Implementation Description

Coordinator

The coordinator will be our server, here we will create a server socket that will communicate with our participants. The coordinator has a timeout error of 500,000ms. After this period if no client i.e a participant will have joined, the coordinator thread will sleep. We pass arguments in the command line, the arg[0] will be used as the port on which the coordinator will listen on. Using a for loop that iterates the number of times equal to the number of participants, we are able to create multiple threads on the coordinator, This multiple threads will act as our way of communicating with the participants;

Votes

When a participant joins, we add it to the Hash Map of votes with its port number as the key, we initially add the vote for the participant as null. We have to wait for all clients to join before we can send to each client the Details of the participants in the network. Using a while loop we check if the number of participants we have are the same as the ones specified when initializing the coordinator. After sending the details of the participants, we can now send to the participant the vote option. We then wait for the clients to give us the feedback of the outcome so that we can print it out

Participants:

Participants communicate with the coordinator in order to get the in order to get the options and all the clients that have joined. They then randomly vote and create a new server socket that will be used to transmit the votes to each participant. Using streams the total count of the votes can be calculated and sent to the Coordinator