Data Structures

- Vector<T>

I used `Vector` in many places since it's a dynamic array of objects which allows random access and is also considered to be thread-safe as well compared to the `ArrayList` which isn't.

The usages for Vector were in:

- `Blockchain`, where the chain is composed of a `Vector` containing the list of blocks creating it.
- List of `ClientSockets`, I had to hold a list of multiple `ClientSocket`s like in `User.Client`, as temporary lists in GUI and on the Server.
- List of 'Objects', sent to/from clients or server to a client containing objects as a respond to the connection.
- List of 'ClientSockets', used in the `ClientSocketDAO` to retrieve multiple `ClientSockets` from the database.

- LinkedBlockingQueue<T>

This one was used in the `transactionPool` as a producer-consumer design pattern in order to add to the pool and retrieve from it from multiple threads while still being thread safe.

- HashMap<T, K>

Used twice, first time in `Blockchain.verifyChain()` to insure that all the transactions have unique ID's, and in the `User.getAllNetworth()` to find the balance for each user, in both cases taking the advantage of O(1) average access time for the HashMap.