Virtual Voting System

The main intend of choosing the Virtual Voting System is because it enables to use the collections framework to a wider extent. Options like vote list, mapping it to the actual candidates, parsing the maps and list using sets, gave the chances of exploring the power of lists, maps, set, iterators. Considering this, the OOP concepts can be well equipped to abstract, capsulate data wherever needed in the life cycle of the application. Test driven approach enabled to use the collections again for providing test inputs.

It also enabled myself to test my strength and knowledge in collections to take it to further level for programming complex logical requirements and validations.

For the given voting system, a simple strategy pattern is applied. Considering the expansion of system, if processing strategy is expanded in future, the system’s behaviour changes based on the context object.

VoteCounterMain invokes the Votecounter for candidate input - file and input validation and processing. VoteCounter acts as a context to invoke the processing based on what strategy is being invoked.

VoteProcesorStrategy interface is created defining vote processing actions and concrete strategy-ProcessingStrtegy1 implementing the strategy interface.

The Votecounter class extends the Validator to validate all the inputs using inheritance.

VoteSystemException is thrown at places where runtime exceptions are thrown.

AbstractTest has the method to generate random inputs for the system, this class tried to achieve abstraction to VoteTest and OracleGivenTestcases, which are identified test cases used while developing.

System Flow

1. Reads the input from the file which has list of candidates and creates a candidatesMap<String, List<String>>
2. After all the users have keyed in and the word “tally” has been entered, the inputs are validated (duplicate vote checking and invalid vote) , then userinputList List<String> is created and candidatesMap is recalculated (If there are no votes for a particular candidate, the candidate must not be considered for vote processing.)
3. Group the userinputList and eliminate if there is no vote for the candidate and add to a new copyMap<String, List<String>> and noOfVoteMap<String,Integer> which holdes the number of votes for each candidate
4. For each candidate , Iterate each list of values by preference and update the noOfVotesMap and the copyMap by relevancy of the candidate
5. Re Calculate quota if there are any elimination of candidate and the string “continue ” is used to proceed for step 4.
6. If the vote count is greater than quota and able to determine the winner then the string “Finish” is used to announce the winner
7. If all the candidates have equal number of votes, string “rhandomize” allows to eliminate the votes randomly and then chose the winner.