TEXTUAL DESCRIPTION:

Comments about the design

We have chosen to use the design given to us in the project description and in the skeleton. This is because we found it sufficient and did not need to add other classes in order for the chat to work as described in the spesification. If we were to implement some of the optional spesification like chat rooms for example, we would possibly have to create a new class.

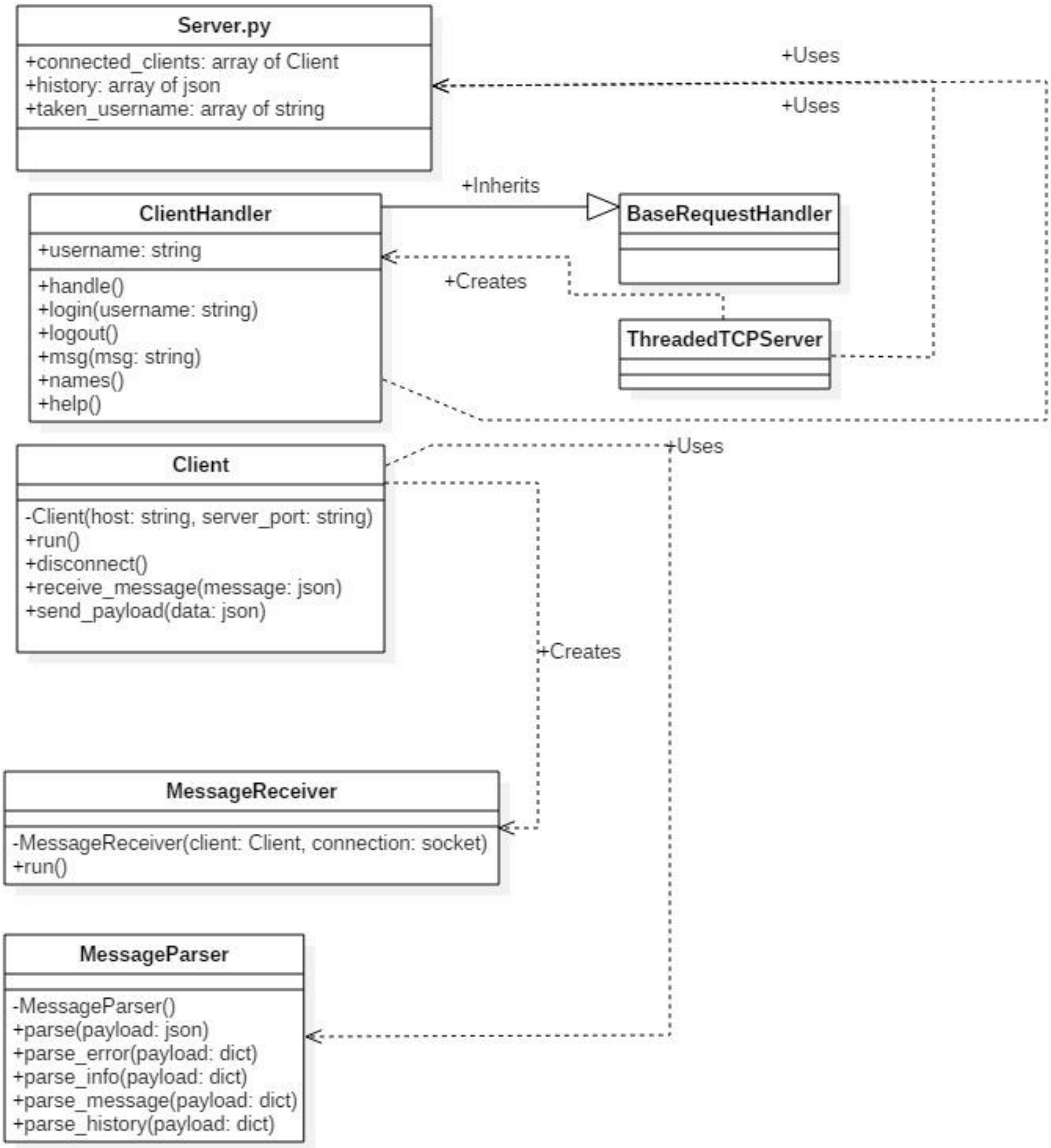
Comments about the class diagrams:

In the class diagrams, we have not specified the variables in predefined classes imported from Python libraries such as BaseRequestHandler, but included our own added variables and methods. This is included to show inheritance and cooperation between classes. The constructors for the different objects are the only private methods.

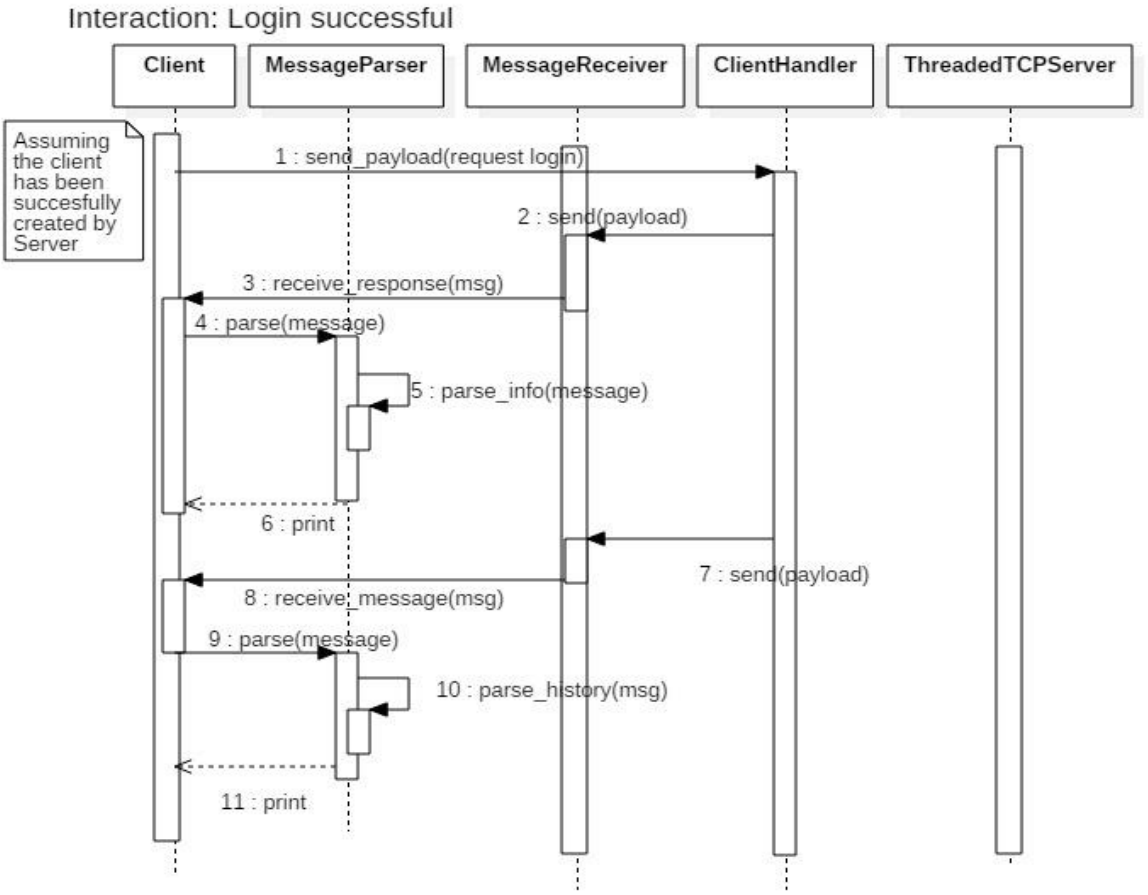
Comments about the sequence diagrams:

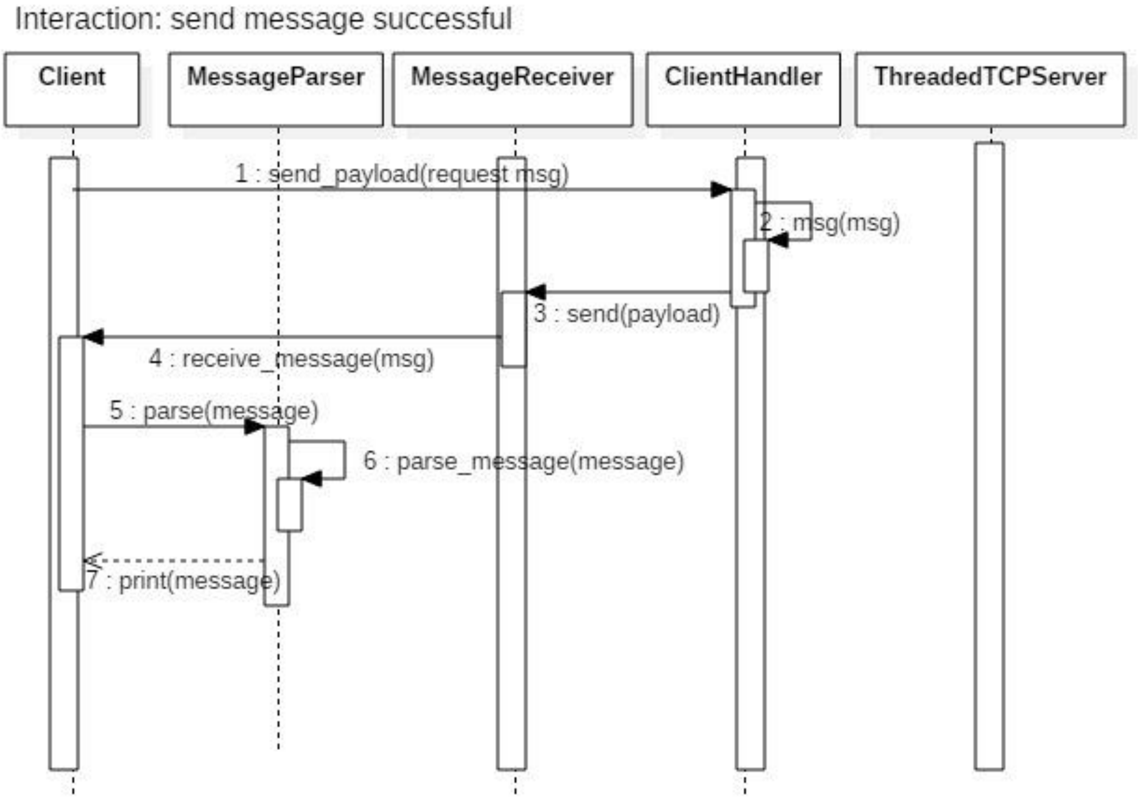
In the sequence diagrams we have focused on the use cases: Login successful, send_message successful and error handling when not logged in, because the initialization process is already given in the example files, and because we think these use cases are sufficient to show interaction between the different classes. In all of the sequence diagrams we have assumed that initialization is already completed and was successful so that requests between client and server can be shown. Therefore there will be no interaction between the ThreadedTCPServer and the other classes in these sequences because the named class will only interact in the initialization process.

Class Diagram

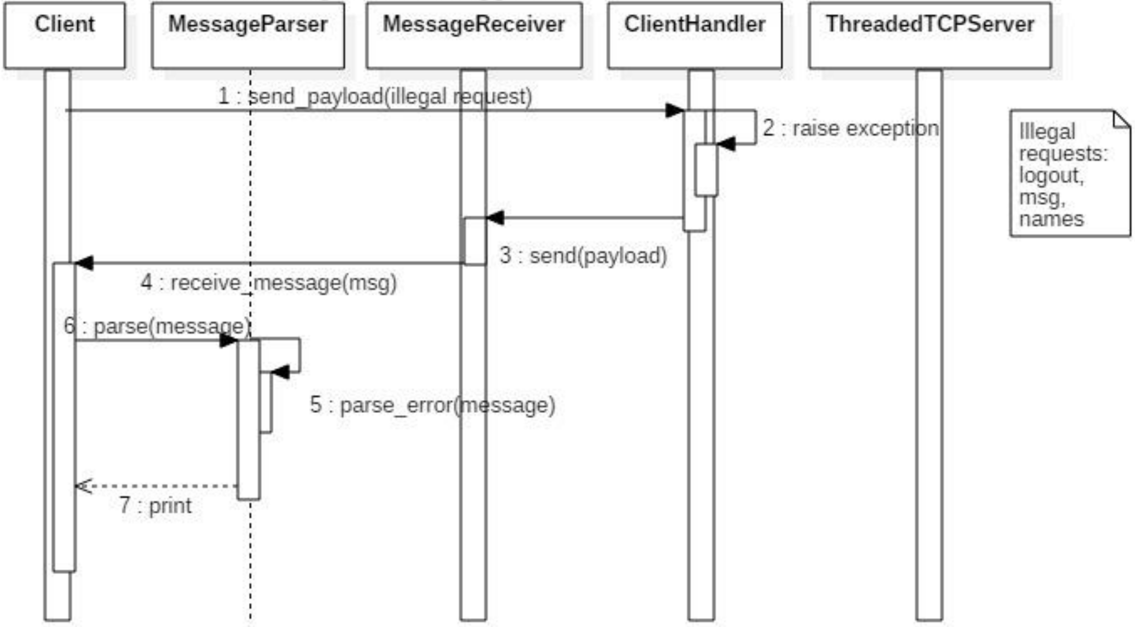

Server.py
+connected_clients: array of Client
+history: array of json
+taken_username: array of string
+Uses
+Uses
ClientHandler
+username: string
+handle()
+login(username: string)
+logout()
+msg(msg: string)
+names()
+help()
+Inherits
BaseRequestHandler
+Creates
ThreadedTCPServer
Client
-Client(host: string, server_port: string)
+run()
+disconnect()
+receive_message(message: json)
+send_payload(data: json)
+Uses
+Creates
MessageReceiver
-MessageReceiver(client: Client, connection: socket)
+run()
MessageParser
-MessageParser()
+parse(payload: json)
+parse_error(payload: dict)
+parse_info(payload: dict)
+parse_message(payload: dict)
+parse_history(payload: dict)

Sequence diagrams for different use cases





Interaction: error handling when not logged in


Client
MessageParser
MessageReceiver
ClientHandler
ThreadedTCPServer
1 : send_payload(illegal request)
2 : raise exception
Illegal requests: logout, msg, names
3 : send(payload)
4 : receive_message(msg)
6 : parse(message)
5 : parse_error(message)
7 : print