# 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;
    }
```

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

class VotingSystem {
    private Set<Voter> voters;
    private Map<String, Candidate> candidates;

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

    public void registerCandidate(String name) {
        String trimmedName = name.trim();
        if (trimmedName.isEmpty()) {
            System.out.println("Invalid candidate name!");
            return;
        }
        if (candidates.containsKey(trimmedName)) {
            System.out.println("Candidate already registered!");
        } else {
            candidates.put(trimmedName, new Candidate(trimmedName));
            System.out.println("Candidate registered successfully!");
        }
    }
```

```
public List<String> getCandidateNames() {
    return new ArrayList<>(candidates.keySet());
}

public void displayCandidates() {
    List<String> candidates = getCandidateNames();
    if (candidates.isEmpty()) {
        System.out.println("No candidates registered!");
        return;
    }
    System.out.println("\nRegistered Candidates:");
    for (int i = 0; i < candidates.size(); i++) {
        System.out.println((i + 1) + ". " + candidates.get(i));
    }
}

public void castVote(String voterId, String voterName, String candidateName) {
    if (!voterId.matches("V\\d+")) {
        System.out.println("Invalid voter ID format!");
        return;
    }

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

```
    Candidate candidate = candidates.get(candidateName.trim());
    if (candidate == null) {
        System.out.println("Invalid candidate selection!");
        return;
    }

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

public void displayResults() {
    List<Candidate> candidateList = new ArrayList<>(candidates.values());
    if (candidateList.isEmpty()) {
        System.out.println("No candidates registered!");
        return;
    }

    int totalVotes = getTotalVotes();
    Collections.sort(candidateList, (c1, c2) -> c2.getVoteCount() - c1.getVoteCount());

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

```
    }

    public void displayVoterDetails() {
        if (voters.isEmpty()) {
            System.out.println("No votes cast yet!");
            return;
        }
        System.out.println("\nVoter Details (" + voters.size() + " voters):");
        System.out.printf("%-10s %-20s%n", "Voter ID", "Name");
        for (Voter voter : voters) {
            System.out.printf("%-10s %-20s%n",
                voter.getVoterId(), voter.getName());
        }
    }

    private 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);
        VotingSystem votingSystem = new VotingSystem();

        while (true) {
```

```
System.out.println("\nVoting System Menu");
System.out.println("1. Register Candidate");
System.out.println("2. Display Candidates");
System.out.println("3. Cast Vote");
System.out.println("4. Display Results");
System.out.println("5. Show Voter Details");
System.out.println("6. Exit");
System.out.print("Enter your choice: ");

int choice;
try {
   choice = Integer.parseInt(scanner.nextLine());
} catch (NumberFormatException e) {
   System.out.println("Invalid input! Please enter a number.");
   continue;
}

switch (choice) {
   case 1:
      System.out.print("Enter candidate name: ");
      votingSystem.registerCandidate(scanner.nextLine());
      break;
   case 2:
      votingSystem.displayCandidates();
      break;
   case 3:
      handleVoteCasting(scanner, votingSystem);
      break;
```

```
            case 4:
                votingSystem.displayResults();
                break;
            case 5:
                votingSystem.displayVoterDetails();
                break;
            case 6:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice! Please try again.");
        }
    }
}

    private static void handleVoteCasting(Scanner scanner, VotingSystem votingSystem)
{
    System.out.print("Enter voter ID (VXX..): ");
    String voterId = scanner.nextLine();
    System.out.print("Enter voter name: ");
    String voterName = scanner.nextLine();

    List<String> candidates = votingSystem.getCandidateNames();
    if (candidates.isEmpty()) {
        System.out.println("No candidates registered! Please register candidates first.");
        return;
    }
```

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

        System.out.print("Enter candidate number: ");
        try {
            int candidateNumber = Integer.parseInt(scanner.nextLine());
            if (candidateNumber < 1 || candidateNumber > candidates.size()) {
                System.out.println("Invalid candidate number!");
                return;
            }
            String selectedCandidate = candidates.get(candidateNumber - 1);
            votingSystem.castVote(voterId, voterName, selectedCandidate);
        } catch (NumberFormatException e) {
            System.out.println("Invalid input! Please enter a number.");
        }
    }
}
```