Protocols

The rules governing client-server interaction

Every individual command sent by the client should be a String, by which I mean a sequence of ***one-byte*** ASCII characters followed by a network newline ("\r\n"). The server will respond with either a String or integer, depending on the nature of the transmission.

For example, usernames and gameIDs will be sent by the server as Strings while return codes will be sent as four-byte ints. While this is pretty much self-evident, in the document below String transmissions will be coloured green while ints will be coloured blue.

Since I'm not really sure what the convention is, I should make it clear that when I say the server sends an int, I mean it send the 4 bytes that correspond to that int and that's it. No network newline afterward. So if you're expecting 4 ints and then a string, perform 4 four-byte reads to get the 4 ints and then start reading the string, keeping an eye out for the "\r\n."

Rather than depending on the integer constants below, there's a class protocol.Protocol which contains one public static inner class for each possible command. Each of these inner classes contains fields, one per possible return code. The Protocol class itself contains fields for the more general return codes, for things like server error or format error. This class can be imported into a project to avoid dependency on individual integer return codes.

We note a couple constants:

- -3 – A return code of -3 means generally that something is wrong with the format of the command. For example, a login command that places a comma between username and password instead of a space, or a move command that omits the "->"
- -2 – The server returns -2 if a client attempts to issue a command without logging in a user. "Logging in" a user means that the server will associate the client with the logged in user, allowing it to access data and make moves on the client's behalf. The only requests clients can make of the server without having logged in a user are to either login a user or create a new account. All other requests for data must be preceded by the logging in of a user.
- -1 - A response of -1 from the server always means that there was a server error and something went wrong server-side. If the connection hasn't been terminated, the client can try again if they'd like, but should not assume that the last command they issued was processed

Now, we address each potential client-server interaction individually.

- Creating a new account for a user
    - Performing this action will automatically log out any previously logged in user. A client cannot access the data of two players at a time
    - Client – "create username password"
        - ***Note:*** the client should only accept usernames and passwords that do not contain either spaces or commas and are non-empty. Otherwise, corruption of data could occur. If the client does not check for this, the server will not allow the account to be created
    - Server – responds with one of the following:
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – account creation successful, client can now make requests on behalf of the new user
        - 1 – the given username is already in use

        - 2 – the username and/or password aren't correctly formatted. For example, one is non-empty, or contains a comma.
- Logging in a user
    - Performing this action will automatically log out any previously logged in user. A client cannot access the data of two players at a time
    - Client – "login username password"
    - Server – responds with one of the following:
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – login successful, client can now make requests on behalf of the given user
        - 1 – the given username does not exist
        - 2 – the given password is invalid for the given username
    - If the login is successful, the server then does the following:
        - First, it sends an integer, which corresponds to the number of ongoing games that the logged in user is involved in
        - Then it sends that many batches of transmissions, where each batch corresponds to the following data points, each sent on a separate line:
            - GameID
            - Name of White player
            - Name of Black player
            - Open – 0 if this game is open, 1 otherwise
            - State - 0 if it's White's turn, 1 if it's Black's
            - Turn – the current turn number
            - White archived – 1 if White has archived this game, 0 otherwise
            - Black archived – 1 if Black has archived this game, 0 otherwise
            - Draw offered – 1 if the last player to move offered a draw, 0 otherwise
            - Drawn – 1 if the game has ended by draw, 0 otherwise
            - Winner – the name of the player who won the game, if there is one, empty otherwise
            - Forfeit – 1 if the game ended by forfeit, 0 otherwise. Has no meaning if the game is ongoing or ended in a draw
            - White check – 1 if White is in check, 0 otherwise
            - Black check -1 if Black is in check, 0 otherwise
            - Promotion needed – 1 if the player whose turn it is needs to promote a pawn before their turn can end, 0 otherwise
                - Tells the client that, if it is their user's turn, they should prompt the user to promote their pawn
                - See below for how to promote a pawn
        - If the server encounters a critical error, such as a flaw in its local database, it will send -1, BEFORE the initial integer or any batches are sent.
- Creating a new game
    - If the client has already logged a user in, and the user wants to create a new game
    - Client – "creategame gameID open"
        - Note: gameIDs cannot include commas, and must contains at least one alphabetical character
        - Open should be 0 or 1, AS A STRING, and is meant to indicate whether the created game is "open" or not (a list of open games can be viewed by any client of the server, while for closed games the owner has to personally give out the gameID for someone to be able to join)
    - Server – responds with one of the following

        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – game was successfully created
            - Whichever user creates the game automatically plays White
        - 1 – the given game already exists
        - 2 – the given gameID is invalidly formatted
- Joining a game
    - If the client has already logged a user in, and the user wants to join a particular game
    - Client – "joingame gameID"
    - Server – responds with one of the following:
        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – the user has successfully joined the game
        - 1 – the given game does not exist
        - 2 – the user is already in the given game
        - 3 – the given game already has two players
- Loading a game
    - If the client has already logged a user in, they can try and load one of the user's games. This gives them access to the most up-to-date information about that game, both high-level (whose turn it is, names(s) of the player(s), whether somebody has won yet, whether or not there's a draw offer, etc.) and low-level (i.e. the actual state of the board)
    - Client – "loadgame gameID"
    - Server – responds with one of the following
        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – success; the user is a player in the given game, and the board data will be sent shortly
        - 1 – the given game does not exist
        - 2 – the user is not in the given game
    - If the server responds with success, it then sends all the game's information in the exact same order and format as described here
    - If the server responded with success, it then also sends the contents of the requested game's data file, one line at a time. For details on how these files are formatted and how game data is encoded, see the file named Data in this directory
        - The first 4 lines of the file and the last line will be sent as integers, while the 8 in the middle that represent the board will be sent as Strings, as detailed at the beginning of this document.
- Loading all games
    - If the client has already logged a user in, they can ask to be given data about all the games that user is in. The data is sent over in the exact same fashion as after a user is first logged in
    - Client – "loadgames"
    - Server – responds as follows:
        - First, one of the following:
            - -3 – the client doesn't have a user logged in, so the request couldn't be handled

                - -1 – server error; the request could not be properly processed due to an error server-side
                - 0 – success, the client can now expect to be sent the game data they asked for
            - If the server responded with 0, the client can now expect the following:
                - First, the server sends a single integer, equal to the number of games the client should expect to receive
                - Then, the server sends that many batches of transmissions, in the same order and format as described here, in the login procedure
- Getting a game's data
    - If the client wants to get the data associated with a particular game (for example to check if its local data is out of date), it can use this command
        - This is different from loading a game because loading a game means getting the BOARD-level details. This command is for getting the exact same higher-level details about the game as are sent during the login process, like the name of the game, the name(s) of the player(s), etc.
        - This command is mainly meant for use by clients as a means of refreshing the data they are storing to make sure it's up to date.
    - Client – "getgamedata gameID"
    - Server – responds as follows:
        - First, one of the following return codes:
            - -3 – the client does not have a user logged in, so the request couldn't be handled
            - -2 – the content of the command is invalidly formatted
            - -1 – server error; request could not be properly processed
            - 0 – success; the user is a player in the given game, and the game's data will be sent shortly
            - 1 – the given game does not exist
            - 2 – the user is not a player in the given game
        - Then, if the server sent a success code (0), it will send the given game's data in exactly the same manner as described here
- Getting a list of Open Games
    - Any client can get a list of games that have been marked as "open", and allow their users to join these games (using the "joingame" command as usual), if someone else hasn't joined them first
    - Client – "opengames"
    - Server – responds as follows:
        - If an error occurs, the server will send -1, as an integer, and won't send anything else
        - Otherwise, the server will first send a non-negative integer equal to the number of games for the client to expect
        - Then, it will send that many batches of transmissions of data, following the exact same protocol as specified here, in the login procedure
- Sending a move
    - If the client has already logged a user in, and the user wants to make a move in a game that he is playing
    - Client – "move gameID src_row,src_col->dest_row,dest_col"
        - src_row,src_col is the square occupied by the piece that is moving
        - dest_row,dest_col is the square the piece is moving to

        - Where row 0 is white's back row, and column 0 is the first column from the left (from white's perspective)
    - Server – responds with one of the following:
        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – move was successfully made; game has been updated
        - 1 – game does not exist
        - 2 – the user is not a player in the given game
        - 3 – the user IS a player in the given game, but an opponent has not yet joined the game, so no move can be made
        - 4 – the given game is over; one player has already won, or a draw has been agreed to
        - 5 – it is not the user's turn to make a move
        - 6 – it is the user's turn, but they need to promote a pawn rather than make a normal move
        - 7 – the user can't move, but instead has to respond to a draw offer
        - 8 – the move is invalid (one of the squares is off the board, move is invalid for the selected piece, etc.)
- Promoting a pawn
    - If the client has logged a user in, and the user wants to promote a pawn which has reached the enemy's back row
    - Client – "promote gameID charRep"
        - Where "charRep" is one of the following:
            - "r" – promotes the pawn to a rook
            - "n" – promotes the pawn to a knight
            - "b" – promotes the pawn to a bishop
            - "q" – promotes the pawn to a queen
    - Server – responds with one of the following:
        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – promotion successful
        - 1 – gameID does not exist
        - 2 – the user is not a player in the given game
        - 3 – the user doesn't have an opponent in the given game
        - 4 – the game is over, so a promotion can't be made
        - 5 – it is not the user's turn to make a move
        - 6 – the user doesn't have a promotion to make
        - 7 – charRep is invalid
- Offering a draw
    - If the client has logged a user in and it's their turn in a game, the user can offer their opponent a draw, making it the other player's turn to respond
    - This same command is also used to accept an active draw offer. That is, if there is no active draw offer, the server will assume the client wants to extend one. If there IS an active offer, the server will assume the client wants to accept it.
    - Client – "draw gameID"
    - Server – responds with one of the following:
        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted

        - -1 – server error; request could not be properly processed
        - 0 – draw successfully offered/accepted
        - 1 – the given game does not exist
        - 2 – the user isn't in the given game
        - 3 – the user does not have an opponent in the given game
        - 4 – the given game is already over
        - 5 – it is not the user's turn
- Rejecting a draw offer
    - If the client has logged a user in and it's their turn in a game, the user can reject a draw offer that their opponent has made
    - Client – "reject gameID"
    - Server – responds with one of the following:
        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – draw successfully rejected
        - 1 – the given game does not exist
        - 2 – the user isn't in the given game
        - 3 – the user does not have an opponent in the given game
        - 4 – the given game is already over
        - 5 – it is not the user's turn
        - 6 – there is no active draw offer from the opponent to reject
- Forfeiting
    - If the client has logged a user in and it's their turn in a game, they can forfeit that game to end it prematurely
    - Client – "forfeit gameID"
    - Server – responds with one of the following:
        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – forfeiture successful
        - 1 – the given game does not exist
        - 2 – the user is not in the given game
        - 3 – the user does not have an opponent in the given game
        - 4 – the given game is already over
        - 5 – it is not the user's turn
- Archiving a game
    - If the client has logged a user in, a user can choose to archive a game that has finished, marking it as archived so that, if desired, the client can allow the user to archive games after they've been played, for example
    - Whether or not a game has been archived is sent along with the rest of the information about each of the user's games immediately after a successful login, and the client can choose to ignore it if they choose
    - Archiving does not practically impact the game at all, it's simply a feature that the client can choose whether or not to implement
    - If the user has already archived the game, returns 0 and nothing changes
    - Client – "archive gameID"
    - Server – responds with one of the following:
        - -3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed

        - 0 – archive successful
        - 1 – the given game does not exist
        - 2 – the user is not in the given game
- Restoring a game
    - If the client has logged a user in, they can allow the user to restore an archived game, which basically just means revert it back to being un-archived
    - Client – "restore gameID"
    - Server – responds with one of the following:
        - --3 – the client does not have a user logged in, so the request couldn't be handled
        - -2 – the content of the command is invalidly formatted
        - -1 – server error; request could not be properly processed
        - 0 – restoration successful
        - 1 – the given game does not exist
        - 2 – the user is not in the given game
- Logging out
    - If the user has logged out of the client, but the client doesn't want to close the connection yet, they should send a logout request so the server can free up some memory associated with the particular user that was signed in
    - This does not close the connection the client has with the server
    - Client – "logout"
    - The server does not respond, but the client should know that it can no longer make requests on behalf of the previously logged in user