CSI2110 Assignment2 Report

Student Name : Zeyan liang

Student Number : 8392063

My miner id : zelian068

Description

public class Block

6 private variables

private int index; // the index of the block in the list
private java.sql.Timestamp timestamp; // time at which transaction has been processed
private Transaction transaction; // the transaction object
private String nonce; // random string (for proof of work)
private String previousHash; // previous hash (set to "" in first block)
private String hash; // hash of the block (hash of string obtained from previous variables via toString() method)

The main purpose of Public class block is creating a block with six different instances. In addition, it also includes GETTER and SETTER function. GETTER and SEETER will be used in Public class Blockchain.

```
public String toString() {
   return timestamp.toString() + ":" + transaction.toString() + "." + nonce + previousHash;
 }
```

This function is used to get hash code.

```
(Sha1.hash(blockchain.get(index).toString));
```

public class Transaction

3 private variables

private String sender;// The sender of transaction.
private String receiver;// The receiver of transaction.
private int amount;//The amont of transaction.

The main purpose of Public class transaction is creating a transaction with three different instances. In addition, it also includes GETTER and SETTER function. GETTER and SEETER will be used in Public class Blockchain.

public String toString() {

return sender + ":" + receiver + "=" + amount;

}

This function is used to get hash code.

(Sha1.hash(blockchain.get(index).toString));

public class Blockchain

// Creat a arraylist of types as Block to store each blcok
static final List<Block> blockchain = new ArrayList<Block>();
//indexOfname in name array
static int indexOfrname = 0;
// bank array to store every user's balance
static int[] bank = new int[1000];
// name array to store every user's name
static String[] name = new String[1000];

public Blockchain fromFile(String fileName)
Use buffer reader to store information and data from txt.file into array list.

public void toFile(String fileName)

Use buffer writter to store information and data from arraylist into a new txt file.

public boolean validateBlockchain()

Checking all the hashes, also check that all the index and previousHash attributes are consistent, and validate all the transactions to make sure that no one spent bitcoins they do not have.

public int getBalance(String username)

Ask a user to enter a new transaction by specifying a sender, a receiver and a bitcoin amount.

You must verify that the sender has enough money to proceed with the transaction.

public void add(Block block)

Add block which created by new transaction to blockchain

public class Sha1

used to get sha1 code by function (Sha1.hash(blockchain.get(index).toString));

Proof of work description

```
while(!Sha1.hash(BLOCK.toString()).substring(0,5).equals("00000")){
  result =(int)(Math.random() * 19+1);
  NONCE = "";
  for (int i = 0; i < result; i++){
      char c = (char) (int) (Math.random() * 93 + 33);
      NONCE += c;
  }
  BLOCK.setNonce(NONCE);
  //System.out.println(BLOCK.toString());
  //System.out.println(Sha1.hash(BLOCK.toString()));
  //System.out.println(BLOCK.getNonce()+"                    __________");
  timeOfrandomtrial ++;
}
//This is used to calculate how many times nonce changed to require the hash code req
//System.out.println(timeOfrandomtrial);
String HASH = Sha1.hash(BLOCK.toString());
//Set the required nonce and hash code.
BLOCK.setHash(HASH);
```

I use a while loop to find the hash code which is satisfy the hash code requirement (start with five zeros “00000”) If the hash code does not satisfy the requirement, the nonce will be random one more time by the code below. If the hash code satisfies the requirement the nonce and hash data will be set into block. After that, block will be added to blockchain. The variable timeOfrandomtrial will store the times of random trial.

result =(int)(Math.random() * 19+1);

NONCE = "";

for (int i = 0; i < result; i++){

char c = (char) (int) (Math.random() * 93 + 33);

NONCE += c;

}

Table

I create 20 new transactions.

745791	1000019	1889538	103249	135017
2701139	2557474	993103	141549	2473010
323917	17915	145286	2634821	2220196
3249491	743061	517434	2263423	565736

The average times is 1317522.