**Test 1: Multiple filenames can be entered for processing.**

**File:** MainTest.java
**Author:** Caleb, Garrett

**Input:**

1. CSV files for IR with differing ballot counts.
2. CSV files for CPL with differing ballot counts.
3. CSV files for PO for individual testing.

**Tests:**

1. IRProcessing can take multiple filenames at once.
2. CPLProcessing can take multiple filenames at once.
3. A single CSV PO file can be inputted.

**Output:** OK or error message indicating faulty input.

**Pass or fail:** pass

**Date:** 4/30/23

**Test 2: Result Table pop-up for IRV election**

**File:** IRRowTest.java
**Author:** Caleb

**Input:**

1. IRRow containing simple candidate and party names.
2. IRRow containing complex candidate and party names.

**Tests:**

1. Check getCandParty() returns successfully for a simple IRRow constructor.
2. Check getCandName() returns successfully for a simple IRRow constructor.
3. Check getCandParty() returns successfully for a complex IRRow constructor.
4. Check getCandName() returns successfully for a complex IRRow constructor.

**Output:** OK or error message indicating faulty object.

**Pass or fail:** pass

**Date:** 4/29/23

**Test 3: Result Table pop-up for IRV election**

**File:** IRRowTest.java
**Author:** Caleb

**Input:**

1. Instantiated and valid IRRow test object.
2. Five integers are to be added to IRRow's stat ArrayList.

**Tests:**

1. Assert the first through fifth inputs are correct. Validates that get_stats() is correctly implemented.
2. Asserts the size of get_stats() ArrayList is cocrrect. Validates that get_length() is correctly implemented.

**Output:** OK or error message indicating function failure.

**Pass or fail:** pass

**Date:** 4/29/23

**Test 4: Attempting to update a ballot that has no remaining candidates causes system failure, check if no candidates remain on a ballot before updating.**

**File:** Ballot.java**,** BallotTest.java
**Author:** Ashton

**Input:**

1. Ballot object with two candidates
2. Ballot object with one candidate
3. Ballot object with no candidates

**Tests:**

1. numRankings is properly updated after call to updateBallot()
2. updateBallot() returns true if a ballot object has remaining candidates before and after updating
3. updateBallot() returns false if a ballot object has no remaining candidates before updating
4. updateBallot() returns false if a ballot object has no remaining candidates after updating

**Output: "**Tests passed" or error message indicating false assertEquals()

**Pass or fail:** pass

**Date:** 4/30/23

**Test 5: Run Popularity Only (PO), read information from PO election file.**

**File:** POProcessing.java, POProcessingTest.java
**Author:** Ashton

**Input:**

1. PO election CSV files, POTest1.csv and POTest2.csv. Test files have different number of candidates and ballot counts.

**Tests:**

1. Tests that candidates and candidateParties ArrayLists get filled with correct candidate and party information from CSV files, setCandidatesAndParties()
2. Tests that updateBallotCounts() distributes ballots correctly, ballotCounts Array contains correct ballot counts

**Output:** "Tests passed" or error message indicating false assertEquals()

**Pass or fail:** pass

**Date:** 4/30/23

**Test 6: Run Popularity Only (PO), process a PO election.**

**File:** POProcessing.java, POProcessingTest.java
**Author:** Ashton

**Input:**

1. PO election CSV files, POTest1.csv and POTest2.csv. Test files have different number of candidates and ballot counts.

**Tests:**

1. processElection1() tests that processElection() returns correct election winner
2. processElection2() tests that processElection() returns correct election winner, candidates had correct ballot counts

**Output:** "Tests passed" or error message indicating false assertEquals()

**Pass or fail:** pass

**Date:** 4/30/23

| |
|---|
| **Test 7: Run Popularity Only (PO), display results.** |
| **File:** POProcessing.java, POProcessingTest.java<br>**Author:** Ashton |
| **Input:**<br>1. PO election CSV file POTest1.csv.<br>**Tests:**<br>1. getVotePercents(), tests that the calculated percent of votes for each candidate is correct and is displayed on screen.<br><br>**Output:** **“**Tests passed” and correct vote percentages displayed to screen, or error message indicating false assertEquals() and incorrect vote percentages displayed to screen |
| **Pass or fail:** pass |
| **Date:** 4/30/23 |

**Test 8: Multiple file names can be entered for processing, IR processing can handle multiple files.**

**File:** IRProcessing.java, IRProcessingTest.java
**Author:** Ashton

**Input:**

1. IR election csv files, IRTesting3.csv, IRTesting5.csv, IRTest6.csv. Files contain different ballot counts.

**Tests:**

1. processElectionMultipleFiles(), uses IRTesting3.csv and IRTesting5.csv. Tests that candidates and parties are set correctly, votes are distributed correctly, and the correct winner is found in multiple file IR elections, specifically with two files.
2. processElectionMultipleFiles()2, uses IRTesting3.csv and IRTesting6.csv. Tests that candidates and parties are set correctly, votes are distributed correctly, and the correct winner is found in multiple file IR elections, specifically with two files. Tests that the outcome of the election is changed to a different winner than if just IRTesting3.csv were used.
3. processElectionMultipleFiles()3, uses IRTesting3.csv, IRTesting5.csv, and IRTesting6.csv. Tests that candidates and parties are set correctly, votes are distributed correctly, and the correct winner is found in multiple file IR elections, specifically with three files.

**Output:** **“**Tests passed” or error message indicating false assertEquals()

**Pass or fail:** pass

**Date:** 4/30/23

**Test 9: Multiple file names can be entered for processing, CPL processing can handle multiple files.**

**File:** CPLProcessing.java, CPLProcessingTest.java
**Author:** Elias and Garrett

**Input:**

1. CPL election csv files (CPLTesting1.csv, CPLTesting2.csv, CPLTesting3.csv, CPLTesting4.csv). Files contain different ballot counts.

**Tests:**

1. distributeSeats(), uses all 4 CPL CSV files to test that seats are distributed correctly.
2. distributeBallots(), uses all 4 CPL CSV files to test that ballots are distributed correctly.
3. processElection(), uses all 4 CPL CSV files to test the winners and output is correct.

**Output:** "Tests passed" or error message indicating false assertEquals()

**Pass or fail:** pass

**Date:** 4/30/23

**Test 10: System fails for CPL election when more seats are allocated to a party than there are candidates**

**File:** CPLProcessing.java, CPLProcessingTest.java
**Author:** Garrett

**Input:**

1. CPL CSV file containing the majority of votes to one party

**Tests:**

1. distributeSeats(), uses CPLTesting2.csv. Distributes the seats correctly when a party has more seats allocated to them than the number of candidates in their party.

**Output:** **“**Tests passed” or error message indicating false assertEquals()

**Pass or fail:** pass

**Date:** 4/30/23

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#:** getNextCandidateAndUpdate
**Test Description:** tests the *getNextCandidate(), updateBallot(), and getNumRankings() methods in the Ballot class.*

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method getNextCandidateAndUpdate() stored in BallotTest.java, using getNextCandidate(), updateBallot(), and getNumRankings()

**Automated: yes**_X_ **no ___**

**Results: Pass** _X___ **Fail**______

**Preconditions for Test:** ArrayList<String> cands1 = <"caleb","ashton">, Ballot blt1 = new Ballot(0,2, cands1)

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | check if getNextCandidate() returns first candidate in blt1 | blt1 | assertEquals(blt1.getNextCandidate(),"caleb") == true | assertEquals(blt1.getNextCandidate(),"caleb") == true | |
| 2 | check if updateBallot() returns true | blt1 | assertEquals(blt1.updateBallot(), true) == true | assertEquals(blt1.updateBallot(), true) == true | |
| 3 | check if updateBallot() removed first candidate in blt1, getNextCandidate() gets first candidate in blt1 | blt1 | assertEquals(blt1.getNextCandidate(),"ashton") == true | assertEquals(blt1.getNextCandidate(),"ashton") == true | |
| 4 | check if updateBallot() returns true | blt1 | assertEquals(blt1.updateBallot(), true) == true | assertEquals(blt1.updateBallot(), true) == true | |
| 5 | check if updateBallot() removes candidate and updates numRanking, check if getNumRankings() works | blt1 | assertEquals(blt1.getNumRankings(), 0) == true | assertEquals(blt1.getNumRankings(), 0) == true | |
| 6 | check getNextCandidate() | ArrayList<String> cands2 = <"caleb","ashton","abc","abcd 123"> Ballot blt2 = new Ballot(0,4,cands2) | assertEquals(blt2.getNextCandidate(),"caleb") == true | assertEquals(blt2.getNextCandidate(),"caleb") == true | |
| 7 | check updateBallot() | blt2 | assertEquals(blt2.updateBallot(), true) == true | assertEquals(blt2.updateBallot(), true) == true | |

| | | | | | |
|---|---|---|---|---|---|
| 8 | check updateBallot() removed candidate, check getNextCandidate() | blt2 | assertEquals(blt2.getNextCandidate(),"ashton") == true | assertEquals(blt2.getNextCandidate(),"ashton") == true | |
| 9 | check updateBallot() | blt2 | assertEquals(blt2.updateBallot(), true) == true | assertEquals(blt2.updateBallot(), true) == true | |
| 10 | check updateBallot() removed candidate, check getNextCandidate() | blt2 | assertEquals(blt2.getNextCandidate(),"abc") == true | assertEquals(blt2.getNextCandidate(),"abc") == true | |
| 11 | check updateBallot() | blt2 | assertEquals(blt2.updateBallot(), true) == true | assertEquals(blt2.updateBallot(), true) == true | |
| 12 | check updateBallot() removed candidate, check getNextCandidate() | blt2 | assertEquals(blt2.getNextCandidate(),"abcd 123") == true | assertEquals(blt2.getNextCandidate(),"abcd 123") == true | |

**Post condition(s) for Test:** blt1 has no remaining candidates, blt1.numRankings==0, blt2.numRankings==2

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#:** getIndex

**Test Description:** tests the getIndex() method in the Ballot class and that Ballots are created with ballotIndex set correctly

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method getIndex() stored in BallotTest.java, using getIndex()

**Automated: yes** X **no** ___

**Results: Pass** X **Fail**

**Preconditions for Test:** ArrayList<String> cands = <>, Ballot blt1 = new Ballot(0, 2, cands), Ballot blt2 = new Ballot(1, 2, cands), Ballot blt3 = new Ballot(2, 2, cands), Ballot blt4 = new Ballot(3, 2, cands), Ballot blt5 = new Ballot(999, 2, cands).

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | check blt1 ballotIndex | blt1 | *assertEquals*(blt1.getIndex(), 0) == true | *assertEquals*(blt1.getIndex(), 0) == true | |
| 2 | check blt2 ballotIndex | blt2 | *assertEquals*(blt2.getIndex(), 1) == true | *assertEquals*(blt2.getIndex(), 1) == true | |
| 3 | check blt3 ballotIndex | blt3 | *assertEquals*(blt3.getIndex(), 2) == true | *assertEquals*(blt3.getIndex(), 2) == true | |
| 4 | check blt4 ballotIndex | blt4 | *assertEquals*(blt4.getIndex(), 3) == true | *assertEquals*(blt4.getIndex(), 3) == true | |
| 5 | check blt5 ballotIndex | blt5 | *assertEquals*(blt5.getIndex(), 999) == true | *assertEquals*(blt5.getIndex(), 999) == true | |

**Post condition(s) for Test:** all ballots have the same ballotIndex as they were instantiated with

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#:** candidateTypeChecking1

**Test Description:** Tests if Candidate constructor properly creates Candidate objects and sets variables correctly

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method candidateTypeChecking() stored in CandidateTest.java, using getParty() and getCandidateName()

**Automated: yes_X_ no ___**

**Results:** **Pass** _X___ **Fail**_________

**Preconditions for Test:** Candidate cand1 = new Candidate("party name", "joe biden")

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | check if getParty() returns correct value | cand1 | *assertEquals*(cand1.getParty(),"party name") == true | *assertEquals*(cand1.getParty(),"party name") == true | |
| 2 | check if getCandidateName() returns correct value | cand1 | *assertEquals*(cand1.getCandidateName(),"joe biden") == true | *assertEquals*(cand1.getCandidateName(),"joe biden") == true | |

**Post condition(s) for Test:** all Candidates have the partyName and candidateName they were instantiated with

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#:** candidateTypeChecking2

**Name(s) of Testers:** Caleb and Ashton

**Test Description:** Tests if Candidate constructor properly creates Candidate objects and sets variables correctly

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method candidateTypeChecking() stored in CandidateTest.java, using getParty() and getCandidateName()

**Automated: yes** X **no**

**Results:** **Pass** _X___ **Fail**_________

**Preconditions for Test:** Candidate cand2 = new Candidate("a123&", "a123&");

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | check if getParty() returns correct value | cand2 | *assertEquals*(cand2.getParty(),"a123&") == true | *assertEquals*(cand2.getParty(),"a123&") == true | |
| 2 | check if getCandidateName() returns correct value | cand2 | *assertEquals*(cand2.getCandidateName(),"a123&") == true | *assertEquals*(cand2.getCandidateName(),"a123&") == true | |

**Post condition(s) for Test:** all Candidates have the partyName and candidateName they were instantiated with

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#:** addRemoveBallot

**Name(s) of Testers:** Caleb and Ashton

**Test Description:** tests the addBallot() and removeBallot() methods in the Candidate class

**Automated: yes** X **no**

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method addRemoveBallot() stored in CandidateTest.java, using addBallot() and, removeBallot()

**Results: Pass** _X___ **Fail**________

**Preconditions for Test:** ArrayList<String> cands1 = <"caleb","ashton","garrett","elias">, Ballot blt1 = new Ballot(0, 4, cands1), ArrayList<String> cands2 = <>, Ballot blt2 = new Ballot(1, 0, cands2), ArrayList<String> cands3 = <"caleb">, Ballot blt3 = new Ballot(2, 1, cands3), Candidate cand1 = new Candidate("party name", "cand name")

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | cand1.addBallot(blt1) performed, then check if getBallotCount() gets correct value | cand1, blt1 | assertEquals(cand1.getBallotCount(), 1) == true | assertEquals(cand1.getBallotCount(), 1) == true | only one ballot with candidate |
| 2 | cand1.removeBallot(0) performed, then check if getBallotCount() gets correct value | cand1 | assertEquals(cand1.getBallotCount(), 0) == true | assertEquals(cand1.getBallotCount(), 0) == true | ballot removed, now zero ballots |
| 3 | cand1.addBallot(blt1), cand1.addBallot(blt2), cand1.addBallot(blt3), cand1.addBallot(blt1), performed, and the check if getBallotCount() gets correct value | cand1, blt1,blt2.blt3 | assertEquals(cand1.getBallotCount(), 4) == true | assertEquals(cand1.getBallotCount(), 4) == true | four ballots added to candidate, now four ballots |
| 4 | cand1.removeBallot(3), cand1.removeBallot(2), performed then check if getBallotCount() gets correct value | cand1 | assertEquals(cand1.getBallotCount(), 2) == true | assertEquals(cand1.getBallotCount(), 2) == true | two ballots removed from candidate, now two ballots |
| 5 | cand1.removeBallot(1), cand1.removeBallot(0), performed, then check if getBallotCount() gets correct value | | assertEquals(cand1.getBallotCount(), 0) == true | assertEquals(cand1.getBallotCount(), 0) == true | two ballots removed from candidate, now zero ballots |

**Post condition(s) for Test:** cand1 has no ballots, cand1.getBallotCount()==0

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage:** **Unit** X **System** __

**Test Date:** 3/25/23

**Test Case ID#:** getBallotCount

**Test Description:** tests the getBallotCount() method in the Candidate class

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**

Method getBallotCount() stored in CandidateTest.java, using getBallotCount()

**Automated:** **yes** X **no**

**Results:** **Pass** X **Fail**

**Preconditions for Test:** Candidate countCand = new Candidate("party name", "cand name"), ArrayList<String> cands1 = <"caleb","ashton","garrett","elias">, Ballot blt1 = new Ballot(0, 4, cands1), ArrayList<String> cands2 = <"caleb","ashton">, Ballot blt2 = new Ballot(1, 0, cands2), ArrayList<String> cands3 = <"caleb">, Ballot blt3 = new Ballot(2, 1, cands3)

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | countCand.addBallot(blt1), countCand.addBallot(blt2), countCand.addBallot(blt3) performed, check getBallotCount() | countCand, blt1,blt2.blt3 | assertEquals(countCand.getBallotCount(), 3) == true | assertEquals(countCand.getBallotCount(), 3) == true | three ballots added, count should be three |
| 2 | countCand.addBallot(blt1), countCand.addBallot(blt2) performed, check getBallotCount() | countCand, blt1, blt2 | assertEquals(countCand.getBallotCount(), 5) == true | assertEquals(countCand.getBallotCount(), 5) == true | two more added |
| 3 | countCand.removeBallot(0), countCand.removeBallot(0) performed, check getBallotCount() | countCand | assertEquals(countCand.getBallotCount(), 3) == true | assertEquals(countCand.getBallotCount(), 3) == true | removed two |
| 4 | countCand.removeBallot(0), countCand.removeBallot(0) performed, check getBallotCount() | countCand | assertEquals(countCand.getBallotCount(), 1) == true | assertEquals(countCand.getBallotCount(), 1) == true | remove one |
| 5 | countCand.removeBallot(0) performed, check getBallotCount() | countCand | assertEquals(countCand.getBallotCount(), 0) == true | assertEquals(countCand.getBallotCount(), 0) == true | last b |

**Post condition(s) for Test:** cand1 has no ballots, cand1.getBallotCount()==0

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System** __

**Test Date:** 3/25/23

**Test Case ID#:** getParty()
**Test Description:** tests the getParty() method in the Candidate class

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method getParty() stored in CandidateTest.java, using getParty()

**Automated: yes_X_ no ___**

**Results: Pass _X___ Fail_________**

**Preconditions for Test:** Candidate cand1 = new Candidate("party name", "cand name"), Candidate cand2 = new Candidate("greenParty", "cand name"), Candidate cand3 = new Candidate("G", "cand name"), Candidate cand4 = new Candidate("123450", "cand name")

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | check cand1 party | cand1 | assertEquals(cand1.getParty(), "party name") == true | assertEquals(cand1.getParty(), "party name") == true | |
| 2 | check cand2 party | cand2 | assertEquals(cand2.getParty(), "greenParty") == true | assertEquals(cand2.getParty(), "greenParty") == true | |
| 3 | check cand3 party | cand3 | assertEquals(cand3.getParty(), "G") == true | assertEquals(cand3.getParty(), "G") == true | |
| 4 | check cand4 party | cand4 | assertEquals(cand4.getParty(), "123450")== true | assertEquals(cand4.getParty(), "123450")== true | |

**Post condition(s) for Test:** four candidates created with correct party names

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System** __

**Test Date:** 3/25/23

**Test Case ID#:** getBallots()

**Test Description:** tests the getBallots() method in the Candidate class

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method getBallots() stored in CandidateTest.java, using getBallots()

**Automated: yes** _X_ **no** ___

**Results: Pass** X **Fail**

**Preconditions for Test:** Candidate getBallotsCand = new Candidate("party name", "cand name"), ArrayList<String> cands1 = <"caleb", "ashton", "garrett", "elias">, Ballot blt1 = new Ballot(0, 4, cands1), ArrayList<String> cands2 = <"ashton","caleb">**,** Ballot blt2 = new Ballot(1, 0, cands2), ArrayList<String> cands3 = <"garrett">, Ballot blt3 = new Ballot(2, 1, cands3)

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | getBallotsCand.addBallot(blt1) getBallotsCand.addBallot(blt2) getBallotsCand.addBallot(blt3) ArrayList<Ballot> ballots = getBallotsCand.getBallots() performed, check ballots indexes | getBallotsCand, blt1, blt2, blt3 | assertEquals(ballots.get(0).getIndex(), 0) == true | assertEquals(ballots.get(0).getIndex(), 0) == true | |
| 2 | check ballots indexes | ballots | assertEquals(ballots.get(1).getIndex(), 1) == true | assertEquals(ballots.get(1).getIndex(), 1) == true | |
| 3 | check ballots indexes | ballots | assertEquals(ballots.get(2).getIndex(), 2) == true | assertEquals(ballots.get(2).getIndex(), 2) == true | |
| 4 | check ballots candidates | ballots | assertEquals(ballots.get(0).getNextCandidate(), "caleb") == true | assertEquals(ballots.get(0).getNextCandidate(), "caleb") == true | |

| | | | | | |
|---|---|---|---|---|---|
| 5 | check ballots candidates | ballots | assertEquals(ballots.get(1).getNextCandidate(), "ashton") == true | assertEquals(ballots.get(1).getNextCandidate(), "ashton") == true | |
| 6 | check ballots candidates | ballots | assertEquals(ballots.get(2).getNextCandidate(), "garrett") == true | assertEquals(ballots.get(2).getNextCandidate(), "garrett") == true | |

**Post condition(s) for Test:** candidate with three ballots created with "caleb", "ashton", and "garrett" as first choice on each ballot

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#:** getCandidateName()
**Test Description:** tests the getCandidateName() method in the Candidate class

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method getCandidateName() stored in CandidateTest.java, using getCandidateName()

**Automated: yes** X **no**

**Results: Pass** X **Fail**

**Preconditions for Test:** Candidate cand1 = new Candidate("party name", "cand name"), Candidate cand2 = new Candidate("party name", "liberals"), Candidate cand3 = new Candidate("party name", "megaLiberals"), Candidate cand4 = new Candidate("party name", "**_LIBerAls_**64")

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | check candidate name | cand1 | assertEquals(cand1.getCandidateName(), "cand name") == true | assertEquals(cand1.getCandidateName(), "cand name") == true | |
| 2 | check candidate name | cand2 | assertEquals(cand2.getCandidateName(), "liberals") == true | assertEquals(cand2.getCandidateName(), "liberals") == true | |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 3 | check candidate name | cand3 | assertEquals(cand3.getCandidateName(), "megaLiberals") == true | assertEquals(cand3.getCandidateName(), "megaLiberals") == true |  |
| 4 | check candidate name | cand4 | assertEquals(cand4.getCandidateName(), "**_LIBerAls_**64") == true | assertEquals(cand4.getCandidateName(), "**_LIBerAls_**64") == true |  |

**Post condition(s) for Test:** four candidates created, getCandidateName() successfully retrieved each candidate's name

---

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit __ System _X_**

**Test Date:** 3/25/23

**Test Case ID#:** prcoessElection1

**Name(s) of Testers:** Caleb and Ashton

**Test Description:** tests the processElection(), getCandidates(), setCandidates(), and getCandidatesArray() methods in the IRProcessing class

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method processElection1() stored in IRProcessingTest.java and IRTesting1.csv in test, using processElection(), getCandidates(), getCandidatesArray(), getCandidateName(), and getBallotCount()

**Automated: yes_X_ no ___**

**Results: Pass _X___ Fail_________**

**Preconditions for Test:** FileReader csvFile = new FileReader("src/test/java/IRTesting1.csv"), BufferedReader br = new BufferedReader(csvFile), br.readLine() doesn't throw IO exception or test will throw IO exception instead of testing processElection(), IRProcessing election = new IRProcessing(br);

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | String[] cands = election.getCandidates() | IRTesting1.csv, br, election, cands | assertEquals("Rosen", cands[0]) == true | assertEquals("Rosen", cands[0]) == true |  |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | performed then check values in cands |  |  |  |  |
| 2 | check values in cands | cands | assertEquals("Kleinberg", cands[1]) == true | assertEquals("Kleinberg", cands[1]) == true |  |
| 3 | check values in cands | cands | assertEquals("Chou", cands[2])== true | assertEquals("Chou", cands[2])== true |  |
| 4 | check values in cands | cands | assertEquals("Royce", cands[3]) == true | assertEquals("Royce", cands[3]) == true |  |
| 5 | ArrayList<Candidate> curCandsArray = election.getCandidateArray() performed then check values in curCandsArray | election, curCandsArray | assertEquals(curCandsArray.get(0).getCandidateName(), "Rosen") == true | assertEquals(curCandsArray.get(0).getCandidateName(), "Rosen") == true |  |
| 6 | check values in curCandsArray | curCandsArray | assertEquals(curCandsArray.get(2).getCandidateName(), "Chou") == true | assertEquals(curCandsArray.get(2).getCandidateName(), "Chou") == true |  |
| 7 | check curCandsArray entries BallotCount, this tests that setCandidates() works | curCandsArray | assertEquals(curCandsArray.get(0).getBallotCount(), 5) == true | assertEquals(curCandsArray.get(0).getBallotCount(), 5) == true | setCandidates() gets tested through this step |
| 8 | check curCandsArray entries BallotCount, this tests that setCandidates() works | curCandsArray | assertEquals(curCandsArray.get(2).getBallotCount(), 1) == true | assertEquals(curCandsArray.get(2).getBallotCount(), 1) == true | setCandidates() gets tested through this step |
| 9 | check if processElection() returns correct winner | election | assertEquals("Rosen", election.processElection()) == true | assertEquals("Rosen", election.processElection()) == true |  |

**Post condition(s) for Test:** processElection() returned “Rosen”, the expected winner

**Test Stage:** **Unit** __ **System** _X_

**Test Date:** 3/25/23

**Test Case ID#:** prcoessElection2

**Name(s) of Testers:** Caleb and Ashton

**Test Description:** tests the processElection(), setCandidates(), getCandidates(), and getCandidatesArray() methods in the IRProcessing class. Uses an election that contains a tie in the first round, but the final winner is unaffected by this tie.

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method processElection2() stored in IRProcessingTest.java and IRTesting2.csv in test, using processElection(), getCandidates(), getCandidatesArray(), getCandidateName(), and getBallotCount()

**Automated:** **yes** X **no**

**Results:** **Pass** _X___ **Fail**________

**Preconditions for Test:** FileReader csvFile = new FileReader("src/test/java/IRTesting2.csv"), BufferedReader br = new BufferedReader(csvFile), br.readLine() doesn't throw IO exception or test will throw IO exception instead of testing processElection(), IRProcessing election = new IRProcessing(br);

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | String[] cands = election.getCandidates() performed then check values in cands | IRTesting2.csv, br, election, cands | assertEquals("Rosen", cands[0]) == true | assertEquals("Rosen", cands[0]) == true | |
| 2 | check values in cands | cands | assertEquals("Kleinberg", cands[1]) == true | assertEquals("Kleinberg", cands[1]) == true | |
| 3 | ArrayList<Candidate> curCandsArray = election.getCandidateArray() performed then check values in curCandsArray | cands | assertEquals("Rosen", curCandsArray.get(0).getCandidateName()) == true | assertEquals("Rosen", curCandsArray.get(0).getCandidateName()) == true | |
| 4 | check values in curCandsArray | cands | assertEquals("Kleinberg", curCandsArray.get(1).getCandidateName()) == true | assertEquals("Kleinberg", curCandsArray.get(1).getCandidateName()) == true | |
| 5 | check curCandsArray entries BallotCount, this tests that setCandidates() works | election, curCandsArray | assertEquals(3, curCandsArray.get(0).getBallotCount()) == true | assertEquals(3, curCandsArray.get(0).getBallotCount()) == true | |
| 6 | check curCandsArray entries BallotCount, this tests that | curCandsArray | assertEquals(4, curCandsArray.get(1).getBallotCo | assertEquals(4, curCandsArray.get(1).getBallotCount()) | |

| | | | | | |
|---|---|---|---|---|---|
| | setCandidates() works | | unt()) == true | == true | |
| 7 | check if processElection() returns correct winner | election | assertEquals("Kleinberg", election.processElection()) == true | assertEquals("Kleinberg", election.processElection()) == true | the tie in the first round between royce and joe can go either way but klein will still win |

**Post condition(s) for Test:** processElection() returned "Kleinberg", the expected winner

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#:** determineLoser

**Test Description:** tests the determineLoser() method in IRProcessing

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method determineLoser() stored in IRProcessingTest.java and IRTesting4.csv in test, using determineLoser()

**Automated: yes**_X_ **no ___**

**Results: Pass** _X___ **Fail________**

**Preconditions for Test:** FileReader csvFile = new FileReader("src/test/java/IRTesting4.csv"), BufferedReader br = new BufferedReader(csvFile), br.readLine() doesn't throw IO exception or test will throw IO exception instead of testing determineLoser()), IRProcessing election = new IRProcessing(br);

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | Candidate loser; loser = election.determineLoser() performed and check loser name | IRTestting4.csv, election, loser | assertEquals(loser.getCandidateName(),"caleb") == true | assertEquals(loser.getCandidateName(),"caleb") == true | |
| 2 | election.redistributeBallots(lose | election, loser | assertEquals(loser.getCandidateNa | assertEquals(loser.getCandidateName(),"as | |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | r),<br>loser =<br>election.determineLoser()<br>performed and check loser<br>name |  | me(),"ashton") == true | hton") == true |  |

**Post condition(s) for Test:** IRProcessing object created, “caleb” is first loser, “ashton” is second loser

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#:** redistributeBallots
**Test Description:** tests the redistributeBallots() method in IRProcessing

**Name(s) of Testers:** Caleb and Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method redistributeBallots() stored in IRProcessingTest.java and IRTesting3.csv in test, using redistributeBallots()

**Automated: yes** X **no**

**Results: Pass** _X___ **Fail**_________

**Preconditions for Test:** FileReader csvFile = new FileReader("src/test/java/IRTesting3.csv"), BufferedReader br = new BufferedReader(csvFile), br.readLine() doesn’t throw IO exception or test will throw IO exception instead of testing determineLoser()), IRProcessing election = new IRProcessing(br);

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | ArrayList<Candidate> cands1 = election.getCandidateArray() performed and check cands1 candidates ballot counts | cands1, election | assertEquals(cands1.get(0).getBallotCount(), 5) == true | assertEquals(cands1.get(0).getBallotCount(), 5) == true |  |

| | | | | | |
|---|---|---|---|---|---|
| 2 | check cands1 candidates ballot counts | cands1, election | assertEquals(cands1.get(1).getBallotCount(), 4) == true | assertEquals(cands1.get(1).getBallotCount(), 4) == true | |
| 3 | election.redistributeBallots(election.determineLoser()), ArrayList<Candidate> cands2 = election.getCandidateArray() performed and check cands2 candidates ballot count | cands2, election | assertEquals(cands2.get(0).getBallotCount(), 9) == true | assertEquals(cands2.get(0).getBallotCount(), 9) == true | redistributeBallots here then check new ballot counts |

**Post condition(s) for Test:** IRProcessing object created, after redistribution cands2.get(0) receives 5 ballots

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#: addCandidateCPL**
**Test Description:** tests the addCandidate() method in the Party class

**Name(s) of Testers:** Garrett

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method addCandidate() stored in PartyTest.java, using addCandidate() and getCandidates()

**Automated: yes_X_ no ___**

**Results: Pass _X___ Fail_________**

**Preconditions for Test:** ArrayList<String> candidates = <"candidate1>, Party party = new Party("party", candidates);

| Step | Test Step | Test | Expected | Actual | |
|---|---|---|---|---|---|

| # | Description | Data | Result | Result | Notes |
|---|---|---|---|---|---|
| 1 | party.addCandidate("CANDIDATE2"), party.addCandidate("cAnDiDaTe3"), party.addCandidate("CANDIdate4"), party.addCandidate("candiDATE5"), performed then check party.getCandidates() values | candidates, party | assertEquals(party.getCandidates().get(0), "candidate1") == true | assertEquals(party.getCandidates().get(0), "candidate1") == true | |
| 2 | check party.getCandidates() values | candidates, party | assertEquals(party.getCandidates().get(1), "CANDIDATE2") == true | assertEquals(party.getCandidates().get(1), "CANDIDATE2") == true | |
| 3 | check party.getCandidates() values | candidates, party | assertEquals(party.getCandidates().get(2), "cAnDiDaTe3") == true | assertEquals(party.getCandidates().get(2), "cAnDiDaTe3") == true | |
| 4 | check party.getCandidates() values | candidates, party | assertEquals(party.getCandidates().get(3), "CANDIdate4") == true | assertEquals(party.getCandidates().get(3), "CANDIdate4") == true | |
| 5 | check party.getCandidates() values | candidates, party | assertEquals(party.getCandidates().get(4), "candiDATE5") == true | assertEquals(party.getCandidates().get(4), "candiDATE5") == true | |

**Post condition(s) for Test:** party has five candidates

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System** __

**Test Date:** 3/25/23

**Test Case ID#: removeCandidateCPL**
**Test Description:** tests the removeCandidate() method in the Party class

**Name(s) of Testers:** Ashton

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method removeCandidate() stored in PartyTest.java, using addCandidate(),

**Automated: yes** X **no**

removeCandidate(), and getCandidates()

**Results: Pass** X **Fail**

**Preconditions for Test:** ArrayList<String> candidates = <"candidate1>, Party party = new Party("party", candidates);

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | party.addCandidate("CANDIDATE2"), party.addCandidate("cAnDiDaTe3"), party.removeCandidate("CANDIDATE2") performed then check party.getCandidates() values | candidates, party | assertEquals(party.getCandidates().get(1), "cAnDiDaTe3") == true | assertEquals(party.getCandidates().get(1), "cAnDiDaTe3") == true | |
| 2 | party.removeCandidate("candidate1") performed then check party.getCandidates() values | candidates, party | assertEquals(party.getCandidates().get(0), "cAnDiDaTe3") == true | assertEquals(party.getCandidates().get(0), "cAnDiDaTe3") == true | |
| 3 | party.removeCandidate("cAnDiDaTe3") performed then check party.getCandidates() size | candidates, party | assertEquals(party.getCandidates().size(), 0) == true | assertEquals(party.getCandidates().size(), 0) == true | removed all candidates so size of ArrayList from getCandidates() should be zero |

**Post condition(s) for Test:** party has zero candidates left

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System __**

**Test Date:** 3/25/23

**Test Case ID#: getPartyCPL**

**Test Description:** tests the getParty() method in the Party class

**Name(s) of Testers:** Garrett

**Automated: yes** _X_ **no ___**

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**

Method getParty() stored in PartyTest.java, using getParty()

**Results: Pass** X **Fail**

**Preconditions for Test:** ArrayList<String> candidates = <”candidate1>

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | Party party1 = new Party("Republican", candidates), Party party2 = new Party("democratic", candidates), Party party3 = new Party("iNdEpEnDeNt", candidates), Party party4 = new Party("REAList", candidates) performed, then check party1.getParty() | candidates, party1 | assertEquals(party1.getParty(), "Republican") == true | assertEquals(party1.getParty(), "Republican") == true | |
| 2 | check party2.getParty() | party2 | assertEquals(party2.getParty(), "democratic") == true | assertEquals(party2.getParty(), "democratic") == true | |
| 3 | check party3.getParty() | party3 | assertEquals(party3.getParty(), "iNdEpEnDeNt")== true | assertEquals(party3.getParty(), "iNdEpEnDeNt")== true | |
| | check party4.getParty() | party4 | assertEquals(party4.getParty(), "REAList") == true | assertEquals(party4.getParty(), "REAList") == true | |

**Post condition(s) for Test:** four parties were created and getParty() correctly identified all four party names

**Test Stage:** **Unit** X **System** __

**Test Date:** 3/25/23

**Test Case ID#:** **getPartiesCPL**

**Test Description:** tests the getParties() method in the CPLProcessing class using CPLTesting.csv

**Name(s) of Testers:** Garrett

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method getParties() stored in CPLProcessing.java and CPLTesting.csv, using getParties()

**Automated:** **yes** _X_ **no** ___

**Results:** **Pass** _X___ **Fail**________

**Preconditions for Test:**

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | String[] parties = election.getParties() performed, then check parties values | CPLTesting.csv, election, parties | assertEquals("Democratic", parties[0]) == true | assertEquals("Democratic", parties[0]) == true | |
| 2 | check parties values | parties | assertEquals("Republican", parties[1]) == true | assertEquals("Republican", parties[1]) == true | |
| 3 | check parties values | parties | assertEquals("New Wave", parties[2]) == true | assertEquals("New Wave", parties[2]) == true | |
| 4 | check parties values | parties | assertEquals("Reform", parties[3]) == true | assertEquals("Reform", parties[3]) == true | |
| 5 | check parties values | parties | assertEquals("Green", parties[4]) == true | assertEquals("Green", parties[4]) == true | |
| 6 | cheack parties values | parties | assertEquals("Independent", parties[5])== true | assertEquals("Independent", parties[5])== true | |

**Post condition(s) for Test:** parties contains all the parties in the election

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit** X **System** __

**Test Date:** 3/25/23

**Test Case ID#: getCandidatesCPL**

**Name(s) of Testers:** Garrett

**Test Description:** tests the getCandidates() method in the CPLProcessing class using CPLTesting.csv

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Method getCandidates() stored in CPLProcessing.java and CPLTesting.csv, using getCandidates()

**Automated: yes**_X_ **no** ___

**Results: Pass** _X___ **Fail**_________

**Preconditions for Test:** FileReader csvFile = new FileReader("src/test/java/CPLTesting.csv"), BufferedReader br = new BufferedReader(csvFile), br.readLine() doesn't throw an IO exception, CPLProcessing election = new CPLProcessing(br);

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | String[] candidates = election.getCandidates() performed then check candidates values | election, candidates | assertEquals("Foster", candidates[0]) == true | assertEquals("Foster", candidates[0]) == true | |
| 2 | check candidates values | candidates | assertEquals("Volz", candidates[1]) == true | assertEquals("Volz", candidates[1]) == true | |
| 3 | check candidates values | candidates | assertEquals("Pike", candidates[2]) == true | assertEquals("Pike", candidates[2]) == true | |
| 4 | check candidates values | candidates | assertEquals("Green", candidates[3]) == true | assertEquals("Green", candidates[3]) == true | |

| | | | | | |
|---|---|---|---|---|---|
| 5 | check candidates values | candidates | assertEquals("Xu", candidates[4])== true | assertEquals("Xu", candidates[4])== true | |
| 6 | check candidates values | candidates | assertEquals("Wang", candidates[5])== true | assertEquals("Wang", candidates[5])== true | |
| 7 | check candidates values | candidates | assertEquals("Jacks", candidates[6]) == true | assertEquals("Jacks", candidates[6]) == true | |
| 8 | check candidates values | candidates | assertEquals("Rosen", candidates[7]) == true | assertEquals("Rosen", candidates[7]) == true | |
| 9 | check candidates values | candidates | assertEquals("McClure", candidates[8]) == true | assertEquals("McClure", candidates[8]) == true | |
| 10 | check candidates values | candidates | assertEquals("Berg", candidates[9])== true | assertEquals("Berg", candidates[9])== true | |
| 11 | check candidates values | candidates | assertEquals("Zheng", candidates[10])== true | assertEquals("Zheng", candidates[10])== true | |
| 12 | check candidates values | candidates | assertEquals("Melvin", candidates[11])== true | assertEquals("Melvin", candidates[11])== true | |
| 13 | check candidates values | candidates | assertEquals("Peters", candidates[12]) == true | assertEquals("Peters", candidates[12]) == true | |

**Post condition(s) for Test:** candidates contains all the candidates in the election

**Project Name: Project 1: Voting System** **Team#** 25

**Test Stage: Unit __ System _X_** **Test Date:** 3/25/23

**Test Case ID#: processElectionCPL** **Name(s) of Testers:** Garrett

**Test Description:** tests the processElection() method in the CPLProcessing class using CPLTesting.csv

|  |  |
|---|---|
| **Automated: yes** _X__ **no** ___ | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.** Method processElection() stored in CPLProcessing.java and CPLTesting.csv, using processElection() |
| **Results: Pass** _X___ **Fail**_________ |  |

**Preconditions for Test:** FileReader csvFile = new FileReader("src/test/java/CPLTesting.csv"), BufferedReader br = new BufferedReader(csvFile), br.readLine() doesn't throw an IO exception, CPLProcessing election = new CPLProcessing(br);

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | check that processElection() returns the correct winners | CPLTesting.csv, election | assertEquals("Foster,Green,McClure,", election.processElection()) == true | assertEquals("Foster,Green,McClure,", election.processElection()) == true |  |

**Post condition(s) for Test:** processElection() returned "Foster,Green,McClure,", the correct election winners

**Project Name: Project 1: Voting System** **Team#** 25

|  |  |
|---|---|
| **Test Stage: Unit** X **System __** | **Test Date:** 3/25/23 |
| **Test Case ID#: getSetBallotCountCPL** **Test Description:** Tests the getBallotCount() method and setballotCount() methods in the party class. | **Name(s) of Testers:** Garrett |
| **Automated: yes** X **no** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.** Test stored in PartyTest.java. Methods being used are getBallotCount() and setBallotCount() stored in Party.java, |
| **Results: Pass** _X___ **Fail**_________ |  |

**Preconditions for Test:** ArrayList<String> candidates = <”candidate1>

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | Create a candidates ArrayList and create a Party object (party) with the candidate ArrayList. Call setballotCount(15) and call getBallotCount() | candidates, party | assertEquals(party.getBallotCount(), 15) == true | assertEquals(party.getBallotCount(), 15) == true | |
| 2 | Call setballotCount(53) and call getBallotCount() | candidates, party | assertEquals(party.getBallotCount(), 53) == true | assertEquals(party.getBallotCount(), 53) == true | |
| 3 | Call setballotCount(1) and call getBallotCount() | candidates, party | assertEquals(party.getBallotCount(), 1) == true | assertEquals(party.getBallotCount(), 1) == true | |

**Post condition(s) for Test:** setBallotCount() correctly sets the ballot count of a party and getBallotCount() successfully retrieves the ballot count.

**Project Name: Project 1: Voting System** **Team#25**

**Test Stage:** **Unit:** X **System:**

**Test Date:** **3-26-23**

**Test Case ID#: getSetNumSeatsCPL**

**Name(s) of Testers:** Elias

**Test Description:** Tests setNumSeats() and getNumSeats()

in the party class

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Test stored in PartyTest.java. The methods being used are getNumSeats() and setNumSeats() that are in Party.java.

**Automated: yes X no**

**Results: Pass _X____ Fail________**

**Preconditions for Test:** ArrayList<String> candidates = <!null>

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | Create an ArrayList of candidates. A Party object is created and fed the ArrayList of candidates as that party's candidates. Call party.setNumSeats(3) and party.getNumSeats(). | candidates, party | assertEquals(party.getNumSeats(), 3)==true | assertEquals(party.getNumSeats(), 3)==true | |
| 2 | Call party.setNumSeats(6) and party.getNumSeats(). | candidates, party | assertEquals(party.getNumSeats(), 6) == true | assertEquals(party.getNumSeats(), 6)== true | |
| 3 | Call party.setNumSeats(0) and party.getNumSeats(). | candidates, party | assertEquals(party.getNumSeats(), 0)== true | assertEquals(party.getNumSeats(), 0) == true | |
| 4 | Call party.setNumSeats(10) and getNumSeats() | candidates, party | assertFalse(party.getNumSeats() == -1) = false | assertFalse(party.getNumSeats() == -1) = false | |
| 5 | Call party.setNumSeats(10) and getNumSeats() | candidates, party | assertTrue(party.getNumSeats() == 10) == true | assertTrue(party.getNumSeats() == 10) == true | |
| **Step #** | **Test Step Description** | **Test Data** | **Expected Result** | **Actual Result** | **Notes** |

**Post condition(s) for Test:**

setNumSeats() should be able to set the number of seats for a party object. getNumSeats() should correctly retrieve the number of seats for a party object.

**Project Name: Project 1: Voting System** **Team#25**

**Test Stage: Unit:** X **System:**

**Test Date:** **3-26-23**

**Test Case ID#: setParties()**
**Test Description:** Tests setParties() method in CPLProcessing

**Name(s) of Testers:** Ashton, Garrett

**Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
Test stored in CPLProcessingTest.java and CPLTesting.csv, uses getParty() and getCandidates()

**Automated: yes X no**

**Results: Pass _X____ Fail_________**

**Preconditions for Test:** FileReader csvFile = new FileReader("src/test/java/CPLTesting.csv"), BufferedReader br = new BufferedReader(csvFile), br.readFile() doesn't throw exception, CPLProcessing election = new CPLProcessing(br), ArrayList<Party> parts = election.getParties(), ArrayList<String> cands = new ArrayList<String>();

| Step # | Test Step Description | Test Data | Expected Result | Actual Result | Notes |
|---|---|---|---|---|---|
| 1 | cands.add("Foster"), cands.add("Volz"), cands.add("Pike") performed. check party name | CPLTesting.csv, parts | assertEquals(parts.get(0).getParty(), "Democratic") == true | assertEquals(parts.get(0).getParty(), "Democratic") == true | |
| 2 | check candidate names | cands, parts | assertEquals(parts.get(0).getCandidates, cands) == true | assertEquals(parts.get(0).getCandidates, cands) == true | |
| 3 | cands.clear(), cands.add("Green"), cands.add("Xu"), cands.add("Wang") performed, check party name | parts | assertEquals(parts.get(1).getParty(), "Republican") == true | assertEquals(parts.get(1).getParty(), "Republican") == true | |
| 4 | check candidate names | cands, parts | assertEquals(parts.get(1).getCandidates(), cands) == true | assertEquals(parts.get(1).getCandidates(), cands) == true | |
| 5 | cands.clear(), cands.add("Jacks"), cands.add("Rosen") performed, check party name | parts | assertEquals(parts.get(2).getParty(), "New Wave") == true | assertEquals(parts.get(2).getParty(), "New Wave") == true | |
| 6 | check candidate names | cands, parts | assertEquals(parts.get(2).getCandidates(), cands) == true | assertEquals(parts.get(2).getCandidates(), cands) == true | |
| 7 | cands.clear(), cands.add("McClure"), cands.add("Berg") performed, check party name | parts | assertEquals(parts.get(3).getParty(), "Reform") == true | assertEquals(parts.get(3).getParty(), "Reform") == true | |
| 8 | check candidate names | cands, parts | assertEquals(parts.get(3).getCandidates(), cands) == true | assertEquals(parts.get(3).getCandidates(), cands) == true | |
| 9 | cands.clear(), cands.add("Zheng"), cands.add("Melvin") performed, check party name | parts | assertEquals(parts.get(4).getParty(), "Green") == true | assertEquals(parts.get(4).getParty(), "Green") == true | |
| 10 | check candidate names | cands, parts | assertEquals(parts.get(4).getCandidates(), cands) == true | assertEquals(parts.get(4).getCandidates(), cands) == true | |
| 11 | cands.clear(), cands.add("Peters") performed, check party name | parts | assertEquals(parts.get(5).getParty(), "Independent") == true | assertEquals(parts.get(5).getParty(), "Independent") == true | |
| 12 | check candidate names | cands, parts | assertEquals(parts.get(5).getCandidates(), cands) == true | assertEquals(parts.get(5).getCandidates(), cands) == true | |

**Post condition(s) for Test:** CPLProcessing election created with parties and associated candidates from CPLTesting.csv