SOLID Principles

- The Single Responsibility Principle (SRP)

To achieve the SRP, I built the system from bottom to top, making sure that every Class is responsible of its own actor, for example:

- `Transacton`, this class holds information about the Transaction like the Sender, Receiver, the Amount and the Signature.
- To further expand the system, a `MinedTransaction` was created, which has the responsibility of holding information about the mining such as the Miner, the Mining Fees and the Mining Reward on top of the transaction information.
- A `Block` is supposed to encapsulate a `Blockable` object (which is a `MinedTransaction` in our application) and add a layer to proof the work of the miner.
- `Blockchain` contains a list of blocks and insures their validity.
- 'User' class is an API to expose the `Blockchain` to users as a CryptoCurrency and thus insure that the chain has a valid sequence of transactions.
- `Client` class took the functionality of communication with other users and were encapsulated in a separate inner-class.
- `ClientSocket` is responsible for saving the contact information about a user.

- The Open-Closed Principle (OCP)

This principle applies to the project since it allows you to design any kind of data to be used in the blockchain, where you can easily extend the functionality by writing your own `Blockable` class and implementing an API that uses the `Blockchain` with this new `Blockable` class without any need to rewrite any code related to the `Blockchain` and what follows it like `Block` class, `Blockable` interface and the factory classes for them.

- The Liskov Substitution Principle (LSP)

It's not easy to apply this principle to the project due to the requirements since the `User` API is required to implement a crypto-currency which make it not easy or efficient to implement a `User` that is generic and then use it specifically later for `MinedTransactions`.

- The Interface Segregation Principle (ISP)

The implementation of `User` API depends on the implementation of `Blockchain` and the implementation of `MinedTransaction`, but they are different classes, and neither of them depends on the other, thus if we had another `Blockable` class other than `MinedTransaction` and another API was build around it along with the `Blockchain`, changing the implementation of the `MinedTransaction` will not in any way affect the new API based on the new `Blockable` class.

- The Dependency Inversion Principle (DIP)

The system on the Server side only has external dependencies outside the build-in java classes, which is the `MySQL` dependency, which I believe is trusted enough to work with.

The system also reply on other dependencies which built in, these dependencies are unavoidable (Like: String, List<T>, KeyPair) and proven to be safe to use and guarantees the back-ward compatibility with the future updates.