Design Additions, Changes, and Justifications

Note, Additional testing script runs with the command:
java -cp . com.cscie97.ledger.test.TestDriver additional.script

+ Class Level Variables, CommandProcessor class:

I added class level variables in the CommandProcessor Class to assist with running ledger commands and storing user authentication type.

- The *currentLedger* variable stores a created ledger in memory in order to run ledger methods.
- The *accessType* variable stores an access type which is verified before creating a new ledger. The access type is set via CLI to either ‘admin’ or ‘user’. (Ledger Service Design Document, Actors, page 4).

+ ledger.script modifications

- I added a script to the ledger.script to accommodate the required accessType as specified above.

```
# specify access-type
access-type admin
```

- Quotes included in the ledger.script document were directional, I changed them to non-directional quotes to mimic the CLI character set.

```
“fund account” ---> "fund account"
```

+ Class, MerkleTree:

I added Vinay Prabhu’s implementation of a Merkle Tree for hashing:

- *getMerkleTree(ArrayList<String> Transactions):* Recursively generates a merkle tree from arraylist and hashes, returning merkle root. Used to generate and verify block hashes.
- *getSha(String input):* returns a SHA-256 encoding of input string, used to generate and verify block hashes.

+ fundLedger() Method, Ledger Class:

“In addition to creating the genesis block, and the master account and initializing the master account balance.” (Initialize Ledger, Page 5)

- The fundLedger(): a void method which creates the master account, and adds the genesis block to the ledger's blockmap. It would not make sense to add this functionality to the constructor. This method is run after the instantiation of a ledger, and will throw an error if a block has already been committed to the ledger.

+ Helper Methods, processTransaction(), Ledger Class:

Private methods added to improve readability of processTransaction() method.

- *computeHash():* Handles the logic for hashing of blocks.
- *blockFull():* Handles the logic of creating a new block when current block is full (10 transactions).

+ *hashCode()*, Block class

"The hash of the current block is computed based on all properties and associations of the current Block except for this attribute. Use the Sha-256 algorithm and Merkle tree to compute the hash per the requirements. (Block Properties, Page 9)

- *hashCode():* Returns a hash of all attributes in a block except for the block hash, used when committing a block to the blockchain

+ *toString()*, Block, Transaction classes

(Page 13, Command Processor)

Get Bock
Output the details for the given block number.

```
get-block <block-number>
```

Get Transaction
Output the details of the given transaction id. Return an error if the transaction was not found in the Ledger.

```
get-transaction <transaction-id>
```

- toString() added to control the output when fetching a Block or Transaction via CLI

Did the design document help with the implementation?

Certainly. I have implemented a blockchain ledger before using Solidity. Even still, having the design document to reference made the implementation much easier. I spent much less time going down a road only to find that the functionality I was adding was not in line with the desired functionality of the program. I look forward to the next assignment, I think it will be a challenge to write the design document.

How could the design have been better, more straightforward, or made the implementation easier?

There were several things that I felt needed to be clarified with the instructors, that may have been avoidable with a more thorough design document.

- Error throwing with respect to the command processor was left up to the discretion of the implementer
- The design could have given a formula for hashing the attributes of a block
- The design document's language around account identifiers is rather vague. At times the document says that the createAccount function accepts a unique address from users(class diagram) and at others says that the createAccount() method is assigned a unique address(requirements). Following the class diagram we can see that the account only has two values, which clarifies that the value is not automatically generated.
- This same issue persists with transaction id. Not a big deal by any means, but slightly unclear.
- The quotation marks in the ledger.script file were directional, whereas the quotation marks in a CLI are non-directional. This threw me for a loop until I corrected them.

Citations:

Prabhu, V (2019) MerkleTree source code.
https://medium.com/@vinayprabhu19/merkel-tree-in-java-b45093c8c6bd