Team:

- Sergey Frolov
- Ian Martiny
- Shirly Montero Quesada

Title: OTR-Messenger

Discussion:
In our new class diagram we didn't refactor the server part, but the client side was heavily changed. When using abductive reasoning to assess our class diagram in the implementation process, we ran into design difficulties –e.g. some of the classes were bloaters. We moved some methods to increase cohesion and adopted two design pattern. Initially, we thought of using four patterns, Observer, Proxy, Memento, and Command. Since then, we decided that only Facade and Memento were good fits.

The server side uses **Facade** for assets handling (Figure 1). AssetHandler has functions for getting/saving user info and x509 certificates, which are delegated to various managers. In this way, we are providing a level of abstraction that can easily change the form in which any subset data is stored and change the database, such that we won't have to change code anywhere, but in those handlers.

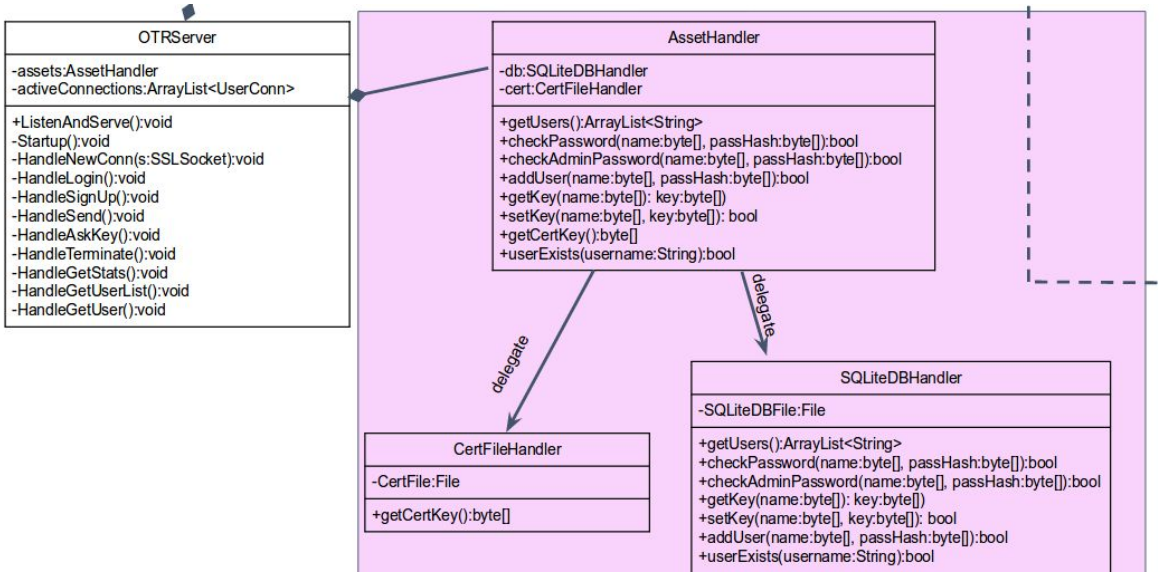

Figure 1. Facade pattern used

Additionally we implement **Memento** on the client side to keep track of the history of communications between clients (Figure 2). The Message class keeps track of particular messages (as the Memento) and the History class keeps track of all messages between two clients (as the Caretaker).

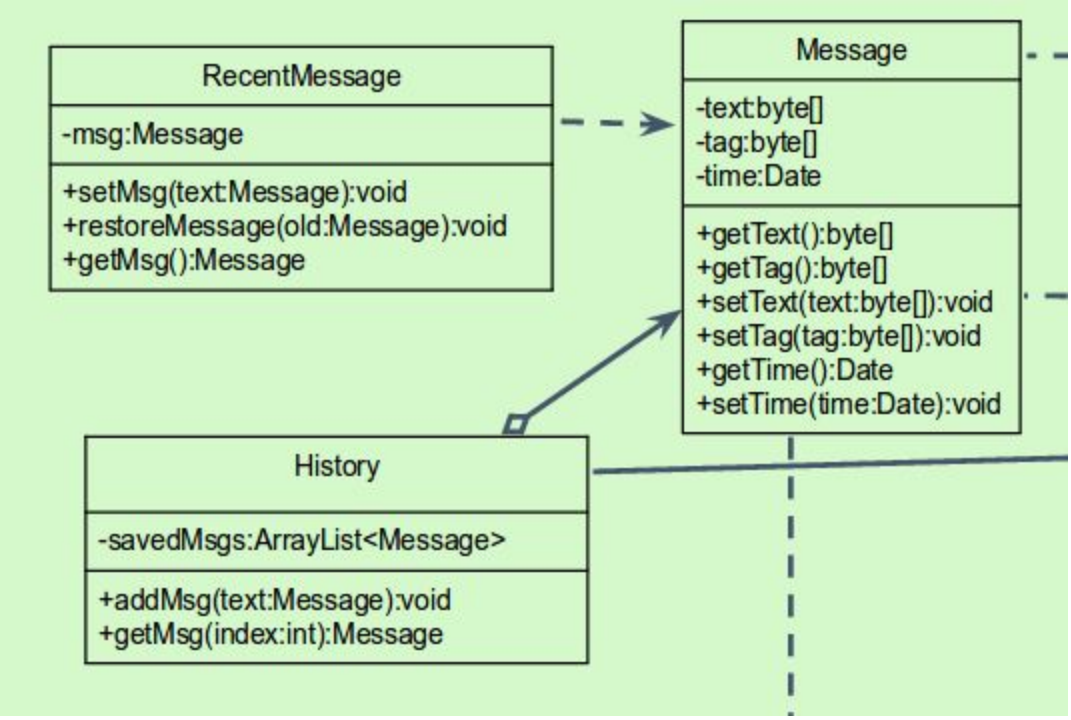

Figure 2. Memento Pattern used

Original Class Diagram: see Figure 3.

New Class Diagram: see Figure 4.

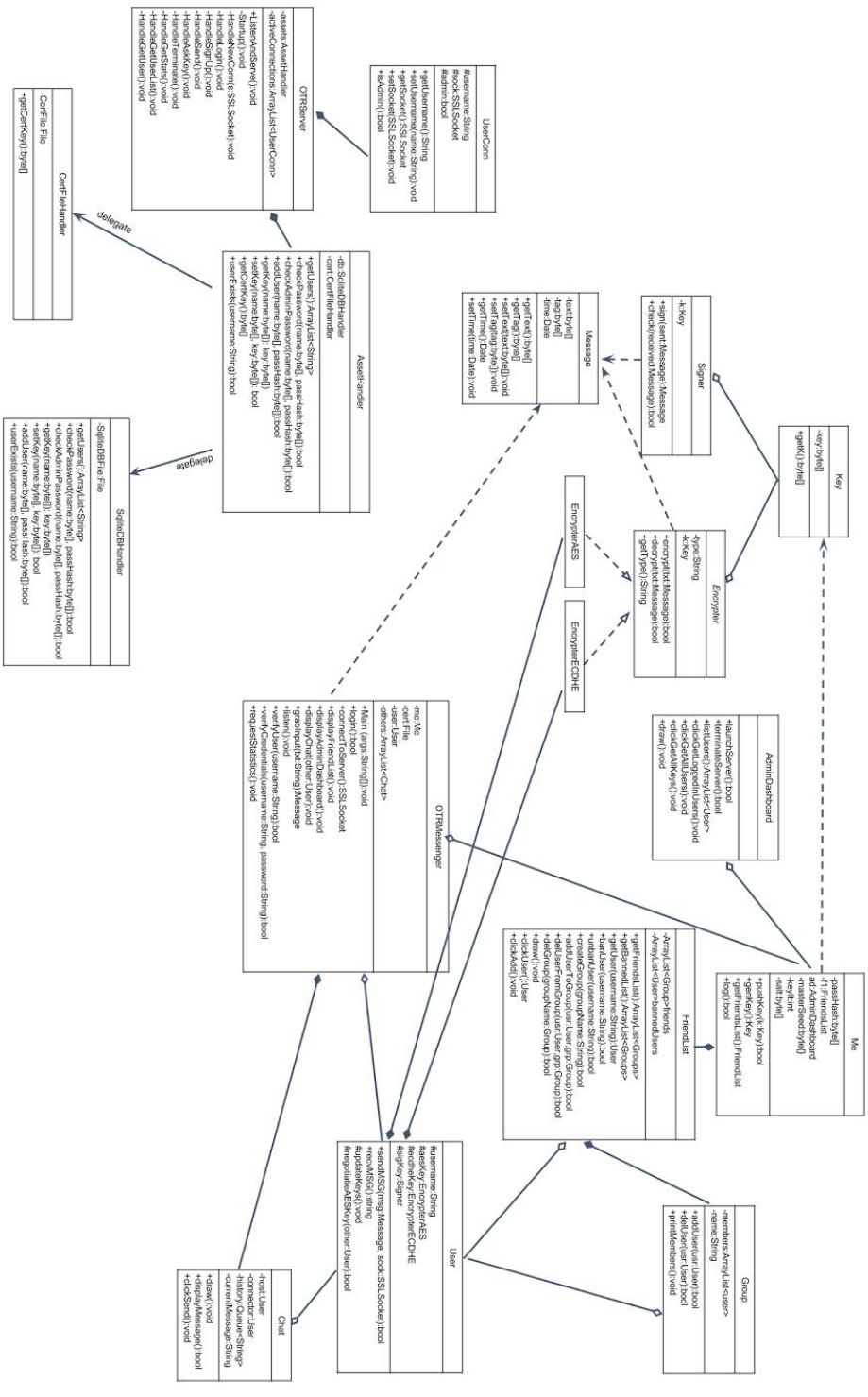

Figure 3. Class diagram (see original in folder support materials)

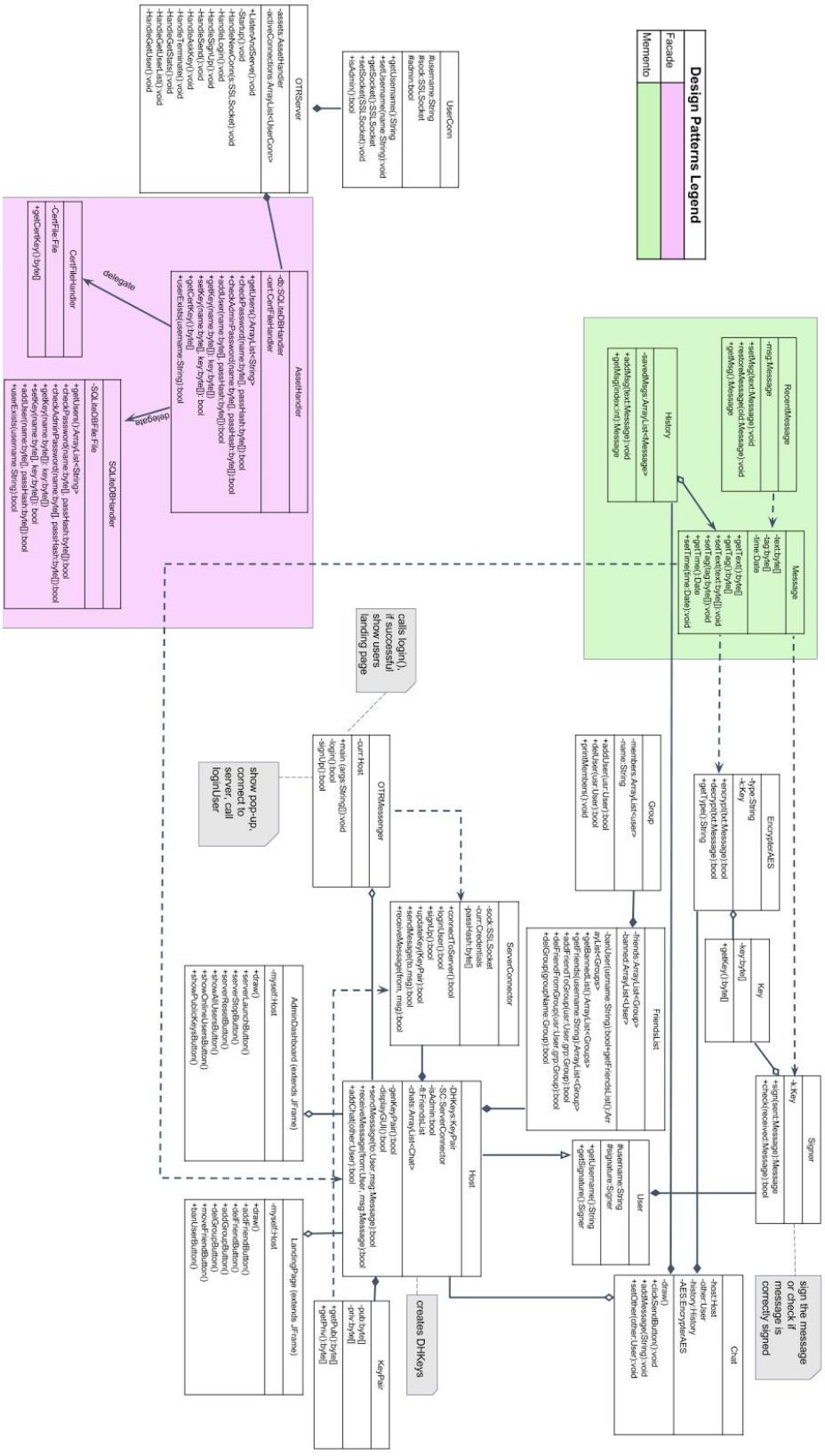

Figure 4. New class diagram (see original in folder support materials)