Simple Client/Server Application (10 points)

Within this assignment a simple Client / Server application using plain Java Sockets should be implemented. This will be evaluated on the 19th of March!

The server should offer a simple computation service. The three binary operations **+** (addition), **-** (subtraction) and ***** (multiplication) as well as the unary operation **fac** (Factorial) should be supported for integers. A client chooses one of the operations, picks one or two arguments (depending on the service) and submits a request to the server. On the server side, the request is processed and the result is returned.

Assignments:

1. Protocol
   Define the protocol to be used for the communication between the client and the server.
   a. Start by defining the set of all required messages
   b. Describe the format of all those messages
   c. Illustrate the behavior of the server by sketching a transition system. (Hint: simply draw a graph where vertices are states and edges are transitions between states. Label transitions with the message type which has to be received / sent or socket events (<open>, <close>) and conditions in which the corresponding transitions shall be triggered. Mark the initial state and eventual terminal states)
   d. Illustrate the behavior of the client using the same means. **(for all: 3.5 points)**

2. Protocol (extended)
   Extend the protocol of the first task by a user authentication mechanism – that is, users have to authenticate themselves before the service is offered. The authentication consists only of stating the name (very secure). If the name is known, the service is offered, otherwise it is denied. **(1.5 points)**

3. Implementation
   Implement the following components using Java (TCP) Sockets (the following architecture is just a suggestion – components may be implemented differently as long as the same features are supported):
   a. Implement a utility class *Protocol* defining a constant for a server port number as well as two methods *request* and *reply*. The first should accept a socket connected to the server as an argument and uses the associated input / output streams to realize the client side of the communication. Additional parameters specify the operation to be conducted (+,-,*,fac) and the operands to be passed. The reply method simply accepts a socket connected to the client and conducts the server's end of the communication protocol.

   b. Implement the Server, which opens a new socket and waits for client connections. In case a client is connecting, the client's request shall be processed according to your protocol by forwarding it to the *reply* method of the protocol class.

c. Implement a Client component which opens a connection to a given server (IP) and requests one of the offered services. Encapsulate the entire request handling within a single method which accepts the address of the server, the username, the operation and the operands as arguments and returns the computed result. Handle potential errors/exceptions within this method. Internally, the *request* method should be utilized. **(for all: 5 points)**

For all implementations, ensure proper exception and resource handling (e.g. close all streams + consider the possibility of IOExceptions and authentication errors).