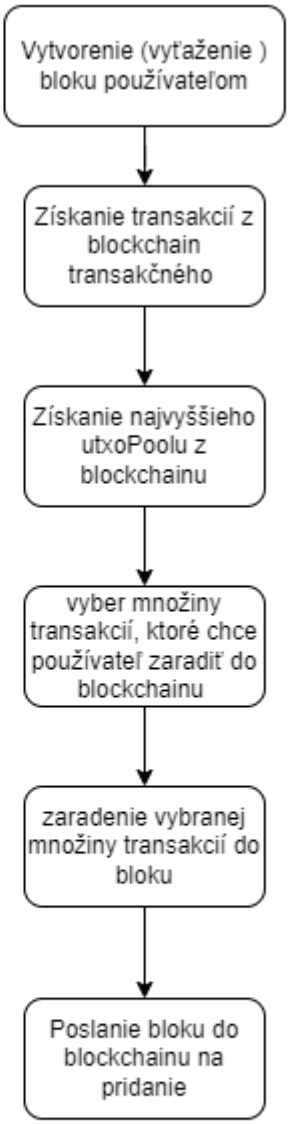
public void transactionAdd(Transaction tx);
pridá transakciu do blockchain transakčného poolu

Blokový návrh fungovania public boolean blockAdd(Block block);



Pridanie transakcie do blockchainpoolu	Pridanie bloku do blockchainu





Implementačné prostredie:

Vývojové prostredie (IDE): intellij idea ultimate
Jazyk: Java SDK: openjdk-17 verzia 17.0.2

Záver

Fáza 1: V prvej fáze som sa naučil, ako funguje overovanie transakcií, resp. že na to, aby bola transakcia považovaná za platnú je potrebné, aby spĺňala určité podmienky. Ďalej som sa zistil, čo je to "double spending" a kedy pri spracovaní transakcie nastáva. A taktiež to, že transakcie sa môžu skladať z viacerých vstupov a výstupov.

Fáza 2: V druhej fáze som pochopil ako približne si uzly medzi sebou vymieňajú transakcie a ako môžu dospieť ku konsenzusu pomocou grafového vzťahu.

Fáza 3: V tretej fáze som sa naučil ako bloky v blockchaine sa napájajú na už existujúce bloky. Ďalej to, že pri každom novo-vyťaženom bloku vzniká odmena za jeho vyťaženie (coinbase), ktorá je pripísaná tomu, kto daný blok vyťažil. Čo je to fork v blockchaine. Ako môže jedna vetva prerásť druhú a čo sa stane s transakciami a odmenami potom.