Matthew Thomas
CSCI e97 — Assignment 1 — Results

**Were any changes made to the proposed design? How do they continue to support the requirements?**

- In my implementation, the Ledger does not include a genesisBlock association. The requirements do not describe any special behaviors/features unique to the genesisBlock compared to any other block in the blockchain. In the class dictionary the only differences listed are that the previousHash and previousBlock fields are empty (as it is the first block in the chain). My implementation continues to support the requirements because you can still access this first block, genesisBlock, by calling `getBlock(1)`, just like you can for any other arbitrary block. This removes the need for a separate association for one specific block; rather, that block can be accessed dynamically.
- In my implementation, `createAccount()` returns an Account object (as indicated by the class diagram), rather than the account address/identifier (as indicated by the class dictionary). The design lists both. My decision to go with an Account object keeps with the requirements in that you can still call `getAddress()` on the account object to retrieve just the account address/identifier.
- In my implementation, `processCommand()` also takes the lineNumber as an argument in addition to the command. This continues to support the requirements as it does not change the behavior of the method, it simply makes it easier to throw a `CommandProcessorException()` and move on to process other commands in an input file.
- As mentioned in Piazza post @16, my computed hash only includes the seed, blockNumber, previousHash, transactionList, and accountBalanceMap, and does not include the previousBlock association. As the instructor's answer mentions, this supports the requirements as "the previousHash attribute is sufficient to link the blocks together."

**Did the design document help with the implementation?**

Except for a few bits of confusion on my part at the start of the process, the design document did help a lot with the implementation. Overall, the design document was very detailed and thorough which made for a generally easy process. It did take me a few cycles of reading the spec and writing code to start discovering inconsistencies and questions (see the next section below). In that sense, I think the design nailed the big-picture aspect in terms of structure.

I also found this assignment in general to be very beneficial. Most other class assignments are usually along the lines of "Here's some starter code and the two new methods to write", which can also be challenging. However, starting from scratch and having to build all the pieces by oneself really helped to solidify a lot of programming

practices I've learned over the last handful of semesters. Having the design document gave me a solid foundation that was supplemented with lots of Java documentation.

**How could the design have been better, clearer, or made the implementation easier?**

As noted in several threads on Piazza, there were many questions about the purpose/role of the Ledger. One example was the requirement listed under Accounts stating that "The Ledger will manage a list of accounts." While this is technically true, as Eric demonstrated in section on Tuesday, September 17, the Ledger really manages the accounts because it manages Blocks (which in turn manage a list of accounts).

In addition, I had questions about what certain methods should return; namely, if results should be based on the last completed block or not. getAccountBalance() and getAccountBalances() both specified "most recent completed block", but getTransaction() does not. Updates to the test script, clarifications on Piazza, and working through what made the most sense in my own implementation helped solidify the "best" approach. Having a "default" of most recent completed block also helped make the implementation easier, since that means fewer paths/conditions to code.