Name : Diana-Alexandra Crintea
Student ID : 29572088
Username : dac1u17

A Series of Indicative Votes - Report

Assumptions, Approach & Implementation

Starting with the assumptions, first and foremost, there are the assumptions settled in the coursework requirements: known number of participants that join the voting session, reliable communication and crash failure tolerance. To those, I added that there should be at least one participant that does not fail in order to transmit the votes to the coordinator, at least one participant to vote and at least two vote options in order to worth starting a voting session.

My approach was to use Sockets to assure the connexions between the coordinator and participants and between participants. In the Coordinator class, in the main thread, there is a ServerSocket which accepts Sockets from the participants. For each participant's Socket, there is a new thread that starts the communication and deals with receiving and sending messages according to the coursework instructions.

For the Participant class, in the beginning, in the class constructor, it is created a new Socket which connects to the Coordinator. Through this socket, the messages are being sent and received to and from the Coordinator in this class's main thread. From this thread, there are a few threads that start to work in parallel. The first thread opens a new ServerSocket which accepts Sockets from other participants. From this one, a thread that reads the received votes starts. Then, there is a thread that sends the randomly chosen vote to the other participant. In the case of flag 1, the vote is sent to a random number of participants between 1 and the total number of participants -1. The other threads work on the same principle but are used to send and receive the lists of votes received in the last step in order to assure that all votes are counted towards the outcome, even when some participants fail.

Lastly, the outcome is calculated depending on the votes. If there is no majority (half of the total number of participants +1) settled on one vote, then the outcome is null, else a clear decision is sent to the Coordinator. For the failed participants, an erroneous output message is sent in order to flag a missing output and to display to the Coordinator the participant that failed.

At the end of each round, an informative message is displayed, stating the outcome, the participants who took part in deciding the vote, the participants who failed and whether there is a tie or no majority meet, case in which a new round starts by creating a new Coordinator with an updated list of options. In the new round will participate only the participants that did not fail.

Extensions

My solution implements all types of extensions. It is sensible to both types of failure, recovering missing votes from the other participants and deciding correctly the outcome even when not all participants send one. Ties are solved by restarting the voting process and reducing the list of vote options. Failed participants do not take part in the new voting session after a tie occurs. In my solution, all participants always come to a consensus with their outcomes.

Design Decisions

The first and the most basic design decision is that if there is only one option in the list of vote options, then there is no need to start a voting session, as all participants would have to vote that single option. On the same note, the voting process does not start when there is no participant to vote (the number of participants is 0). Messages are being printed out accordingly in standard output.

One of the key design decisions is changing the structure of the *OUTCOME* statements sent by the participants to the coordinator as it follows:

- The outcome with a clear decision remains the same – *OUTCOME <outcome> [<port>]*
- The outcome with a null decision (the case when there is a tie or the majority is not met) – *OUTCOME <outcome null> <vote option to be removed> [<port>]*

This decision was made in order to improve the following round in case of a tie, such that the least or one of the least voted options is removed, and not just a random one.