Assignment 1

Results and Implementation Changes

Brett Bloethner

Implementation Changes

There were a handful of implementation changes I made in my project. I've broken them out into a bullet point list below where I've also added details and reasoning for some of the implementation changes.

- A LedgerException is thrown for getTransaction() if the requested transaction does not exist
- Various getters and setters are used in each class as well as add methods to add items to maps
- The pattern for account balance storage in blocks was modified so that blocks store maps of account balance ints keyed to account Id strings. This made it much easier to avoid the account reference and copying issue many other students had when trying to store account data in a block they wanted to be immutable. Due to this change, the ledger now has to store the most up to date copy of the Account objects (both committed and non-committed). Right before a block is to be processed the Accounts stored in the ledger accountMap get their Ids and balances copied into the blocks accountBalanceMap.
- A generateHash and getMerkleRoot methods were added to the Block class.
  - Using a generateHash method made generating the hash more convenient and obvious when working in the ledger class as opposed to having the hash be generated as some sort of side effect of another feature
  - getMerkleRoot allowed for recursive calculations of the different layers of the Merkle tree all the way up to the top root layer
- The ledger has createTransaction and processTransaction methods since processTransaction requires a Transaction object arg (as per the docs). createTransaction creates that transaction object from the strings and ints provided by the command processor
- A currentBlock block was added to the ledger service, this is where the currently worked on block lives while transactions are added before getting committed to the ledger and a new current block is created and worked on

- A checkBlockSize method was added to the ledger for convenience. This method will evaluate the current blocks transactions map size and commit the block if necessary. It's called at the end of processTransaction().
- Some toString() methods were overridden in order to provide more intelligible terminal output
- A utility class was added. This is where the method for generating the SHA 256 hash lives. The idea was that this hashing could possibly be used in multiple classes for validating the blocks, transactions etc…

Design Document

I think the design document was a huge help in making the blockchain app easier to build. There were a few parts of the document that were a bit confusing. One example would be where parts of the document suggested that the ledger service had both processTransaction and createTransaction methods while other parts of the documentation showed the ledger service only having a processTransaction method. Overall, I think the implementation would have been far more difficult without the document's class definitions and diagrams.

Design Impressions

I think the design was pretty straight forward. I don't really see the value in having the Accounts stored on the block though. The values of each account can be derived from looking at all the transactions in each previous block in the blockchain and those transaction are immutable (thanks to the hash chain), so the account balances derived from running through all the transactions would also be immutable in a way. Storing the accounts certainly makes getting account balances more convenient but I think that comes at a cost of added complexity in the ledger and block classes.

Results

The test command file I used in this project was based on the original ledger.script file and additional commands were added to verify more functionality.

The input file is **test.script**

The output is available in the txt file **testScriptOutput.txt**

Resources

There were two parts where I got stuck and had to resort to solutions whose design implementations came from an outside source as opposed to standard Java documentations.

- Reading a file line for line. Specifically the regex that separated strings on spaces except for when surrounded by quotations.
    - https://stackoverflow.com/questions/7804335/split-string-on-spaces-in-java-except-if-between-quotes-i-e-treat-hello-wor
- Properly using SHA 256 hashing against a string using only Java standard libraries
    - https://stackoverflow.com/questions/5531455/how-to-hash-some-string-with-sha256-in-java