Project 2: Voting System Team# 6

TestCandidate

Test Stage: Unit X System___ **Test Date:** 3-26-21
Test Case ID#: 001 **Name(s) of Testers:** Hazel Dunn

Test Description: testAddBallot: This test will check to see if the addBallot function in the Candidate class completes. It tests to see if ballots have proper IDs, and checks to see if they can be added properly to a candidate's ballot array.
→ This test is stored in the file TestCandidate.java in the src folder and is called testAddBallot. The name of the function being tested is addBallot in the Candidate.java file.
It uses getters and setters for both candidate and ballot objects, and calls the addBallot function, then repeatedly checks the size to make sure they are added correctly.

Automated: Yes__X_ No __ **Results:** Pass X Fail___

Preconditions for Test: The candidates must be created from the OPL file, and the ballots must be read in order for them to be added. In setup, ballot objects are created and the IDs of each are set in order.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Check that the ID of ballot 1 is correct	001	001	
2	Check that the ID of ballot 2 is correct	002	002	
3	Check that the ID of ballot 3 is correct	003	003	
4	Check that the ID of ballot 1 is not 0	True	True	IDs must be assigned, cannot be zero
5	Check that the ID of ballot 2 is not 0	True	True	
6	Check that the ID of ballot 3 is not 0	True	True	
7	Create a new candidate object and add ballots 1 and 2 to the candidate ballot array	N/A	N/A	Want ballots to add to the candidate array
8	Check that the ballot at index 0 of the candidate's ballot array is correct	ballot1	ballot1	
9	Check that the ballot at index 1 of	ballot2	ballot2	

	the candidate’s ballot array is correct			
10	Check that the size of the candidate’s current ballot array is correct	2	2	
11	Add ballot4 (null ballot) to the candidate	N/A	N/A	Should not add, if there is a null ballot we don’t want it to be counted
12	Check that the ballot at index 0 of the candidate’s ballot array is correct	ballot1	ballot1	
13	Check that the ballot at index 1 of the candidate’s ballot array is correct	ballot2	ballot2	
14	Check that the size of the candidate’s current ballot array is correct	2	2	
15	Add ballot3 (non-null ballot) to the candidate	N/A	N/A	This ballot should add and alter the data
16	Check that the ballot at index 0 of the candidate’s ballot array is correct	ballot1	ballot1	
17	Check that the ballot at index 1 of the candidate’s ballot array is correct	ballot2	ballot2	
18	Check that the ballot at index 2 of the candidate’s ballot array is correct	ballot3	ballot3	
19	Check that the size of the candidate’s current ballot array is correct	3	3	

Post Condition(s) for Test: All ballots have been added to their proper candidate’s ballot array after being read in.

TestIRElection

Test Stage: Unit X System___ **Test Date:** 3-13-21

Test Case ID#: 002 **Name(s) of Testers:** Olivia Hansen

Test Description: testFindMajority() will ensure that the correct majority is found out of the currCandidates array, if there is a majority. If not, it will ensure that the function returns null.

testFindMajority() is in the TestIRElection.java file. It tests the function findMajority() from the IRElection.java file. Both of these files can be found in the src folder of the Project1 directory.

Automated: Yes_X__ No __ **Results:** Pass X Fail___

Preconditions for Test: Candidate objects are assigned a name, party, and cNumBallots. An election object is created and the currCandidates array, the eliminatedCandidates array, and the totalNumBallots are also set to the appropriate values.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	findMajority() is called on the election object	null	null	Set up so no candidate has the majority.
2	findMajority() is called on the election object, and getcName() is called on the returned candidate.	Rosen	Rosen	Set up so candidate1 has the majority.
3	findMajority() is called on the election object	null	null	Set up so there is only one candidate in the array, but they still do not have the majority
4	findMajority() is called on the election object, and getcName() is called on the returned candidate.	Rosen	Rosen	Set up so there is only one candidate in the array and they do have the majority.

Post Condition(s) for Test: The findMajority() function returns the expected values in multiple edge case scenarios.

Test Stage: Unit X System___ **Test Date:** 3-13-21
Test Case ID#: 003 **Name(s) of Testers:** Olivia Hansen

Test Description: testFindLeastCand() will ensure that the correct candidate with the least amount of votes is found out of the currCandidates array.

testFindLeastCand() is in the TestIRElection.java file. It tests the function findLeastCand() from the IRElection.java file. Both of these files can be found in the src folder of the Project1 directory.

Automated: Yes_X__ No __ **Results:** Pass X Fail___

Preconditions for Test: Candidate objects are assigned a name, party, and cNumBallots. An election object is created and the currCandidates array, the eliminatedCandidates array, and the totalNumBallots are also set to the appropriate values.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	findLeastCand() is called on the election object and getcName(0 is called on the returned candidate.	Kleinberg	Kleinber g	Set up so candidate2 has the least amount of ballots (which is zero).
2	findLeastCand() is called on the election object and getcName(0 is called on the returned candidate.	Royce	Royce	Set up so candidate4 has the least amount of ballots.
3	findLeastCand() is called on the election object and getcName(0 is called on the returned candidate.	Chou	Chou	Set up so candidate3 has the least amount of ballots.
4	findLeastCand() is called on the election object and getcName(0 is called on the returned candidate.	Rosen	Rosen	Set up so candidate1 is the only one candidate in the array, but they still technically have the least amount of ballots.
5	findLeastCand() is called on the election object and getcName(0 is called on the returned candidate.	Kleinberg || Chou	Kleinber g || Chou	Set up so candidate2 and candidate3 are tied for the least amount of votes, so the function should return one or the other.

Post Condition(s) for Test: The findLeastCand() function returns the expected values in multiple edge case scenarios.

Test Stage: Unit X System___ **Test Date:** 3-13-21
Test Case ID#: 004 **Name(s) of Testers:** Olivia Hansen

Test Description: testRedistributeBallots() will ensure that the correct candidate is determined to be the least candidate and their ballots are redistributed to the correct Candidates.

testRedistributeBallots() is in the TestIRElection.java file. It tests the function redistributeBallots() from the IRElection.java file. Both of these files can be found in the src folder of the Project1 directory.

Automated: Yes_X__ No __ **Results:** Pass X Fail___

Preconditions for Test: Candidate objects are assigned a name, party, and cNumBallots. Ballot objects are also created and their ranking arrays are set. Said ballots are added to their respective candidate's cBallots arrayList. An election object is created and the currCandidates array, the eliminatedCandidates array, and the totalNumBallots are also set to the appropriate values.

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	Checked that a candidate was added to eliminatedCandidates by calling eliminatedCandidates.size()	1	1	
2	Check that the correct candidate was added to eliminatedCandidates by calling eliminatedCandidates.get(0).getcName();	Kleinberg	Kleinberg	
3	Check that candidate was removed from the currCandidates array by calling currCandidates.size()	3	3	
4	Check that the correct candidate was removed from the currCandidates array by calling currCandidates.contains(candidate2)	false	false	
5	Checked that a candidate was added to eliminatedCandidates by calling eliminatedCandidates.size()	2	2	
6	Check that the correct candidate was added to eliminatedCandidates by calling eliminatedCandidates.get(1).getcName();	Royce	Royce	
7	Check that candidate4's ballot was not redistributed, since candidate4 was the only candidate in it's	true	true	

	ranking array. This is checked by calling candidate4.getcBallots().contains(ballot6)			
8	Check that candidate was removed from the currCandidates array by calling currCandidates.size()	2	2	
9	Check that the correct candidate was removed from the currCandidates array by calling currCandidates.contains(candidate4)	false	false	
10	Checked that a candidate was added to eliminatedCandidates by calling eliminatedCandidates.size()	3	3	
11	Check that the correct candidate was added to eliminatedCandidates by calling eliminatedCandidates.get(2).getcName();	Chou	Chou	
12	Check that one of candidate3's ballots was not redistributed, since candidate4 was the only other candidate in it's ranking array, and they have already been eliminated. This is checked by calling candidate3.getcBallots().contains(ballot5)	true	true	
13	Check that candidate was removed from the currCandidates array by calling currCandidates.size()	1	1	
14	Check that the correct candidate was removed from the currCandidates array by calling currCandidates.contains(candidate3)	false	false	
15	Check that candidate3's other ballot (ballot4) was not distributed to an eliminated candidate, even though they are the next candidate in the ranking array. This is checked by calling candidate2.getcBallots().contains(b	false	false	

	allot4)			
16	Check that ballot4 is redistributed to the correct candidate. This is checked by calling candidate1.getcBallots().contains(ballot4)	true	true	

Post Condition(s) for Test: The redistributeBallots() function sets the expected values in multiple edge case scenarios.

Test Stage: Unit__ System_X__

Test Date: 3-13-2021

Test Case ID#: 005

Name(s) of Testers: Olivia Hansen

Test Description: testRunElection() will ensure that all of the correct variables are set at the end of an IR Election.

testRunElection() is in the TestIRElection.java file. It tests the function runElection() from the IRElection.java file. Both of these files can be found in the src folder of the Project1 directory.

Automated: Yes_X__ No__

Results: Pass_X_ Fail___

Preconditions for Test: The correct input for the CSV File, the audit file, and the media files are set.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Check if the electionType is set to the correct value. This is tested by calling election.getElectionType()	IR	IR	
2	Check if the CSV name is set to the correct value. This is tested by calling election.getCsvName().	IRTest	IRTest	
3	Check if the totalNumBallots is set to the correct value. This is tested by calling election.getTotalNumBallots().	6	6	
4	Check if the correct winner is the only candidate left in the	Rosen	Rosen	

	currCandidates array. This is tested by calling election.getCurrCandidate().get(0).getcName()			
5	Check that the winner has the correct number of ballots. This is tested by calling election.getCurrCandidate().get(0).getcNumBallots()	4	4	
6	Check that all other candidates are moved to the eliminated candidates array. This is tested by calling election.getEliminatedCandidates().size()	3	3	
7	Make sure losers were removed from currCandidates. This is tested by calling election.getCurrCandidates().size()	1	1	
8	Make sure that the correct candidate was moved to eliminatedCandidates first. This is tested by calling election.getEliminatedCandidates().get(0).getcName().	Kleinberg	Kleinberg	
9	Make sure that the correct candidate was moved to eliminatedCandidates second. This is tested by calling election.getEliminatedCandidates().get(1).getcName().	Royce	Royce	
10	Make sure that the correct candidate was moved to eliminatedCandidates third. This is tested by calling election.getEliminatedCandidates().get(2).getcName().	Chou	Chou	
11	Make sure Kleinberg’s ballots were redistributed correctly. This is tested by calling election.getEliminatedCandidates().get(0).getcNumBallots()	0	0	
12	Make sure Royce’s ballots were redistributed correctly. This is tested	1	1	

	by calling election.getEliminatedCandidates(). get(1).getcNumBallots()			
13	Make sure Chou's ballots were redistributed correctly. This is tested by calling election.getEliminatedCandidates(). get(2).getcNumBallots()	1	1	

Post Condition(s) for Test: The runElection() function sets all the variables to their expected values.

Test Stage: Unit__ System_X__
Test Date: 3-13-2021
Test Case ID#: 006
Name(s) of Testers: Olivia Hansen

Test Description: testIRMajority() will ensure that the correct candidate is selected as the winner in the edge case that there is a candidate with the majority right away.

testIRMajority() is in the TestIRElection.java file.

Automated: Yes_X__ No__
Results: Pass_X_ Fail___

Preconditions for Test: The correct input for the CSV File, the audit file, and the media files are set.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Checks to see if the correct winner is found by calling getCurrCandidates.get(0).getcName()	Weasley	Weasley	
2	Checks to see if the winner had the correct number of ballots by calling getCurrCandidates.get(0).getcNumBallots()	5	5	
3	Check that the 1st candidate to be eliminated has 0 ballots	0	0	
4	Check that the 2nd candidate to be eliminated has 1 ballot	1	1	

5	Check that the 3rd candidate to be eliminated has 0 ballots	0	0	
6	Check that the 4th candidate to be eliminated has 0 ballots	0	0	

Post Condition(s) for Test: Ensures the system runs correctly in the edge case that there is a clear majority right off the bat.

Test Stage: Unit__ System_X__ **Test Date:** 3-13-2021
Test Case ID#: 007 **Name(s) of Testers:** Olivia Hansen

Test Description: testIRPopularity() will ensure that the correct candidate is selected as the winner in the edge case that there is never a majority.

testIRMajority() is in the TestIRElection.java file.

Automated: Yes_X__ No__ **Results:** Pass_X_ Fail___

Preconditions for Test: The correct input for the CSV File, the audit file, and the media files are set.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Checks to see if the correct winner is found by calling getCurrCandidates.get(0).getcName()	Shaggy	Shaggy	
2	Checks to see if the winner had the correct number of ballots by calling getCurrCandidates.get(0).getcNumBallots()	3	3	
3	Checks to see if all of the losers still have the correct number of ballots.	0	0	

Post Condition(s) for Test: Ensures the system runs correctly in the edge case that there is never a majority.

Test Stage: **Unit** **System** X **Test Date:** 3-25-2021

Test Case ID#: 008 **Name(s) of Testers:** Maranda Donaldson

Test Description: testIRPopularityTie:
This function tests that if there is a tie when determining which candidate won by popularity, the chances of being picked are equal. The test runs the election algorithm 1000 times and records which candidate won each time.

Automated **Yes** X **No**

Results **Pass** 〉**Fail**

Preconditions for Test: The file "IRTestPopularityTie.csv" is in the project directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the first candidate won between 450 and 550 times		(stylesWon > 450 && stylesWon < 550) => true	true	
2	Check that the second candidate won between 450 and 550 times		(horanWon > 450 && horanWon < 550) => true	true	
3	Check that no other candidates won		true	true	This ensures that there are no other errors when calculating a tie

Post condition(s) for Test:

Test Stage: **Unit** **System** X **Test Date:** 3-25-2021

Test Case ID#:
009 **Name(s) of Testers:** Maranda Donaldson

Test Description: testIRRedistributeTie:
This function tests that if there is a tie when determining which candidate to redistribute ballots from, the chances of being picked are equal. The test runs the election algorithm 1000 times and records which candidate was eliminated each time.

Automated **Yes** X **No**

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

Preconditions for Test: The file “IRTestRedistributeTie.csv” is in the project directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the second candidate was eliminated between 450 and 550 times		(tolkienWon > 450 && tolkienWon < 550) => true	true	The first candidate winning indicated that the second candidate was eliminated
2	Check that the first candidate was eliminated between 450 and 550 times		(rowlingWon > 450 && rowlingWon < 550) => true	true	
3	Check that no other candidates won		true	true	This ensures that there are no other errors when calculating a tie

Post condition(s) for Test: The voting algorithm will have been run 1000 times and it is expected that the variables tolkienWon and rowlingWon are set to a number around 500 and the variable error is set to 0

Test Stage: **Unit** X **System** **Test Date:** 3-25-2021

Test Case ID#:
010

Name(s) of Testers: Maranda Donaldson

Test Description: testReadIRCSVValidInput:
This function tests that the readIRCSV function correctly sets up all voting system information for ballots to be read in

Automated **Yes** X **No**

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

Preconditions for Test: The file “IRTest.csv” is in the project directory

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that readIRCSV correctly sets totalNumBallots		6	6	
2	Check that readIRCSV correctly sets the size of the Candidate array		4	4	
3	Check that readIRCSV correctly sets Candidate information		“Rosen”	“Rosen”	Only checks the first object
4	Check that readIRCSV correctly sets Party information		“D”	“D”	Only checks the first object

Post condition(s) for Test: A voting system will be created with the csvFile, csvName, and totalNumBallots variables set. The Candidate and Party arrays will also be populated

---

Test Stage: **Unit** X **System** **Test Date:** 3-25-2021

Test Case ID#:
011

Name(s) of Testers: Maranda Donaldson

Test Description: testReadIRCSVInvalidSecondLine:
This function tests that the readIRCSV function correctly deals with the case when the 2nd line (number of candidates) of the CSV file is not in the expected format

Automated **Yes** X **No**

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

Preconditions for Test: The file “IRInvTest2.csv” is in the project directory

Step	**Test Step**	**Test Data**	**Expected**	**Actual**	**Notes**

#	Description		Result	Result	
1	Check that readIRCSV does not set totalNumBallots		0	0	Check that the variable is set to 0 since this is the default in java
2	Check that readIRCSV does not set the candidate ArrayList		null	null	

Post condition(s) for Test: A voting system with only the csvFile and csvName variables set

---

Test Stage: **Unit** X **System** **Test Date:** 3-25-2021

Test Case ID#:
012 **Name(s) of Testers:** Maranda Donaldson

Test Description: testReadIRCSVInvalidThirdLine:
This function tests that the readIRCSV function correctly deals with the case when the 3rd line (candidate list) of the CSV file is not in the expected format

Automated **Yes** X **No**

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

Preconditions for Test: The file “IRInvTest3.csv” is in the project directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that readIRCSV does not set totalNumBallots		0	0	Check that the variable is set to 0 since this is the default in java
2	Check that readIRCSV does not set the candidate ArrayList		null	null	

Post condition(s) for Test: A voting system with only the csvFile and csvName variables set

---

Test Stage:	**Unit** X **System**	**Test Date:** 3-25-2021	
Test Case ID#: 013		**Name(s) of Testers:** Maranda Donaldson	

Test Description: testReadIRCSVInvalidFourthLine:
This function tests that the readIRCSV function correctly deals with the case when the 4th line (number of ballots) of the CSV file is not in the expected format

Automated **Yes** X **No**

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

Preconditions for Test: The file “IRInvTest4.csv” is in the project directory

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that readIRCSV does not set totalNumBallots		0	0	Check that the variable is set to 0 since this is the default in java
2	Check that readIRCSV correctly sets the size of the candidates ArrayList		4	4	
3	Check that readIRCSV correctly sets the size of the currCandidates ArrayList		4	4	
4	Check that readIRCSV correctly sets the size of the eliminatedCandidates ArrayList		0	0	

Post condition(s) for Test: A voting system with the csvFile, csvName, and all candidate arrayLists initialized

---

Test Stage: **Unit** X **System** **Test Date:** 3-25-2021

Test Case ID#:
014

Name(s) of Testers: Maranda Donaldson

Test Description: testReadBallotsValidInput:
This function tests that the readBallots function correctly reads and sets input ballot data

Automated **Yes** X **No**

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

Preconditions for Test: The file “IRTest.csv” is in the project directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that readBallots correctly reads the totalNumberofBallots		6	6	
2	Check that readBallots assigns the correct number of ballots		6	6	

Post condition(s) for Test:

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 015						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testReadBallotsInvalidInput: This function tests that the readBallots function correctly deals with the case when ballot data is incorrect or missing					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: The file “IRTestInvalid.csv” is in the project directory						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that readBallots correctly reads the total number of ballots supposed to be in the file		6	6	
2	Check that readBallots does not create any ballot objects		0	0	

Post condition(s) for Test: A voting system will be created with the csvFile, csvName, and totalNumBallots variables set

---

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 017					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testWriteToMediaFile: This function tests that the writeToMediaFile function correctly prints to the opened media file				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: None, all set up is done in the test function					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the output to the media file is as expected	A multi-line string representing the expected output	Strings are equal	Strings are equal	The multi-line string is compared to a string representation of the printed file

Post condition(s) for Test: A voting system with the mediaFile, currCandidates, eliminatedCandidates, and totalNumBallots variables set. Corresponding Candidate objects will have been created and assigned.

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 018						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testWriteToAuditFile: This function tests that the writeToAudit function correctly prints to the opened audit file					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, all set up is done in the test function						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the output to the audit file is as expected	A multi-line string representing the expected output	Strings are equal	Strings are equal	The multi-line string is compared to a string representation of the printed file

Post condition(s) for Test: A voting system with the auditFile, candidate, and totalNumBallots variables set. There will also be Ballot and Candidate objects that are created and assigned as necessary

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 019						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPrintToScreen: This function tests that the printToScreen function correctly prints to the screen					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			

Preconditions for Test: None, all set up is done in the test function

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the output to the screen is as expected	A multi-line string representing the expected output	Strings are equal	Strings are equal	System.out is redirected to an OutputStream so that output can be compared

Post condition(s) for Test: A voting system with the totalNumBallots and candidate variables set. The corresponding candidate objects will also be created with necessary information

Test Stage: Unit__ System_X__ **Test Date:** 4-27-2021
Test Case ID#: 061 **Name(s) of Testers:** Eileen Campbell

Test Description: testInvalidIRBallotsUp() ensure the functions readIRCSV() and readBallots() do not distribute invalid ballots to candidates. Round up when calculating the required number of candidates each ballot must rank.

testInvalidIRBallotsUp() exists in the TestIRElection.java file. readIRCSV() and readBallots() exist in the IRElection.java file

Automated: Yes_X__ No___ **Results:** Pass_X_ Fail___

Preconditions for Test: The file "IRInvalidTestUp.csv" is in the same directory as IRElection.java and TestIRElection.java

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Test the correct number of ballots variable was set	17	17	
2	Test the correct the number of valid ballots were initialized	11	11	
3	Test the var for total number of unvalid ballots is correct	6	6	
4	Test the correct number of invalid ballots were initialized	6	6	
5	Test the correct number of ranks each ballot but have is correct	3	3	

6	Test the correct person won	Spiderman	Spiderman	
7	Test all the losers were removed from the candidates list	1	1	
8	Test the winner received the correct number of ballots	7	7	
9	Test the correct number of candidates were added to the eliminated candidate list	4	4	
10	Test the 1st eliminated candidate had the correct number of ballots assigned to them	0	0	
11	Test the 2nd eliminated candidate had the correct number of ballots assigned to them	0	0	
12	Test the 3rd eliminated candidate had the correct number of ballots assigned to them	0	0	
13	Test the 4th eliminated candidate had the correct number of ballots assigned to them	4	4	

Post Condition(s) for Test: All invalid ballots that are distributed to candidates are valid, all invalid a ballots are added to a separate list.

Test Stage: Unit__ System__X_ **Test Date:** 4-27-2021
Test Case ID#: 062 **Name(s) of Testers:** Eileen Campbell

Test Description: testInvalidIRBallotsDown() ensure the functions readIRCSV() and readBallots() do not distribute invalid ballots to candidates.

testInvalidIRBallotsDown() exists in the TestIRElection.java file. readIRCSV() and readBallots() exist in the IRElection.java file

Automated: Yes_X__ No__ **Results:** Pass_X_ Fail___

Preconditions for Test: The file “IRInvalidTestDown.csv” is in the same directory as IRElection.java and TestIRElection.java

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**

1	Test the correct number of ballots variable was set	15	15	
2	Test the correct the number of valid ballots were initialized	110	110	
3	Test the var for total number of unvalid ballots is correct	5	5	
4	Test the correct number of invalid ballots were initialized	5	5	
5	Test the correct number of ranks each ballot but have is correct	2	2	
6	Test the correct person won	CaptainAmerica	CaptainAmerica	
7	Test all the losers were removed from the candidates list	1	1	
8	Test the winner received the correct number of ballots	8	8	
9	Test the correct number of candidates were added to the eliminated candidate list	3	3	
10	Test the 1st eliminated candidate had the correct number of ballots assigned to them	0	0	
11	Test the 2nd eliminated candidate had the correct number of ballots assigned to them	0	0	
12	Test the 3rd eliminated candidate had the correct number of ballots assigned to them	0	0	
13	Test the 4th eliminated candidate had the correct number of ballots assigned to them	4	4	

Post Condition(s) for Test: All invalid ballots that are distributed to candidates are valid, all invalid a ballots are added to a separate list.

TestOPLEelection

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 020						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testReadOPLCSVValidInput: This function tests that the readOPLCSV function correctly sets up the OPLElection object to read in ballots. The variables totalNumBallots, totalNumSeats, numSeatsLeft, and quota should be set. The party and candidate ArrayLists should be created and have Party and Candidate objects assigned to them					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: The file “OPLTest.txt” is is in the project directory						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that readOPLCSV correctly sets totalNumBallots		9	9	
2	Check that readOPLCSV correctly sets totalNumSeats		3	3	
3	Check that readOPLCSV correctly sets numSeatsLeft		3	3	
4	Check that readOPLCSV correctly sets quota		3	3	The expected is calculated as (9/3)
5	Check that readOPLCSV correctly sets the size of the Party ArrayList		3	3	
6	Check that readOPLCSV correctly sets the size of the Candidate ArrayList		6	6	
7	Check that readOPLCSV correctly sets candidate		“Pike”	“Pike”	Only checks first object

	information				
8	Check that readOPLCSV correctly sets party information		“D”	“D”	Only checks first object

Post condition(s) for Test: An OPLElection voting system will be created with totalNumBallots, totalNumSeats, numSeatsLeft, and quota set. The Party and Candidate ArrayLists will also be populated.

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 021						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testReadOPLCSVInvalidSecondLine: This function tests that the readOPLCSV function correctly deals with the case when the 2nd line (number of candidates) is not in the expected format					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: The file “OPLInvTest2.csv” is in the project directory						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that readOPLCSV does not set totalNumBallots		0	0	Check that the variable is set to 0 since this is the default in java
2	Check that readOPLCSV does not set totalNumSeats		0	0	Check that the variable is set to 0 since this is the default in java
3	Check that readOPLCSV does not set numSeatsLeft		0	0	Check that the variable is set to 0 since this is the default in java
4	Check that readOPLCSV does not set quota		0	0	Check that the variable is set to 0 since this is the default in java

5	Check that readOPLCSV did not instantiate the party ArrayList		null	null	
6	Check that readOPLCSV did not instantiate the candidate ArrayList		null	null	

Post condition(s) for Test: An OPLElection voting system will be created with no variables initialized

---

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 022					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testReadOPLCSVInvalidThirdLine: This function tests that the readOPLCSV function correctly deals with the case when the 3rd line (list of candidates) is not in the expected format				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: The file “OPLInvTest3.csv” is in the project directory					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that readOPLCSV does not set totalNumBallots		0	0	Check that the variable is set to 0 since this is the default in java
2	Check that readOPLCSV does not set totalNumSeats		0	0	Check that the variable is set to 0 since this is the default in java
3	Check that readOPLCSV does not set numSeatsLeft		0	0	Check that the variable is set to 0 since this is the default in java
4	Check that		0	0	Check that the variable is

	readOPLCSV does not set quota				set to 0 since this is the default in java
5	Check that readOPLCSV did not instantiate the party ArrayList		null	null	
6	Check that readOPLCSV did not instantiate the candidate ArrayList		null	null	

Post condition(s) for Test: An OPLElection voting system will be created with no variables initialized

---

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 023					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testReadOPLCSVInvalidFourthLine: This function tests that the readOPLCSV function correctly deals with the case when the 4th line (number of seats) is not in the expected format				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: The file “OPLInvTest4.csv” is in the project directory					

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that readOPLCSV does not set totalNumBallots		0	0	Check that the variable is set to 0 since this is the default in java
2	Check that readOPLCSV does not set totalNumSeats		0	0	Check that the variable is set to 0 since this is the default in java
3	Check that readOPLCSV does not		0	0	Check that the variable is set to 0 since this is the

	set numSeatsLeft				default in java
4	Check that readOPLCSV does not set quota		0	0	Check that the variable is set to 0 since this is the default in java
5	Check that readOPLCSV creates the correct number of parties		3	3	
6	Check that readOPLCSV creates the correct number of candidates		6	6	
7	Check that readOPLCSV correctly set all candidate information		“Pike”	“Pike”	Checks the first object only
8	Check that readOPLCSV correctly set all party information		“D”	“D”	Checks the first object only

Post condition(s) for Test: A voting system with the candidate and party arrayLists populated with the correct number of corresponding objects.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 024					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testReadOPLCSVInvalidFifthLine: This function tests that the readOPLCSV function correctly deals with the case when the 5th line (number of votes) is not in the expected format				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: The file “OPLInvTest5.csv” is in the project directory					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that readOPLCSV does not set totalNumBallots		0	0	Check that the variable is set to 0 since this is the default in java
2	Check that readOPLCSV correctly sets totalNumSeats		3	3	
3	Check that readOPLCSV correctly sets numSeatsLeft		3	3	
4	Check that readOPLCSV does not set quota		0	0	Check that the variable is set to 0 since this is the default in java
5	Check that readOPLCSV creates the correct number of parties		3	3	
6	Check that readOPLCSV creates the correct number of candidates		6	6	
7	Check that readOPLCSV correctly set all candidate information		“Pike”	“Pike”	Checks the first object only
8	Check that readOPLCSV correctly set all party information		“D”	“D”	Checks the first object only

Post condition(s) for Test: A voting system with the totalNumSeats and numSeatsLeft variables set. The candidate and party arrayLists are also populated with the correct number of corresponding objects.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 025					**Name(s) of Testers:** Maranda Donaldson

Test Description:	testReadOPLCSVQuotaDivByZero: This function tests that the readOPLCSV (which sets the quota variable) correctly deals with the case when the total number of seats is set to 0					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: The file “OPLInvTestDivByZero.csv” is in the project directory						

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that readOPLCSV does not set totalNumBallots		0	0	Check that the variable is set to 0 since this is the default in java
2	Check that readOPLCSV correctly sets totalNumSeats		0	0	
3	Check that readOPLCSV correctly sets numSeatsLeft		0	0	
4	Check that readOPLCSV does not set quota		0	0	Check that the variable is set to 0 since this is the default in java
5	Check that readOPLCSV correctly set the number of parties		3	3	
6	Check that readOPLCSV correctly set the number of candidates		6	6	
7	Check that readOPLCSV correctly set all candidate information		“Pike”	“Pike”	Checks the first object only
8	Check that readOPLCSV correctly set all party information		“D”	“D”	Checks the first object only

Post condition(s) for Test: A voting system with the correct number of parties and candidates created and assigned is created.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 026					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testReadBallotsValidInput: This function tests that the readBallots function correctly creates and assigns ballots				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: The file “OPLTest.csv” is in the project directory					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to readOPLCSV correctly read the total number of ballots supposed to be in the file	The string “OPLTest” is passed in	9	9	
2	Check that the call to readBallots correctly created and assigned ballot objects		9	9	This check counts the number of ballot objects assigned to all candidates in the system

Post condition(s) for Test: A voting system with the totalNumBallots, totalNumSeats, numSeatsLeft, quota, party, and candidate variables set. Corresponding candidate and ballot objects will also be created

---

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 027					**Name(s) of Testers:** Maranda Donaldson

Test Description:	testReadBallotsInvalidInput: This function tests that the readBallots function correctly deals with the case when input ballot data is incorrect or missing.					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: The file “OPLTestInvalid.txt” is in the project directory						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the call to readOPLCSV correctly read the total number of ballots supposed to be in the file	The string “OPLTestInvalid\n” is passed in	9	9	
2	Check that no actual ballot objects were created		0	0	

Post condition(s) for Test: A voting system with the totalNumBallots, totalNumSeats, numSeatsLeft, quota, party, and candidate variables set. Corresponding candidate objects will also be created

---

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 028					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPrintToScreen: This function tests that the printToScreen function correctly prints to the screen				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: None, all set up is done in the test function					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the output to the screen is as expected	A multi-line string representing the expected output	Strings are equal	Strings are equal	System.out is redirected to an OutputStream so that output can be compared

Post condition(s) for Test: A voting system with the totalNumBallots and party variables set. The corresponding candidate and party objects will also be created with necessary information

---

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 029					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testWriteToMediaFile: This function tests that the writeToMediaFile function correctly prints to the opened media file				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: None, all set up is done in the test function					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the output to the media file is as expected	A multi-line string representing the expected output	Strings are equal	Strings are equal	The multi-line string is compared to a string representation of the printed file

Post condition(s) for Test: A voting system with the mediaFile, totalNumBallots, and party variables set. The corresponding candidate and party objects will also be created with necessary information

---

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021

Test Case ID#: 030						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testWriteToAuditFile: This function tests that the writeToAudit function correctly prints to the opened audit file					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, all set up is done in the test function						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the output to the audit file is as expected	A multi-line string representing the expected output	Strings are equal	Strings are equal	The multi-line string is compared to a string representation of the printed file

Post condition(s) for Test: A voting system with the auditFile, totalNumBallots, and party variables set. The corresponding candidate, ballot, and party objects will also be created with necessary information

---

Test Stage: Unit X System___ **Test Date:** 3-13-21
Test Case ID#: 031 **Name(s) of Testers:** Hazel Dunn

Test Description: testAllocateByQuota checks that the allocateByQuota function in the OPLElection.java file runs correctly. It will check to see that based on the quota of the current election, the correct number of seats were allocated to each candidate.
→This test is stored in the TestOPLElection testing file and this test itself is named testAllocateByQuota. It will make sure the function/method allocateByQuota in OPLElection runs correctly.

Automated: Yes___ No X **Results:** Pass X Fail___

Preconditions for Test: Allocate by quota is one of the last things to run after almost the whole system has successfully run. It occurs immediately after the ballots are read in. Once they are read in they are able to be allocated to their respective spots.

In the setup, set the total number of seats, num seats left to total number of seats, party array list to parties from file, total num ballots to ballots included in file, set quota based on given number of ballots and number of seats. Set party num ballots for each party based on the number of ballots they receive in the file.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Check that there is at least one party in the current election	True	True	Can't allocate any seats if there are no parties
2	Check that the current quota is greater than 0	True	True	A quota of zero would not allow for any seat allocation
3	Call election allocateByQuota	N/A	N/A	
4	Check that party at index 0 (D) has correct number of seats allocated	1	1	
5	Check that party at index 1 (R) has correct number of seats allocated	1	1	
6	Check that party at index 2 (I) has correct number of seats allocated	0	0	
7	Check that party at index 0 (D) has correct remainder	2	2	
8	Check that party at index 1 (R) has correct remainder	0	0	
9	Check that party at index 2 (I) has correct remainder	1	1	

Post Condition(s) for Test: Allocate by quota completes successfully, and the ballots have been allocated to their respective candidates and parties. It has assigned seats to the parties based on the quotas.

Test Stage: Unit X System___ **Test Date:** 3-14-21
Test Case ID#: 032 **Name(s) of Testers:** Hazel Dunn

Test Description: testAllocateByRemainder will verify that the function allocateByRemainder works correctly. After seats have been allocated to parties with the quota for the election, there may be remaining seats that must be allocated, which is what this function will do.
→ This test is stored in the TestOPLElection.java in the testing folder. It uses setters for OPLElection objects to set the election data, it uses getParty and getRemainder, then calls allocateByQuota and allocateByRemainder to distribute seats.
Automated: Yes__X_ No **Results:** Pass X Fail ___

Preconditions for Test: Allocate by remainder is one of the last things to run after almost the whole system has successfully run. It occurs immediately after allocateByQuota if there are seats remaining.
In setup, a new OPL election is created, total num seats is set to 3, set num seats left to 3, set array of parties to {“D”, “R”, “I”}, set total num ballots to 9, calculate and set the quota, and set party num ballots for each party.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Check that the remainder for party at index 0 is not null	True	True	
2	Check that the remainder for party at index 1 is not null	True	True	
3	Check that the remainder for party at index 2 is not null	True	True	
4	Check that totalNumSeats is not equal to 0	True	True	Don’t want to allocate by remainder if there are no seats remaining
5	Call allocate by quota and allocate by remainder to allocate seats to parties	N/A	N/A	Allocate by remainder must be called after allocate by quota in order to work properly
6	Check that the number of seats for party at index 0 is correct	2	2	
7	Check that the number of seats for party at index 1 is correct	1	1	

8	Check that the number of seats for party at index 2 is correct	0	0	

Post Condition(s) for Test: All seats have been allocated to parties. Following the allocateByQuota, allocateByRemainder allocates the remaining seats if there were any left.

Test Stage: Unit_X_ System___ **Test Date:** 3-26-2021
Test Case ID#: 016 **Name(s) of Testers:** Eileen Campbell

Test Description: testPartyNumBallots() ensures the function partyNumBallots() works correctly. Set all parties' total number of ballots and sort the candidate arrays by popularity.

testPartyNumBallots() is in the TestOPLElection.java file and tests the function PartyNumBallots() from the OPLElection.java file.

Automated: Yes_X__ No__ **Results:** Pass_X_ Fail___

Preconditions for Test: Party objects can be successfully created. Candidate and ballot objects can be successfully created. And distributed.

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	okayestParty has zero candidates and zero ballots	0	0	
2	bestParty has 7 ballots	7	7	7 ballots were assigned to candidates that are in the bestParty
3	partyRock has 6 ballots	6	6	6 ballots were assigned to candidates that are in partyRock
4	bestParty candidates are sorted by popularity	candidate@*id*	candidate@*id*	The object id is assigned by the system to each candidate object. The assertEquals will ensure the objects are the same. There are 4 assertEquals for the 4 candidates of the party. The asserts that are associated with this step are referenced in

				the test code.
5	partyRock candidates are sorted by popularity	Candidate Object: *candidate@id*	Candidate Object: *candidate@id*	The id is assigned by the system to each candidate object. The assertEquals will ensure the objects are the same. There are 2 assertEquals for the 2 candidates of the party. The asserts that are associated with this step are referenced in the test code.

Post Condition(s) for Test: Each party will have an accurate pNumBallots variable value and candidates array will be sorted by popularity.

Test Stage: Unit___ System_X__ **Test Date:** 3-14-2021
Test Case ID#: 033 **Name(s) of Testers:** Eileen Campbell

Test Description: testOPLRunElection() from the TestOPLElection.java file ensures the method runElection() from the OPLElection.java file functions correctly.

Automated: Yes_X__ No__ **Results:** Pass_X__ Fail___

Preconditions for Test: The file OPLTest.csv exists in the same folder as the TestOPLElection file. The CSV file is in the correct format specified in the SDD.

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	All candidates are added to votingSystem array	6	6	
2	Election type is set correctly	OPL	OPL	
3	CSVFile name is set correctly	OPLTest	OPLTest	
4	Number of seats was set correctly	3	3	
5	Number of seats left was set correctly	0	0	At the end of the election, all seats should be given out
6	Candidates were added the D party	2	2	
7	Correct number if seat allocated to	2	2	

	the D party			
8	Correct number of ballots associated with the D party	5	5	
9	Correct number of candidates were added the R party	2	2	
10	Correct number if seat allocated to the R party	1	1	
11	Correct number of ballots associated with the R party	3	3	
12	Candidates were added the I party	2	2	
13	Correct number if seat allocated to the I party	0	0	
14	Correct number of ballots associated with the I party	1	1	

Post Condition(s) for Test: The OPL election system is run in the correct order with the correct winners and number of allocated seats

---

Test Stage: Unit_X_ System___ **Test Date:** 3-14-2021
Test Case ID#: 034 **Name(s) of Testers:** Eileen Campbell

Test Description: testOPLTiedHighestRemainder() is in the TestOPLElection.java file. It tests the function renElection() in the OPLElection.java file. This tests the special edge case when all but one seat was allocated and there is a tie between parties for the highest remainder
Automated: Yes__X_ No__ **Results:** Pass_X__ Fail___

Preconditions for Test: The OPLTiedHighestReminder.csv file is in the same folder as the TestOPLElection.java file. The CSV file is in the correct format specified in the SDD.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Number of candidates	4	4	
2	Election Type	OPL	OPL	

3	CSV File Name	OPLTiedHighestRemainder	OPLTiedHighestRemainder	
4	Total Number of Ballots	12	12	
5	Number of Seats to be allocated	3	3	
6	Number of seats left after the election system has run	0	0	
7	Number of Political Parties	2	2	
8	Calculate the Quota	4	4	
9	Number of candidates associated with the D party	2	2	
10	Number of seats allocated to the D Party is either 1 or 2	True	True	Allocated 2 seats if the last seat was allocated to the D party during the coin flip tie. 1 seats if it did not win the tie
11	Number of ballots assigned to the D Party	6	6	
12	Number of candidates associated with the N party	2	2	
13	Number of seats allocated to the N Party is either 1 or 2	True	True	Allocated 2 seats if the last seat was allocated to the N party during the coin flip tie. 1 seats if it did not win the tie
14	Number of ballots assigned to the N Party	6	6	
15	The D party was randomly assigned 50% of the time with a 5% error during the coin toss	True	True	The D party won the last seat between 450-550 out of 100 times
16	The N party was randomly assigned 50% of the time with a 5% error during the coin toss	True	True	The N party won the last seat between 450-550 out of 100 times

Post Condition(s) for Test: The last seat was randomly assigned to one of the parties with the highest remainder using a coin toss function.

Test Stage: Unit___ System X **Test Date:** 3-24-2021
Test Case ID#: 035 **Name(s) of Testers:** Hazel Dunn

Test Description: testOPLTieBetweenCandidates will check the edge case of a tie between candidates in an OPL election. This will occur if a party receives a seat, and two candidates in that party have the same number of votes.
→ This test is stored in the TestOPLElection.java file, and the test is called testOPLTieBetweenCandidates()
→ This test calls the runElection() method
→ This test will use the sample CSV file in the testing folder called OPLTieBetweenCandidates.csv

Automated: Yes_X__ No__ **Results:** Pass_X_ Fail___

Preconditions for Test: PromptCSV, PromptMedia, and PromptAudit must run successfully and successfully open the sample CSV file.
In setup, user input is simulated, the method promptCSV() is called, as well as promptAudit() and promptMedia(), opens sample CSV file for use.
Create 2 counters and set them to 0 initially, run a for loop 1000 times to check election results with a tie, will be used to keep track of election results and check that a tie will occur.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Check that the counter for the first candidate is not greater than 700	True	True	Tie limit we set is 500+-200
2	Check that the counter for the first candidate is not less than 300 (tie limit we set is 500+-200)	True	True	Tie limit we set is 500+-200
3	Check that the counter for the second candidate is not greater than 700 (tie limit we set is 500+-200)	True	True	Tie limit we set is 500+-200
4	Check that the counter for the first candidate is not less than 300 (tie limit we set is 500+-200)	True	True	Tie limit we set is 500+-200

Post Condition(s) for Test: An OPLElection with a tie between two candidates results in equal chances of both candidates winning the tie for the seat for their party.

TestParty

Test Stage: Unit_X_ System___ **Test Date:** 3-26-2021
Test Case ID#: 036 **Name(s) of Testers:** Eileen Campbell

Test Description: testAddCandidate() will ensure that candidates are added to a candidates array of the party object. Null candidates will not be added. Candidates which don't match the party name will not be added

testAddCandidate() is in the TestParty.java file. It tests the function addCandidates from the Party.java file.

Automated: Yes_X__ No__ **Results:** Pass_X_ Fail___

Preconditions for Test: Candidate objects are assigned cName and cParty and.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Add two regular candidates to the party array	2	2	Expect the party's candidate array to be of size 2
2	Add 1 null candidate to the party array	2	2	Null candidates should not be added to the array
	Add 1 candidate with a different party association	2	2	Should not add candidates with a different party to the array

Post Condition(s) for Test: Only non null candidate objects which are part of the party will be added to the party's candidate array.

Test Stage: Unit_X_ System___ **Test Date:** 3-26-2021
Test Case ID#: 037 **Name(s) of Testers:** Eileen Campbell

Test Description: testCalculateNumBallots() will ensure that the method calculateNumBallots goes into a party and sets the number of ballots associated with the party.

testCalculateNumBallots() is in the TestPaty.java file. It tests the function calculateNumBallots() in the Party.java file.

Automated: Yes__X_ No__ **Results:** Pass_X_ Fail___

Preconditions for Test: Candidate objects are successfully made and added to the party array of candidates. A Party object is successfully created

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Zero ballots assigned to parties with no candidates	0	0	
2	BestParty should have 9 ballots	9	9	2 candidates assigned 9 ballots in total

Post Condition(s) for Test: the party class variable pNumBallots will be set to the total number of ballots assigned to the candidates in the party's candidate array, the total number of ballot votes for the party.

Test Stage: Unit X System___ **Test Date:** 3-13-21
Test Case ID#: 038 **Name(s) of Testers:** Olivia Hansen

Test Description: testSortCandidates() will ensure that the candidates array for the party the function is called on is sorted by number of ballots.

testSortCandidates() is in the TestParty.java file. It tests the function sortCandidates() from the Party.java file. Both of these files can be found in the src folder of the Project1 directory.

Automated: Yes_X__ No __ **Results:** Pass X Fail___

Preconditions for Test: Candidate objects are assigned cName, cParty, and cNumBallots. They are assigned to a party.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Tests that candidate1 is in the first position. This is tested by calling bestParty.getCandidates().get(0).getcName()	Rosen	Rosen	
2	Tests that candidate3 is in the second position. This is tested by calling bestParty.getCandidates().get(1).getcName()	Chou	Chou	Set up so candidate1 has the majority.

3	Tests that candidate4 is in the third position. This is tested by calling bestParty.getCandidates().get(2).getcName()	Royce	Royce	Set up so there is only one candidate in the array, but they still do not have the majority
4	Tests that candidate2 is in the fourth position. This is tested by calling bestParty.getCandidates().get(3).getcName()	Kleinberg	Kleinberg	Set up so there is only one candidate in the array and they do have the majority.
5	Tests that candidate1 is in the first position. This is tested by calling bestParty.getCandidates().get(0).getcName()	Rosen	Rosen	
6	Tests that candidate4 is in the second position. This is tested by calling bestParty.getCandidates().get(1).getcName()	Royce	Royce	
7	Test that candidate2 OR candidate3 is in the third position of the array. This is tested by calling bestParty.getCandidates().get(2).getcName().equals("Kleinberg") || bestParty.getCandidates().get(2).getcName().equals("Chou")	Kleinberg || Chou	Kleinberg || Chou	
8	Test that candidate2 OR candiate3 is in the fourth position of the array. This is tested by calling bestParty.getCandidates().get(3).getcName().equals("Kleinberg") || bestParty.getCandidates().get(3).getcName().equals("Chou")	Kleinberg || Chou	Kleinberg || Chou	

Post Condition(s) for Test: All candidates are sorted in the candidates array. The cases for no ties and tie are both tested and return the expected values.

TestVotingSystem

Test Stage: Unit ___ System X **Test Date:** 3-13-21
Test Case ID#: 039 **Name(s) of Testers:** Hazel Dunn

Test Description: testIRElec tests the main function for the general voting system, in this case if there is an IR election that is being run. The sample file being used to test correctness is IRTest.csv in the src folder.

→ This function is in the TestVotingSystem.java file, and is testing the main function from VotingSystem.java

- method/functions: testing main, calls main and uses Voting System getters to check that results are correct.

Automated: Yes__X_ No __ **Results:** Pass X Fail ___

Preconditions for Test: promptCSV, promptMedia and promptAudit have to complete successfully. The files have to exist and be created. In the setup, a new voting system is created.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Check that CSV name is correct (what was inputted)	IRTest.csv	IRTest.csv	
2	Check that election type is correct	IR	IR	
3	Check that total num ballots is correct	6	6	
4	Check that candidates array is correct at index 0	Rosen	Rosen	
5	Check that candidates array is correct at index 1	Kleinberg	Kleinberg	
6	Check that candidates array is correct at index 2	Chou	Chou	
7	Check that candidates array is correct at index 3	Royce	Royce	
8	Check that size of candidates array is correct	4	4	

Post Condition(s) for Test: Main has been run with the IR voting system. Results have been checked for the voting system objects with the IRTest.csv file.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021

Test Case ID#: 040			**Name(s) of Testers:** Maranda Donaldson
Test Description:	testIRElecLong: This function tests that runElection can process just over 100,000 ballots in under 8 minutes (480,000 milliseconds)		

Automated	**Yes**	X	**No**				
Results	**Pass**	X	**Fail**				

Preconditions for Test: The file “IRTestLong.csv” is in the project directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the timer set up for this function is under 480,000 milliseconds	The string “IRTestLong” is input	End-start < 480000	End-start < 480000	

Post condition(s) for Test: A voting system with all necessary variables set. The specifics of the variables are not necessary for this test as all that is checked is the runtime. The files “IRTestLongAuditFile.txt” and “IRTestLongMediaFile.txt” will have been created and subsequently deleted.

Test Stage: Unit ___ System X **Test Date:** 3-13-21
Test Case ID#: 041 **Name(s) of Testers:** Hazel Dunn

Test Description: testOPLElec tests the main function for the general voting system, in this case if there is an OPL election that is being run. The sample file being used to test correctness is OPLTest.csv in the src folder.
→ This function is in the TestVotingSystem.java file, and is testing the main function from VotingSystem.java
- method/functions: testing main, calls main and uses Voting System getters to check that results are correct.

Automated: Yes_X__ No __ **Results:** Pass X Fail___

Preconditions for Test: promptCSV, promptMedia and promptAudit have to complete successfully. The files have to exist and be created. In the setup, a new voting system is created.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Set election type to OPL	N/A - just setup	N/A	
2	Go through main code that is included in the text with current input - like calling main	N/A - just calling for setup	N/A	Like a call to main, we inserted the main code here so we could run it like it would be run.
3	Check that CSV filename is correct	OPLTest	OPLTest	
4	Check that election type is correct	OPL	OPL	
5	Check that total num ballots is correct	9	9	
6	Check that candidates array is correct at index 0	Pike	Pike	
7	Check that candidates array is correct at index 1	Foster	Foster	
8	Check that candidates array is correct at index 2	Deutsch	Deutsch	
9	Check that candidates array is correct at index 3	Borg	Borg	
10	Check that candidates array is correct at index 4	Jones	Jones	
11	Check that candidates array is correct at index 5	Smith	Smith	
12	Check that size of candidates array is correct	6	6	

Post Condition(s) for Test: Main has been run with the OPL voting system. Results have been checked for the voting system objects with the OPLTest.csv file.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 042					**Name(s) of Testers:** Maranda Donaldson

Test Description:	testOPLElecLong: This function tests that runElection can process just over 100,000 ballots in under 8 minutes (480,000 milliseconds)					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: The file “OPLTestLong.csv” is in the project directory						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the timer set up for this function is under 480,000 milliseconds	The string “OPLTestLong” is input	End-start < 480000	End-start < 480000	

Post condition(s) for Test: A voting system with all necessary variables set. The specifics of the variables are not necessary for this test as all that is checked is the runtime. The files “OPLTestLongAuditFile.txt” and “OPLTestLongMediaFile.txt” will have been created and subsequently deleted.

Test Stage: Unit_X_ System___ **Test Date:** 3-26-2021
Test Case ID#: 043 **Name(s) of Testers:** Eileen Campbell

Test Description: testCoinToss() ensures the function coinToss() functions property. Generate and return a random number.

testCoinToss() is in the TestVotingSystem.java file and tests the function coinToss() in the VotingSystem.java file. It also tests the coinToss() function in Party.java because the code is the same as in VotingSystem.java testCoinToss().

Automated: Yes_X__ No__ **Results:** Pass_X__ Fail___

Preconditions for Test: A VotingSystem object can be generated.

Step #	Test Step Description	Expected Result	Actual Result	Notes

1	Negative integer as the argument	-1	-1	
2	Zero as the argument	-1	-1	
3	Loop generates random number: result is within the [0,13)	0 <= result < 13	0 <= result < 13	

Post Condition(s) for Test: The integer returned from coinToss will be between 0, inclusive, a and 13, exclusive, if a positive integer is passed in. Else a -1 will be returned.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 044					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptCSVFirstLineIR: This function tests that the promptCSV function correctly handles the case when the first line of the input file is equal to the string "IR"				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: The file "IRTest.csv" is in the project directory					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to promptCSV returns an non-null object	The string "IRTest" is input	A non-null object is returned	A non-null object is returned	
2	Check that the returned object is of type IRElection		The returned object is an instance of IRElection	The returned object is an instance of IRElection	
3	Check that the variable csvFile is initialized		Variable is non-null	Variable is non-null	

4	Check that the variable electionType is set		“IR”	“IR”	

Post condition(s) for Test: A voting system of type IRElection is returned with the csvFile, csvName, and electionType variables set

---

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 045						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptCSVFirstLineOPL: This function tests that the promptCSV function correctly handles the case when the first line of the input file is equal to the string “OPL”					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: The file “OPLTest.csv” is in the project directory						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to promptCSV returns a non-null object	The string “OPLTest” is input	A non-null object is returned	A non-null object is returned	
2	Check that the returned object is of type OPLElection		The returned object is an instance of OPLElection	The returned object is an instance of OPLElection	
3	Check that the variable csvFile is initialized		Variable is non-null	Variable is non-null	
4	Check that the variable electionType is set to “OPL”		“OPL”	“OPL”	

Post condition(s) for Test: A voting system of type OPLElection is returned with the csvFile, csvName, and electionType variables set.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 046					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptCSVFirstLineInvalid: This function tests that the promptCSV function correctly deals with the case when the first line of the input file is not one of the expected strings (“IR” or “OPL”)				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: The file “InvalidTest.csv” is in the project directory					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to promptCSV returns a null object	The string “InvalidTest” is input -> this will instantiate a Scanner for the “InvalidTest.txt” file	null object returned	null object returned	
2	Check that the system outputs the expected string to the screen	A multi-line string representing the expected printed output	Strings are equal	Strings are equal	System.out has been reassigned to an OutputStream that is accessible by the tests

Post condition(s) for Test: A voting system with no variables set

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 047					**Name(s) of Testers:** Maranda Donaldson

Test Description:	testPromptCSVInvalidFile: This function tests that the promptCSV function correctly handles when the user enters a file name that cannot be found by the system. (For both inputs)					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: The file blah.csv does not exist within the project directory						

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that a call to promptCSV returns a null object	The string "blah\n" is input	null object returned	null object returned	
2	Check that the system outputs the expected string to the screen	A multi-line string representing expected printed output	Strings are equal	Strings are equal	System.out has been reassigned to an OutputStream that is accessible by the tests

Post condition(s) for Test: A voting system with no variables set

---

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 048					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptAuditFirstInput: This function tests that the promptAudit function correctly deals with the case that the user accepts their first input				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: None, all set up is done in the testing function					

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**

1	Check that a call to promptAudit correctly instantiates the auditFile PrintWriter variable	The string "testAudit1\n Y" is input	auditFile is not null	auditFile is not null	The input string represents 2 lines of user input
2	Check that the created .txt file has the expected naming conventions	Pathname = "testAudit1.txt"	File exists	File exists	

Post condition(s) for Test: A voting system with the auditFile variable set. The file "testAudit1.txt" is created and subsequently deleted.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 049					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptAuditSecondInput: This function tests that the promptAudit function correctly deals with the case that the user rejects their first input				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: None, setup is done in the testing function					

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that a call to promptAudit correctly initializes the auditFile PrintWriter variable	The string "testAudit1\n N\n testAudit2" is input	auditFile is not null	auditFile is not null	The input string represents 3 lines of user input
2	Check that the .txt file produced has the expected naming conventions	Pathname = "testAudit2.txt"	File exists	File exists	

Post condition(s) for Test: A voting system with the auditFile variable set. The file testAudit2.txt is created and subsequently deleted.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 050					**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptAuditDefault: This function tests that the promptAudit function correctly deals with the case that the user chooses to use the default naming convention				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: None, setup is done in test function					

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to promptAudit will correctly initialize the mediaFile PrintWriter variable	The string "D" is passed in	auditFile is not null	auditFile is not null	The string "D" is used to indicate that the default naming convention should be used ("d" or "default" or "Default" could have been passed in)
	Check that the created .txt file has the expected naming conventions	Pathname = "DefaultTestAuditFile.txt"	File exists	File exists	

Post condition(s) for Test: A voting system with the mediaFile and CSVName variables set. The file "DefaultTestAuditFile.txt" will have been created and subsequently deleted.

Test Stage:	**Unit**	X	**System**		**Test Date:** 3-25-2021
Test Case ID#: 051					**Name(s) of Testers:** Maranda Donaldson

Test Description:	testPromptAuditInvalidInput: This function tests that the promptAudit function function correctly deals with any user input that would cause an I/O exception					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, all set up is done in the testing function						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to promptAudit with the given input will not initialize the auditFile PrintWriter variable	The string “help?” is passed in as the name for the audit file	auditFile is null	auditFile is null	The string “/help?” is not seen as a valid file name on windows and linux machines, this may not be true for all machines

Post condition(s) for Test: A voting system with no variables set

---

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 052						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptMediaFirstInput: This function tests that the promptMedia function correctly deals with the case that the user accepts their first input					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, setup is done in test function						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to	The string	mediaFile	mediaFile	The input string represents

	promptMedia will correctly initialize the mediaFile PrintWriter variable	“testMedia1\n Y” is input	is not null	is not null	2 lines of user input
2	Check that the created .txt file has the expected naming conventions	Pathname = “testMedia1.txt”	File exists	File Exists	

Post condition(s) for Test: A voting system with the mediaFile variable set. The file “testMedia1.txt” is created and subsequently deleted.

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 053						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptMediaSecondInput: This function tests that the promptMedia function correctly deals with the case that the user rejects their first input					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, setup is done in test function						

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that a call to promptMedia will correctly initialize the mediaFile PrintWriter variable	The string “testMedia1\n N\n testAudit2” is input	mediaFile is not null	mediaFile is not Null	The input string represents 3 lines of user input
2	Check that the created .txt file has the expected naming conventions	Pathname = “testMedia2.txt”	File exists	File exists	

Post condition(s) for Test: A voting system with the mediaFile variable set. The file “testMedia2.txt” will have been created and subsequently deleted.

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 054						**Name(s) of Testers:** Maranda Donaldson
Test Description:	testPromptMediaDefault: This function tests that the promptMedia function correctly deals with the case that the user chooses to use the default naming convention					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, setup is done in the test function						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to promptMedia will correctly initialize the mediaFile PrintWriter variable	The string “D” is passed in	mediaFile is not null	mediaFile is not null	The string “D” is used to indicate that the default naming convention should be used (“d” or “default” or “Default” could have been passed in)
2	Check that the created .txt has the expected naming conventions	Pathname = “DefaultTestMedia File.txt”	File exists	File exists	

Post condition(s) for Test: A voting system with the mediaFile and CSVName variables set. The file “DefaultTestMediaFile.txt” will have been created and subsequently deleted.

Test Stage:	**Unit**	X	**System**			**Test Date:** 3-25-2021
Test Case ID#: 055						**Name(s) of Testers:** Maranda Donaldson

Test Description:	testPromptMediaInvalidInput: This function tests that the promptMedia function correctly deals with any user input that would cause an I/O exception					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, setup is done in the test function						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that a call to promptMedia with the given input will not initialize the mediaFile PrintWriter variable	The string “help?” is passed in as the name for the media file	mediaFile is null	mediaFile is null	The string “/help?” is not seen as a valid file name on windows and linux machines, this may not be true for all machines

Post condition(s) for Test: A voting system with no variables set

Test Stage: Unit___ System_X_ **Test Date:** 4-28-21
Test Case ID#: 056 **Name(s) of Testers:** Hazel Dunn

Test Description: testIRElecMultipleFiles() tests the success of running the election system with multiple CSV input files, testing the main function for the general voting system, in this case for an IR Election. The sample files being used are IRTest.csv, IRTestMult1.csv, IRTestMult2.csv, IRTestMult3.csv, and IRTestMult4.csv.They are all run within the same system, but CSV files may come from different locations, but are all in the src folder.

→ This function is in the TestVotingSystem.java file, and is testing the main function from VotingSystem.java

- method/functions: testing main, calls main and uses Voting System getters to check that results are correct.

Automated: Yes_X__ No__ **Results:** Pass_X_ Fail___

Preconditions for Test: promptCSV, promptMedia and promptAudit have to complete successfully. The files have to exist and be created. In the setup, a new voting system is created.

Step #	Test Step Description	Expected Result	Actual Result	Notes

1	Check that CSV name is correct (what was inputted)	IRTestMu lt4	IRTestMu lt4	
2	Check that election type is correct	IR	IR	
3	Check that total num ballots is correct	43	43	
4	Check that candidates array is correct at index 0	Rosen	Rosen	
5	Check that candidates array is correct at index 1	Kleinberg	Kleinberg	
6	Check that candidates array is correct at index 2	Chou	Chou	
7	Check that candidates array is correct at index 3	Royce	Royce	
8	Check that size of candidates array is correct	4	4	

Post Condition(s) for Test: Main has been run with the IR voting system. Results have been checked for the voting system objects with the IRTest.csv, IRTestMult1.csv, IRTestMult2, IRTestMult3, and IRTestMult4.csv files.

Test Stage: Unit ___ System X **Test Date:** 4-28-21
Test Case ID#: 057 **Name(s) of Testers:** Hazel Dunn

Test Description: testOPLElecMultipleFiles() tests success of running the election system with multiple CSV input files, testing the main function for the general voting system, in this case for an IR Election. The sample files being used are OPLTest.csv, OPLTestMult1.csv, OPLTestMult2.csv, OPLTestMult3.csv, and OPLTestMult4.csv. They are all run within the same system, but CSV files may come from different locations, but are all in the src folder. T

→ This function is in the TestVotingSystem.java file, and is testing the main function from VotingSystem.java

- method/functions: testing main, calls main and uses Voting System getters to check that results are correct.

Automated: Yes_X__ No __ **Results:** Pass X Fail___

Preconditions for Test: promptCSV, promptMedia and promptAudit have to complete successfully. The files have to exist and be created. In the setup, a new voting system is created.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Set election type to OPL, go through main code that is included in the text with current input - like calling main	N/A - just setup	N/A	Like a call to main, we inserted the main code here so we could run it like it would be run.
2	Check that CSV filename is correct	OPLTestMult4	OPLTestMult4	
4	Check that election type is correct	OPL	OPL	
5	Check that total num ballots is correct	50	50	
6	Check that candidates array is correct at index 0	Pike	Pike	
7	Check that candidates array is correct at index 1	Foster	Foster	
8	Check that candidates array is correct at index 2	Deutsch	Deutsch	
9	Check that candidates array is correct at index 3	Borg	Borg	
10	Check that candidates array is correct at index 4	Jones	Jones	
11	Check that candidates array is correct at index 5	Smith	Smith	
12	Check that size of candidates array is correct	6	6	

Post Condition(s) for Test: Main has been run with the OPL voting system. Results have been checked for the voting system objects with the OPLTest.csv file.

Test Stage:	**Unit**		**System**	X	**Test Date:** 4-28-2021
Test Case ID#: 058					**Name(s) of Testers:** Hazel Dunn

Test Description:	testPOElecMultipleFiles(): This function tests that the runElection() for the POElection class set the expected variables to the expected values for the files POTest.csv, POTestMult1.csv, POTestMult2.csv, POTestMult3.csv, and POTestMult4.csv.					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, setup is done in the test function						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the size of the candidates array was read in correctly	getCandidates().size()	6	6	
2	The variable election type was read in and set correctly	getElectionType()	"PO"	"PO"	
3	The cvsFile name was read in and set correctly	getCSVName()	"POTestMult4"	"POTestMult4"	
4	The variable totalNumBallots was read in and set correctly	getTotalNumBallots()	45	45	
5	The first candidate was read in correctly	getCandidates.get(0).getcName()	"Pike"	"Pike"	
6	The second candidate was read in correctly	getCandidates.get(1).getcName()	"Foster"	"Foster"	
7	The third candidate was read in correctly	getCandidates.get(2).getcName()	"Deutsch"	"Deutsch"	
8	The fourth candidate was read in correctly	getCandidates.get(3).getcName()	"Borg"	"Borg"	
9	The fifth candidate was read in correctly	getCandidates.get(4).getcName()	"Jones"	"Jones"	
10	The sixth candidate was read in correctly	getCandidates.get(5).getcName()	"Smith"	"Smith"	
11	The number of ballots	getCandidates.get	15	15	

	for the first candidate was read in correctly	(0).getcBallots().size()			
12	The number of ballots for the second candidate was read in correctly	getCandidates.get(1).getcBallots().size()	10	10	
13	The number of ballots for the third candidate was read in correctly	getCandidates.get(2).getcBallots().size()	0	0	
14	The number of ballots for the fourth candidate was read in correctly	getCandidates.get(3).getcBallots().size()	10	10	
15	The number of ballots for the fifth candidate was read in correctly	getCandidates.get(4).getcBallots().size()	5	5	
16	The number of ballots for the sixth candidate was read in correctly	getCandidates.get(5).getcBallots().size()	5	5	

Post condition(s) for Test: There is a poElection object with the candidates array set, the total number of ballots set, the election type set, the csvFile name set, and all of the ballots assigned to candidates.

TestPOElection

Test Stage:	**Unit**	X	**System**			**Test Date:** 4-27-2021
Test Case ID#: 059						**Name(s) of Testers:** Olivia Hansen
Test Description:	testPORunElection1: This function tests that the runElection() for the POElection class set the expected variables to the expected values for the file POTest.csv					
Automated	**Yes**	X	**No**			
Results	**Pass**	X	**Fail**			
Preconditions for Test: None, setup is done in the test function						

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the size of the candidates array was read in correctly	getCandidates().size()	6	6	
2	The variable election type was read in and set correctly	getElectionType()	“PO”	“PO”	
3	The cvsFile name was read in and set correctly	getCSVName()	“POTest”	“POTest”	
4	The variable totalNumBallots was read in and set correctly	getTotalNumBallots()	9	9	
5	The first candidate was read in correctly	getCandidates.get(0).getcName()	“Pike”	“Pike”	
6	The second candidate was read in correctly	getCandidates.get(1).getcName()	“Foster”	“Foster”	
7	The third candidate was read in correctly	getCandidates.get(2).getcName()	“Deutsch”	“Deutsch”	
8	The fourth candidate was read in correctly	getCandidates.get(3).getcName()	“Borg”	“Borg”	
9	The fifth candidate was read in correctly	getCandidates.get(4).getcName()	“Jones”	“Jones”	
10	The sixth candidate was read in correctly	getCandidates.get(5).getcName()	“Smith”	“Smith”	
11	The number of ballots for the first candidate was read in correctly	getCandidates.get(0).getcBallots().size()	3	3	
12	The number of ballots for the second candidate was read in correctly	getCandidates.get(1).getcBallots().size()	2	2	
13	The number of ballots for the third candidate was read in correctly	getCandidates.get(2).getcBallots().size()	0	0	

14	The number of ballots for the fourth candidate was read in correctly	getCandidates.get(3).getcBallots().size()	2	2	
15	The number of ballots for the fifth candidate was read in correctly	getCandidates.get(4).getcBallots().size()	1	1	
16	The number of ballots for the sixth candidate was read in correctly	getCandidates.get(5).getcBallots().size()	1	1	

Post condition(s) for Test: There is a poElection object with the candidates array set, the total number of ballots set, the election type set, the csvFile name set, and all of the ballots assigned to candidates.

Test Stage:	**Unit**	X	**System**		**Test Date:** 4-27-2021
Test Case ID#: 060					**Name(s) of Testers:** Olivia Hansen
Test Description:	testPORunElection2: This function tests that the runElection() for the POElection class set the expected variables to the expected values for the file POTest2.csv				
Automated	**Yes**	X	**No**		
Results	**Pass**	X	**Fail**		
Preconditions for Test: None, setup is done in the test function					

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that the size of the candidates array was read in correctly	getCandidates().size()	4	4	
2	The variable election type was read in and set correctly	getElectionType()	“PO”	“PO”	
3	The cvsFile name was read in and set	getCSVName()	“POTest2”	“POTest2”	

	correctly				
4	The variable totalNumBallots was read in and set correctly	getTotalNumBallots()	12	12	
5	The first candidate was read in correctly	getCandidates.get(0).getcName()	“Spongebob”	“Spongebob”	
6	The second candidate was read in correctly	getCandidates.get(1).getcName()	“Ferb”	“Ferb”	
7	The third candidate was read in correctly	getCandidates.get(2).getcName()	“PerryThePlatypus”	“PerryThePlatypus”	
8	The fourth candidate was read in correctly	getCandidates.get(3).getcName()	“Patrick”	“Patrick”	
9	The number of ballots for the first candidate was read in correctly	getCandidates.get(0).getcBallots().size()	4	4	
10	The number of ballots for the second candidate was read in correctly	getCandidates.get(1).getcBallots().size()	2	2	
11	The number of ballots for the third candidate was read in correctly	getCandidates.get(2).getcBallots().size()	4	4	
12	The number of ballots for the fourth candidate was read in correctly	getCandidates.get(3).getcBallots().size()	2	2	

Post condition(s) for Test: There is a poElection object with the candidates array set, the total number of ballots set, the election type set, the csvFile name set, and all of the ballots assigned to candidates.