Preferential Vote Counting System

Design

Class Candidate: Holds information about a candidate e.g. name, unique option character and preference (vote) count

Class Ballot: Holds information about a ballot e.g. Set of candidates with preference; A ballot can be invalid if the preferences are not set properly. For example, if the preferences are not set in sequential order starting with 1, then the ballot will be considered as invalid.

This class has a important method called reOrderPreference(), which essentially moves the preferences of each candidates in the ballot up by 1.

For example a ballot has preferences as A 1 B 2 C 3 D4 . When reordering of preference happens, the preference becomes A0, B1, C3 D1.

Interface VotingService: Provides abstract methods e.g. castVote and countVote()

Class VotingServiceImpl: Which has required implementation of the domain logic; e.g castVote() and countVotes()

Class Result: Holds required results for a particular round of counting e.g currentQuota to win , currentVoteCount, set of candidates eliminated (if any) and winner (if any).

Implementation of Counting of votes:

At every round of counting, an internal map called validBallots is computed, in which ballots assigned to each candidate according to first preference is added.

When any candidate gets eliminated, reordering of preference happens for the ballot, so that when validBallots is recomputed, the ballot will be mapped to appropriate candidate (according next highest preference)

If any ballot is found to be exhausted (as per the problem statement), the entry is removed from the validBallots map.

So, at any point of time, quota to win the election can be computed using the validBallots map.

Class Diagram: