Online Voting System

Source Code

```
import java.util.*;


class Voter {

    private String voterId;

    private String name;


    public Voter(String voterId, String name) {

        this.voterId = voterId;

        this.name = name;

    }


    public String getVoterId() {

        return voterId;

    }


    public String getName() {

        return name;

    }


    @Override

    public boolean equals(Object o) {

        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Voter voter = (Voter) o;
        return Objects.equals(voterId, voter.voterId);
    }

    @Override
    public int hashCode() {
        return Objects.hash(voterId);
    }
}

class Candidate {
    private String name;
    private int voteCount;

    public Candidate(String name) {
        this.name = name;
        this.voteCount = 0;
    }

    public String getName() {
        return name;
    }

    public int getVoteCount() {
        return voteCount;
    }
```

```java
    public void incrementVoteCount() {
        voteCount++;
    }
}

interface VotingService {
    void registerCandidate(String name);
    void unregisterCandidate(String name);
    void resetVotingSystem();
    void displayCandidates();
    void castVote(String voterId, String voterName, String candidateName);
    void displayResults();
    void displayVoterDetails();
    int getTotalVotes();
}

class VotingSystem implements VotingService {
    private Set<Voter> voters;
    private Map<String, Candidate> candidates;
    private boolean votingActive;

    public VotingSystem() {
        voters = new HashSet<>();
        candidates = new LinkedHashMap<>();
        votingActive = true;
    }

    @Override
```

```
public void registerCandidate(String name) {
  if (!votingActive) {
    System.out.println("Voting is closed. Cannot register candidates.");
    return;
  }

  String trimmedName = name.trim();
  if (trimmedName.isEmpty()) {
    System.out.println("Invalid candidate name! Name cannot be empty.");
    return;
  }

  if (candidates.containsKey(trimmedName)) {
    System.out.println("Candidate '" + trimmedName + "' is already registered!");
  } else {
    candidates.put(trimmedName, new Candidate(trimmedName));
    System.out.println("Candidate '" + trimmedName + "' registered successfully!");
  }
}

@Override
public void unregisterCandidate(String name) {
  if (!votingActive) {
    System.out.println("Voting is closed. Cannot unregister candidates.");
    return;
  }

  String trimmedName = name.trim();
```

```
        Candidate candidate = candidates.get(trimmedName);

        if (candidate == null) {
            System.out.println("Candidate '" + trimmedName + "' not found!");
            return;
        }

        if (candidate.getVoteCount() > 0) {
            System.out.println("Cannot unregister '" + trimmedName + "' with existing
votes!");
            return;
        }

        candidates.remove(trimmedName);
        System.out.println("Candidate '" + trimmedName + "' unregistered successfully!");
    }

    @Override
    public void resetVotingSystem() {
        voters.clear();
        candidates.clear();
        votingActive = true;
        System.out.println("Voting system has been reset. All data cleared.");
    }

    public List<String> getCandidateNames() {
        return new ArrayList<>(candidates.keySet());
    }
    @Override
    public void displayCandidates() {
        if (candidates.isEmpty()) {
            System.out.println("No candidates registered!");
            return;
        }

        System.out.println("\nRegistered Candidates:");
        int index = 1;
        for (String name : candidates.keySet()) {
            System.out.println(index++ + ". " + name);
        }
    }

    @Override
    public void castVote(String voterId, String voterName, String candidateName) {
        if (!votingActive) {
            System.out.println("Voting is closed. Cannot cast votes.");
            return;
        }

        if (!voterId.matches("V\\d{3,}")) {
            System.out.println("Invalid voter ID format! Must be V followed by numbers (e.g.,
V100).");
            return;
        }
        String trimmedVoterName = voterName.trim();
        if (trimmedVoterName.isEmpty()) {
            System.out.println("Voter name cannot be empty!");
            return;
        }

        Voter voter = new Voter(voterId, trimmedVoterName);
        if (voters.contains(voter)) {
            System.out.println("Voter ID " + voterId + " has already voted!");
            return;
        }

        String trimmedCandidateName = candidateName.trim();
        Candidate candidate = candidates.get(trimmedCandidateName);
        if (candidate == null) {
            System.out.println("Invalid candidate '" + trimmedCandidateName + "'!");
            return;
        }

        candidate.incrementVoteCount();
        voters.add(voter);
        System.out.println("Vote cast successfully for " + trimmedCandidateName + "!");
    }

    @Override
    public void displayResults() {
        if (candidates.isEmpty()) {
            System.out.println("No candidates registered!");
        return;
    }

    votingActive = false; // Close voting when results are displayed
    List<Candidate> candidateList = new ArrayList<>(candidates.values());
    int totalVotes = getTotalVotes();

    // Sort descending by vote count
    Collections.sort(candidateList, (c1, c2) -> c2.getVoteCount() - c1.getVoteCount());

    System.out.println("\nVoting Results:");
    for (Candidate candidate : candidateList) {
        double percentage = totalVotes > 0 ?
            (candidate.getVoteCount() * 100.0) / totalVotes : 0;
        System.out.printf("%s: %d votes (%.2f%%)%n",
            candidate.getName(), candidate.getVoteCount(), percentage);
    }

    // Determine winner
    if (totalVotes > 0) {
        int maxVotes = candidateList.get(0).getVoteCount();
        List<String> winners = new ArrayList<>();

        for (Candidate c : candidateList) {
            if (c.getVoteCount() == maxVotes) {
              winners.add(c.getName());
            } else {
              break;
```

```
                }
            }

            if (winners.size() == 1) {
                System.out.println("Winner: " + winners.get(0));
            } else {
                System.out.println("Tie between: " + String.join(", ", winners));
            }
        }
    }

    @Override
    public void displayVoterDetails() {
        if (voters.isEmpty()) {
            System.out.println("No votes cast yet!");
            return;
        }

        System.out.println("\nVoter Details (" + voters.size() + " voters):");
        System.out.println("---------------------------------");
        System.out.printf("%-10s | %-20s%n", "Voter ID", "Name");
        System.out.println("---------------------------------");

        List<Voter> sortedVoters = new ArrayList<>(voters);
        sortedVoters.sort(Comparator.comparing(Voter::getVoterId));

        for (Voter voter : sortedVoters) {
            System.out.printf("%-10s | %-20s%n", voter.getVoterId(), voter.getName());
    }
    System.out.println("--------------------------------");
  }

  @Override
  public int getTotalVotes() {
    return candidates.values().stream()
      .mapToInt(Candidate::getVoteCount)
      .sum();
  }
}

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    VotingService votingSystem = new VotingSystem();

    while (true) {
      printMenu();
      int choice = getIntInput(scanner, "Enter your choice: ");

      switch (choice) {
        case 1:
          registerCandidate(scanner, votingSystem);
          break;
        case 2:
          unregisterCandidate(scanner, votingSystem);
          break;
        case 3:
          votingSystem.displayCandidates();
          break;
        case 4:
          castVote(scanner, votingSystem);
          break;
        case 5:
          votingSystem.displayResults();
          break;
        case 6:
          votingSystem.displayVoterDetails();
          break;
        case 7:
          votingSystem.resetVotingSystem();
          break;
        case 8:
          System.out.println("Exiting...");
          scanner.close();
          return;
        default:
          System.out.println("Invalid choice! Please enter 1-8.");
      }
    }
  }

  private static void printMenu() {
    System.out.println("\n===== Voting System Menu =====");
    System.out.println("1. Register Candidate");
```

```
        System.out.println("2. Unregister Candidate");

        System.out.println("3. Display Candidates");

        System.out.println("4. Cast Vote");

        System.out.println("5. Display Results");

        System.out.println("6. Show Voter Details");

        System.out.println("7. Reset Voting System");

        System.out.println("8. Exit");

    }


    private static int getIntInput(Scanner scanner, String prompt) {

        while (true) {

            System.out.print(prompt);

            try {

                return Integer.parseInt(scanner.nextLine());

            } catch (NumberFormatException e) {

                System.out.println("Invalid input! Please enter a number.");

            }

        }

    }


    private static void registerCandidate(Scanner scanner, VotingService votingSystem) {

        System.out.print("Enter candidate name: ");

        String name = scanner.nextLine();

        votingSystem.registerCandidate(name);

    }


    private static void unregisterCandidate(Scanner scanner, VotingService
votingSystem) {
    System.out.print("Enter candidate name to unregister: ");
    String name = scanner.nextLine();
    votingSystem.unregisterCandidate(name);
  }

  private static void castVote(Scanner scanner, VotingService votingSystem) {
    System.out.print("Enter voter ID (VXXX format): ");
    String voterId = scanner.nextLine();
    System.out.print("Enter voter name: ");
    String voterName = scanner.nextLine();

    List<String> candidates = ((VotingSystem) votingSystem).getCandidateNames();
    if (candidates.isEmpty()) {
      System.out.println("No candidates available!");
      return;
    }

    System.out.println("\nAvailable Candidates:");
    for (int i = 0; i < candidates.size(); i++) {
      System.out.println((i + 1) + ". " + candidates.get(i));
    }

    int candidateNum = getIntInput(scanner, "Enter candidate number: ");
    if (candidateNum < 1 || candidateNum > candidates.size()) {
      System.out.println("Invalid candidate number!");
      return;
    }
```

```
        votingSystem.castVote(voterId, voterName, candidates.get(candidateNum - 1));
    }
}
```