Santos Gagbegnon
300010769
Miner ID: sgagb074

October 2018
CSI 21110 [B]

Assignment 2: Programming a BlockChain Results Report

Class: Transaction

Transaction is a class that represents a very basic version of a BitCoin transaction. It stores the sender's name, receiver's name and the amount being sent. In order to successfully create a transaction, the sender & receiver names cannot be the same and a value greater than 0 must be sent.
Noteable methods: Basic setters and getters.

Enum: BlockChainKeys

This enum object was created in order to neatly contain the keys of values given in a block chain text file (index, timestamp, sender, receiver, amount, nonce and hashmap). Using an enum as opposed to strings helped reduced the chances of mistyping keys when accessing values in the Enum Maps.

Class: Block

Block represents a simple block in a blockchain that contains only one transaction.

```
Public Block(EnumMap<BlockChainKeys,String> blockEnumMap)
```

This constructor was specifically created to handle creating blocks from the text file following the format given for the assignment. Given the 'blockEnumMap' it is able to go through the required keys and create a Block.

```
Public void updateHash() **NONCE METHOD**
```

This method creates a hash from the toString of the block. In order to achieve the proof of work (zeros for the 1st 5 characters in the hash) I decided to create a nonce from random ascii characters from [33,126] of max length 9. When the length became 9 the nonce was split in half and rebuilt up with more random characters. The method will keep going this until the hash value is valid. Random ascii characters were used because there are a wide range of possible combinations which is perfect for guess and check approach.

```
public ArrayList<String> fileRepresentationOfBlock(int location)
```

Creates a file representation of a block at the given locations. A file representation of the block is an array containing index, timestamp, sender, receiver, amount, nonce and hashmap.

```
private boolean isValidTransaction(int location)
```

Checks if the transaction @ location is valid by looking at the previous transactions and verifying the sender has received enough money to send money.

```
Public boolean validateBlockChain()
```

Santos Gagbegnon
300010769
Miner ID: sgagb074

Checks if the blockchain is valid by ensuring all hashes begin with “00000” & were formed via their toString, previous hashes are properly linked, indices are correct, and all transactions are verified,

```
public static BlockChain fromFile(String fileName)
```

Creates blockchain by reading specified file. Does this by collecting the 7 attributes of a blockchain in the file, creating a block object and adding it to a newly created blockchain.

```
public void add(Block blockToAdd)
```

Adds and links a new block to the blockChain

```
public int userBalance(String username)
```

Calculates user balance by (money they’ve received) - (money they’ve sent)

```
public void toFile(String fileName)
```

Saves the blockChain to the specified file

Table 1: Number of Trials to Create Proper Hash

Trial #	Value
1	2746769
2	254947
3	1046613
4	1715434
5	439620
6	1123396
7	3497123
8	3784073
9	1436556
10	1665601
AVERAGE	1771013.2

Details on more methods are accessible via the comments in the actual project code.