Subject: The network game system

1 The task consists in design and implementation of a protocol and application using it, which allows to play the network version of the Tic-Tac-Toe game (with a standard 3x3 board).

2 The application consists of 3 processes:

2.1 The client process, which:

2.1.1 During execution it connects to the server (see 2.2) using the TCP protocol (parameters such as the server's address and its port number are execution parameters). After executed and connected, a client receives from the server the ID of the player, which is automatically assigned by the server.

2.1.2 Any time when not playing, the client may disconnect from the server. In such a case it should send the "LOGOUT" message to the server, which results in termination of the client process.

2.1.3 On user's demand, the client can send to the server a request for current player's list ("LIST"). The server sends back a list, which should include such information as all players' identifiers and their IP/port numbers. This information should be shown on the screen.

2.1.4 When the user decides, the client sends to the server the "PLAY" message and then waits for server's response about the opponent found. The response also includes the information, who begins the game. The duel starts at this moment.

2.1.5 The duel consists in sending to the server a message about a decision of the player which should make his move at the moment and then waiting for a response with the current game status. The game status should be shown to the user (the form is not important). The status message may include information about the end of the game – then the client process shows proper information and returns to its previous state, i.e. waiting for user's commands (PLAY, LOGOUT, LIST).

Multiple independent player processes may be started.

2.2 A process acting as a game server. Only one server is needed in the application, which:

2.2.1 Opens a TCP socket on a port given as a parameter and listens on it for connections of clients (players).

2.2.2 Has a list of currently connected players. The list is updated any time a new player connects or a player disconnects from the server. Each next client gets its unique identifier (any generation method is allowed). The client gets disconnected when it sends the LOGOUT message (2.1.2). The list is sent to clients on demand (the LIST command, 2.1.3).

2.2.3 When at least 2 players report their will to play a game (PLAY command, 2.1.4), a duel should be arranged. One of these players is selected as the one to start and both of them must be informed about the duel. Then, the following steps are performed in a cycle:

2.2.3.1 receive a message form the player, which currently makes a move,

2.2.3.2 update the game state according to this move (we assume that the moves are valid),

2.2.3.3 send to both players a message about current game state and information, who makes the next move,

2.2.3.4 broadcast the state of the duel with the UDP protocol to the audience (see 2.3). The UDP for broadcasts is passed to the server as one of its starting parameters.

When the game ends, both players must be informed about this fact and then finish the duel.

2.3 A viewer process, which listens to broadcast messages using a UDP port. Such a message contains information about the players and current state of their duel. Each received message should be printed. Multiple viewer processes can be executed

simultaneously in the network forming the audience (only one per node due to UDP port limitation).

3 Additional remarks:
- 3.1 The UDP being parameter of the server is the port, **to which** the server should broadcast messages about duel states. The audience must listen using these ports
- 3.2 The way in which the opponents are selected can be any, for instance random. It is only important, that only the players currently willing to play can be selected.
- 3.3 It can be assumed, that all the moves are valid, according to game rules (no need to check their correctness).
- 3.4 The server must allow for multiple concurrent duels.

4 The solutions must be stored in appropriate directories in the EDUX system before 26.01.2020.

5 For a correct solution of the problem the author can obtain up to 8 points:
- 5.1 Up to 3 points for implementation of server and client processes with functionality described in 2.1.1+2.1.2+2.1.3+ 2.2.1+2.2.2, i.e. an infrastructure for duels.
- 5.2 Up to 3 points for implementation of elements allowing to play a game (2.1.4+2.2.3 excluding 2.2.3.4).
- 5.3 Up to 2 points for additional viewer process as described in 2.3 and expanding the server as described in 2.2.3.4.

6 The project archive should contain:
- 6.1 Source files (for JDK 1.8)
- 6.2 Binary (class) files,
- 6.3 Scripts for Linux (and/or Windows system) which allow to run the proposed solution (optional).
- 6.4 The Readme.txt file with author's description and remarks, especially:
  - 6.4.1 **Detailed description of the implementation** (no description or incompleteness of protocol description may significantly reduce the grade), especially with the detailed description of the communication protocol (message formats for connection/disconnection of a new player, messages sent during duels and messages to the audience).
  - 6.4.2 how to compile/install,
  - 6.4.3 how to run,
  - 6.4.4 how to use,
  - 6.4.5 what does not work (if anything).

7 IF NOT DETAILED, ALL UNCERTENTIES SHOULD BE DISCUSSED WITH THE TEACHER. OTHERWISE, THE SOLUTION MAY BE NOT ACCEPTED, IN CASE OF WRONG SELF-INTERPRETATION.