5. initializeCandidates(List<String> candidates)

- Time Complexity: O(n) where n is the number of candidates that is added into the priority queue
- Space Complexity: O(n) where n is the number of candidates because every candidate is stored in both the map and heap.

setTotalVotes(int p)

- Time: O(1) assigned a value to a variable.
- Space: O(1) setting an integer.

castVote(String candidate)

- Time: O(log n) where n is number of candidates. Updating in the map is fast (O(1)), but adding to the heap takes log n time because it needs to reorder
- Space: O(1) per vote

castRandomVote()

- Time: O(n) where n is number of candidates. Converting the map keys into a list takes O(n) time. After that, casting the vote is O(log n)
- Space: O(n) where n is number of candidates is added to a temporary list

rigElection(String candidate)

- Time: O(n log n) where n is number of candidates. First, it loops through all candidates to reset their votes (O(n)). Then we insert every candidate into the heap again (each insert is log n), so together it's O(n log n).
- Space: O(n) where n is number of candidates in the sorted and result lists

getTopKCandidates(int k)

- Time: O(n log n) where n is number of candidates because it sorts the entire list of candidates by votes and name to find the top k. Sorting n items is O(n log n).
- Space: O(n) where n is the number of candidates because the sort operation and result list hold n elements.

auditElection()

- Time: O(n log n) where n is number of candidates where we sort all the candidates to print them in order.
- Space: O(n) Sorting creates a new list temporarily, so space usage grows with the number of candidates