Task 1: The great tournament
SKJ (2018)

The task sketch

There is a permanent tournament in the local network. Each of the players (agents) must play with every other agent a round of a "sailor game". For a given pair of agents A and B the game consists in selection of the two numbers n_A and n_B. The winner is selected by counting from 1 to $n_A + n_B$.

A new player can enter the tournament at **any** moment. It can also leave the tournament at **any** moment but only if it already played with all the other players, which were in the tournament then.

More precise description

The tournament takes place in the local network between independent processes, which can be run potentially on different machines. The only allowed communication mechanism are messages sent through **TCP connections**.

The beginning

Each player is identified by its **name**, its **IP address** and its **port number**. In order to take part in the tournament, an agent must know one active introducing agent (the first player is its own introducing agent). The beginning of the tournament for a given player looks like this:

1. An agent A wants to join the tournament, in which an agent I already participates. Do do so, the agent A sends to the agent I a message *JOIN*.
2. The newly added agent A acquires (in a way proposed by the programmed) knowledge about all other players participating in the tournament.

The duels

After joining the tournament, the agent A performs the following steps:

3. For each active player B (except for itself), the agent A plays a simple game, in which:
   (a) Agent A selects a value n_A.
   (b) Agent B selects a value n_B.
   (c) It is selected, from whom the counting starts.
   (d) Counting from 1 to $n_A + n_B$ determines, who is the winner.

   For instance, if A selected 3 and B selected 7, and it was decided, that counting starts from agenta B, then A wins.

The end

Every agent can finish its work at **any moment**, but only after it **played a gmae with every opponent known**. In particular:

4. An agent at any moment can receive (from the keyboard) a command $QUIT$, being the ending signal. In such case it ensures, that **it is not and will not be a part in any duel**.

5. An agent leaving the tournament should let all other agents know about this fact. All other agents **remove this agent from their memories**.

6. An agent finishes after it prints a summary about all duels played.

Additional functionality - HTTP monitor

As an additional subtask, we propose to add an implementation of a **monitor** – a simple HTTP server, presenting on-line results of duels between tournament participants (agents leaving the game are also removed from the monitor data).

Additional functionality - Fairness of the game

The game, as described in (3) above, **doesn't guarantee its fairness**. In particular, a protocol, in which it is first selected, that counting starts at agent A and then agent B sends its number n_B to agent A, is not fair. In such case agent A can select n_A in such way, that it wins.

Solution of this task should implement each duel in such way, that **chances of wining for each agent are equal**. Acceptable solutions include **usage of simple cryptography**, **building an external randomness source** and others.

Solution speciffication

Program

Each agent is executed with the following parameters:

- NAME: agent name.
- PORT: number of a main communication port.
- IIP: IP of an introducing agent.
- IPORT: port of an introducing agent.

Report

The solution should be additionally described in a file in **PDF** format, containing the following elements:

1. **A general solution description**
   Must include example of execution of multiple agents in the same tournament. It can include a short description of a solution and of used techniques.
2. **Precise solution description**
   Must include (1) description of how the data about the network are kept; (2) how duels after joining a tournament by a new player were implemented; (3) how fairness was granted (if implemented).
3. **Observations, experiments and conclusions**
   As stated. In this paragraph also those solutions can be mentioned, which were not implemented due to lack of time, the results of scaled experiments, etc.

Solutions without a report will not be accepted.

The task

1. Correct and complete project is worth 4 points (+2 additional):
   - For functionality covering correct connection of the players with the tournament one can get **2 point**.
   - For correct and effective solution of playing the duels by a joining player with all other players already in the tournament, one can get up to **2 points**.
   - For implementation of the HTTP monitor **1 additional point** can be scored (but only if the previous elements were also implemented).

- For correct implementation of the fairness one can get up to **1 point**.

2. The program should be written in Java, in compliance with Java 8 standard (JDK 1.8). Only basic TCP socket classes (ServerSocket, Socket) can be used for implementing the network functionality.

3. The projects should be saved in appropriate directories of EDUX system not later than on 25.XI.2018 (the deadline can be changed by the teaching assistant).

4. The archived project file should contain:

   - Full description, according to the report format described above.
   - Source files (JDK 1.8) (together with all the libraries used by the author that are not the part of Java standard library). The application must compile and run on PJA lab's computers.

5. Solutions are evaluated with respect to the correctness and compliance with the specification. The quality of code and following software engineering rules can influence the final grade.

6. UNLESS SPECIFIED OTHERWISE, ALL THE AMBIGUITIES THAT MIGHT ARISE SHOULD BE DISCUSSED WITH TEACHING ASSISTANT. INCORRECT INTERPRETATION MAY LEAD TO REDUCING THE GRADE OR REJECTING THE SOLUTION.