Project Name: Project 1: Voting System **Team# 6**

TestCandidate

Test Stage: Unit X System___ **Test Date:** 3-14-21
Test Case ID#: 001 **Name(s) of Testers:** Hazel Dunn

Test Description: 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. 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___ No _X_ **Results:** Pass X
Fail___

Preconditions for Test: The step that precedes this function is that the candidates must be created from the OPL file, and the ballots must be read in order for them to be added.

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	Ballot objects are created and the IDs of each are set in order.	N/A	N/A	Using 4 sample ballot objects to test the correctness of this function
2	Check that the ID of ballot 1 is correct	001	001	
3	Check that the ID of ballot 2 is correct	002	002	
4	Check that the ID of ballot 3 is correct	003	003	
5	Check that the ID of ballot 1 is not 0	True	True	IDs must be assigned, cannot be zero
6	Check that the ID of ballot 2 is not 0	True	True	
7	Check that the ID of ballot 3 is not 0	True	True	
8	Create a new candidate object and add ballots 1 and 2 to the candidate ballot array	N/A	N/A	
9	Check that the ballot at index 0 of the candidate's ballot array is	ballot1	ballot1	

	correct			
10	Check that the ballot at index 1 of the candidate’s ballot array is correct	ballot2	ballot2	
11	Check that the size of the candidate’s current ballot array is correct	2	2	
12	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
13	Check that the ballot at index 0 of the candidate’s ballot array is correct	ballot1	ballot1	
14	Check that the ballot at index 1 of the candidate’s ballot array is correct	ballot2	ballot2	
15	Check that the size of the candidate’s current ballot array is correct	2	2	
16	Add ballot3 (non-null ballot) to the candidate	N/A	N/A	This ballot should add and alter the data
17	Check that the ballot at index 0 of the candidate’s ballot array is correct	ballot1	ballot1	
18	Check that the ballot at index 1 of the candidate’s ballot array is correct	ballot2	ballot2	
19	Check that the ballot at index 2 of the candidate’s ballot array is correct	ballot3	ballot3	
20	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.

Test Stage: Unit_X_ System___ **Test Date:** 3-13-2021
Test Case ID#: 002 **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 eh party will not be added

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

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

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

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Create and distribute party and candidate objects	N/A	N/A	1 party, 2 candidates, 1 null candidate, 1 candidate with a different party
2	Add two regular candidates to the party array	2	2	Expect the party's candidate array to be of size 2
3	Add 1 null candidate to the party array	2	2	Null candidates should not be added to the array
4	Add 1 candidate with a different party association	2	2	Should not add candidates with a different party association

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:** 3-12-2021
Test Case ID#: 003 **Name(s) of Testers:** Eileen Campbell

Test Description: testCalculateNumBallots() will ensure that the method calculateNumBallots goes into a party and totals 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___ No_X_ **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	Ensure no candidates are part of the “okayestParty”	0	0	
2	Zero ballots assigned to parties with no candidates	0	0	
3	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 the party’s candidate array, the total number of ballot votes for the party.

Test Stage: Unit_X_ System___ **Test Date:** 3-13-2021
Test Case ID#:004 **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.

Automated: Yes___ No_X_ **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	Create and distribute party and candidate objects	N/A	N/A	1 party, 4 candidates. The candidate’s cNumBallots are set as follows: candidate1: 3 candidate2: 0 candidate3: 2 candidate4: 1
2	sortCandidates() is called on the party object	N/A	N/A	N/A
3	Tests that candidate1 is in the first	Rosen	Rosen	Candidate1 has the most ballots, so it

	position. This is tested by calling bestParty.getCandidates().get(0).getName()			should be sorted into the first position of the array.
4	Tests that candidate3 is in the second position. This is tested by calling bestParty.getCandidates().get(1).getName()	Chou	Chou	Candidate3 has the second most ballots, so it should be sorted into the second position of the array.
5	Tests that candidate4 is in the third position. This is tested by calling bestParty.getCandidates().get(2).getName()	Royce	Royce	Candidate4 has the third most ballots, so it should be sorted into the third position of the array.
6	Tests that candidate2 is in the fourth position. This is tested by calling bestParty.getCandidates().get(3).getName()	Kleinberg	Kleinberg	Candidiate2 has the fourth most ballots, so it should be sorted into the fourth position of the array.
7	Changed the cNumBallots as follows: candidate1: 4 candidate2: 0 candidate3: 0 candidate4: 2	N/A	N/A	This set-up is designed to test a tie in the sortCandidates() function.
8	sortCandidates() is called on the party object	N/A	N/A	N/A
9	Tests that candidate1 is in the first position. This is tested by calling bestParty.getCandidates().get(0).getName()	Rosen	Rosen	Candidate1 has the most ballots, so it should be sorted into the first position of the array.
10	Tests that candidate4 is in the second position. This is tested by calling bestParty.getCandidates().get(1).getName()	Royce	Royce	Candidate4 has the second most ballots, so it should be sorted into the second position of the array.
11	Test that candidate2 OR candidate3 is in the third position of the array. This is tested by calling bestParty.getCandidates().get(2).getName().equals("Kleinberg") || bestParty.getCandidates().get(2).getName().equals("Chou")	Kleinberg || Chou	Kleinberg || Chou	Since Kleinberg and Chou are tied for the third and fourth position, the tests should check if one OR the other is in each position.
12	Test that candidate2 OR candiate3	Kleinberg	Kleinberg	Since Kleinberg and Chou are tied for

	is in the fourth position of the array. This is tested by calling bestParty.getCandidates().get(3).getName().equals("Kleinberg") || bestParty.getCandidates().get(3).getName().equals("Chou")	|| Chou	|| Chou	the third and fourth position, the tests should check if one OR the other is in each position.

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

TestIRElection

Test Stage: Unit_X_ System___ **Test Date:** 3-13-2021
Test Case ID#: 005 **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.

Automated: Yes___ No_X_ **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	Create candidate objects and election objects and assign the appropriate variables as stated in the preconditions	N/A	N/A	4 candidate objects, 1 election object. candidate1 has 3 ballots, candidate2 has 0 ballots, candidate3 has 2 ballots, and candidate4 has 1 ballot. All candidates are added to currCandidates array and totalNumBallots is set to 6.
2	findMajority() is called on the election object	null	null	Since there is currently no candidate with the majority, it is expected that

				findMajority() should return null.
3	Add a ballot to candidate1 and remove a ballot from candidate3	N/A	N/A	We add a ballot to candidate1 and remove a ballot from candidate3, keeping the totalNumBallots the same but giving candidate1 the majority
4	findMajority() is called on the election object	Rosen	Rosen	It is expected that findmajority() will return candidate1, as they now have the majority. getcName() is called on candidate1 to return the name of the candidate, which should be Rosen.
5	Remove all candidates from currCandidates except for candidate1. Set candidate1's cNumBallots back to 3	N/A	N/A	This set-up is designed to test the situation where there is only 1 candidate left, but there is still no majority.
6	findMajority() is called on the election object	null	null	It is expected that even though there is only 1 candidate left, there is still no majority, so findMajority() should still return null.
7	Set candidate1's numBallots to 4	N/A	N/A	This set-up is designed to test if findMajority() will work when there is only 1 candidate left and there is a clear majority
8	findMajority() is called on the election object	Rosen	Rosen	It is expected that findmajority() will return candidate1, as they now have the majority. getcName() is called on candidate1 to return the name of the candidate, which should be Rosen.

Post Condition(s) for Test: currCandidates has been set to only have candidate1 in it, and candidate 1 is set to have a majority. The cases of multiple candidates no majority, multiple candidates majority, 1 candidate no majority, and 1 candidate majority have all been tested and return the expected results.

Test Stage: Unit_X_ System___ **Test Date:** 3-13-2021
Test Case ID#: 006 **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.

Automated: Yes___ No_X_ **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	Create candidate objects and election objects and assign the appropriate variables as stated in the preconditions	N/A	N/A	4 candidate objects, 1 election object. candidate1 has 3 ballots, candidate2 has 0 ballots, candidate3 has 2 ballots, and candidate4 has 1 ballot. All candidates are added to currCandidates array and totalNumBallots is set to 6.
2	findLeastCand() is called on the election object	Kleinberg	Kleinberg	Candidate2 currently has 0 ballots, so it is expected the findLeastCand() should return candidate2. getcName() is called on candidate2 to return the name of the candidate, which should be Kleinberg.
3	Candidate2 is removed from the currCandidates array and added to the eliminated candidates array	N/A	N/A	This set-up is designed to be able to test if findLeastCand() can find the next least candidate in the currCandidates array.
4	findLeastCand() is called on the election object	Royce	Royce	Candidate4 currently has 1 ballot, which is now the least amount of ballots in the currCandidates array, so it is expected the findLeastCand() should return candidate4. getcName() is called on candidate4 to return the name of the candidate, which should be Royce.
5	Candidate4 is removed from the currCandidates array and added to the eliminated candidates array	N/A	N/A	This set-up is designed to be able to test if findLeastCand() can find the next least candidate in the currCandidates array
6	findLeastCand() is called on the election object	Chou	Chou	Candidate3 currently has 2 ballots, which is now the least amount of

				ballots in the currCandidates array, so it is expected the findLeastCand() should return candidate3. getcName() is called on candidate3 to return the name of the candidate, which should be Chou.
7	Candidate3 is removed from the currCandidates array and added to the eliminated candidates array	N/A	N/A	This set-up is designed to be able to test is findLeastCand() can find the least candidate in the currCandidates array when there is only 1 candidate left.
8	findLeastCand() is called on the election object	Rosen	Rosen	Candidate1 currently has 3 ballots, which is now the least amount of ballots in the currCandidates array, so it is expected the findLeastCand() should return candidate1. getcName() is called on candidate1 to return the name of the candidate, which should be Rosen.
9	All of the candidates cNumBallots variables are set to the following values: candidate1: 4 candidate2: 0 candidate3: 0 candidate4: 2 All candidates are added back into the currCandidates array and removed from the eliminatedCandidates array	N/A	N/A	This set-up is designed to test how findLeastCand() handles ties for the least number of ballots in the currCandiates array
10	findLeastCand() is called on the election object	Kleinberg || Chou	Kleinberg || Chou	Candidate2 and candidate3 both currently have 0 ballots, which means they are tied for the last position. findLeastCand() is expected to call coinToss() to determine which is to be removed from the currCandidates array. Since coinToss() is random, the results of this test could be different depending on the results of coinToss(), so the findLeastCand() should return either candidate2 OR candidate3. getcName() is called on the returned candidate, which should be Kleinberg OR Chou.

Post Condition(s) for Test: The cases for multiple candidates left, one candidate left, and ties are all tested and returning the expected values.

Test Stage: Unit_X_ System___ **Test Date:** 3-13-2021
Test Case ID#: 007 **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.

Automated: Yes___ No_X_ **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	Create candidate objects, Ballot objects, and election objects and assign the appropriate variables as stated in the preconditions	N/A	N/A	4 candidate objects, 6 Ballot objects, 1 election object. candidate1 has 3 ballots, candidate2 has 0 ballots, candidate3 has 2 ballots, and candidate4 has 1 ballot. All candidates are added to currCandidates array and totalNumBallots is set to 6. The ballots ranking arrays are set to the following (number referring to candidates): ballot1: 1,4,2,3 ballot2: 1,3 ballot3: 1,2,3 ballot4: 3,2,1,4 ballot5: 3,4 ballot6: 4
2	redistrubuteBallots() is called on the election object	N/A	N/A	Since redistribute ballots does not return anything, the tests for actual vs

				expected will be on what values it sets.
3	Checked that a candidate was added to eliminatedCandidates by calling eliminatedCandidates.size()	1	1	It is expected that one candidate should be added to the eliminatedCandidates array, so eliminatedCandidates.size() should return 1.
4	Check that the correct candidate was added to eliminatedCandidates by calling eliminatedCandidates.get(0).getcName();	Kleinberg	Kleinberg	Candidate2 currently has 0 ballots, so it is expected that it will be the candidate who is added to the eliminated candidates array.
5	Check that candidate was removed from the currCandidates array by calling currCandidates.size()	3	3	It is expected that one candidate should be removed from the currCandidates array, so currCandidates.size() should return 3.
6	Check that the correct candidate was removed from the currCandidates array by calling currCandidates.contains(candidate2)	false	false	It is expected that candidate2 should be removed from the currCandidates array, so currCandidates.contains(candidate2) should return false.
7	redistrubuteBallots() is called on the election object	N/A	N/A	Since redistribute ballots does not return anything, the tests for actual vs expected will be on what values it sets.
8	Checked that a candidate was added to eliminatedCandidates by calling eliminatedCandidates.size()	2	2	It is expected that one more candidate should be added to the eliminatedCandidates array, so eliminatedCandidates.size() should return 2.
9	Check that the correct candidate was added to eliminatedCandidates by calling eliminatedCandidates.get(1).getcName();	Royce	Royce	Candidate4 currently has 1 ballot, so it is expected that it will be the candidate who is added to the eliminated candidates array.
10	Check that candidate4's ballot was not redistributed, since candidate4 was the only candidate in it's ranking array. This is checked by calling candidate4.getcBallots().contains(ballot6)	true	true	Since candidate4 only had one ballot, and that ballot only had candidate4 in its ranking array, this ballot should not be redistributed, so candidate4.getcBallots().contains(ballot6) should return true.

11	Check that candidate was removed from the currCandidates array by calling currCandidates.size()	2	2	It is expected that one more candidate should be removed from the currCandidates array, so currCandidates.size() should return 2.
12	Check that the correct candidate was removed from the currCandidates array by calling currCandidates.contains(candidate4)	false	false	It is expected that candidate4 should be removed from the currCandidates array, so currCandidates.contains(candidate4) should return false.
13	redistrubuteBallots() is called on the election object	N/A	N/A	Since redistribute ballots does not return anything, the tests for actual vs expected will be on what values it sets.
14	Checked that a candidate was added to eliminatedCandidates by calling eliminatedCandidates.size()	3	3	It is expected that one more candidate should be added to the eliminatedCandidates array, so eliminatedCandidates.size() should return 3.
15	Check that the correct candidate was added to eliminatedCandidates by calling eliminatedCandidates.get(2).getcName();	Chou	Chou	Candidate3 currently has 2 ballots, so it is expected that it will be the candidate who is added to the eliminated candidates array.
16	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	Since one of candidate3's ballots only had candidate4 in its ranking array, this ballot should not be redistributed, so candidate3.getcBallots().contains(ballot5) should return true.
17	Check that candidate was removed from the currCandidates array by calling currCandidates.size()	1	1	It is expected that one more candidate should be removed from the currCandidates array, so currCandidates.size() should return 1.
18	Check that the correct candidate was removed from the currCandidates array by calling currCandidates.contains(candidate3)	false	false	It is expected that candidate3 should be removed from the currCandidates array, so currCandidates.contains(candidate3) should return false.
19	Check that candidate3's other ballot (ballot4) was not distributed to an eliminated candidate, even though	false	false	Candidate2 is the next candidate in ballot4's ranking array, but they have already been eliminated, so the ballot

	they are the next candidate in the ranking array. This is checked by calling candidate2.getcBallots().contains(ballot4)			should not be added to their ballot array.
20	Check that ballot4 is redistributed to the correct candidate. This is checked by calling candidate1.getcBallots().contains(ballot4)	true	true	The next candidate in ballot4's ranking array is candidate1. Since candidae1 is still in play, the ballot should be added to candidate1's ballot array.

Post Condition(s) for Test: Candidate1 should be the only candidate left in the currCandidates array. All other candidates should be in the eliminated candidates array.

Test Stage: Unit__ System_X__ **Test Date:** 3-13-2021
Test Case ID#: 008 **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.

Automated: Yes___ No_X_ **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	The correct input for the CSV file, the media file, and the audit file are set. An IRElection object is created.	N/A	N/A	N/A
2	runElection() is called on the election object.	N/A	N/A	N/A
3	Check if the electionType is set to the correct value. This is tested by calling election.getElectionType()	IR	IR	Makes sure that run election sets the electionType variable to IR.

4	Check if the CSV name is set to the correct value. This is tested by calling election.getCsvName().	IRTest	IRTest	Makes sure that run election sets the CSVName variable to IRTest.
5	Check if the totalNumBallots is set to the correct value. This is tested by calling election.getTotalNumBallots().	6	6	Makes sure that run election sets the totalNumBallots variable to 6.
6	Check if the correct winner is the only candidate left in the currCandidates array. This is tested by calling election.getCurrCandidate().get(0).getcName()	Rosen	Rosen	Makes sure the correct winner is in the first position of the currCandidates array.
7	Check that the winner has the correct number of ballots. This is tested by calling election.getCurrCandidate().get(0).getcNumBallots()	4	4	Makes sure the winner has the correct number of ballots. In this case, it should be 4.
8	Check that all other candidates are moved to the eliminated candidates array. This is tested by calling election.getEliminatedCandidates().size()	3	3	Makes sure all 3 other candidates are moved to eliminatedCandidates.
9	Make sure losers were removed from currCandidates. This is tested by calling election.getCurrCandidates().size()	1	1	Makes sure that only the winner is left in the currCandidates array.
10	Make sure that the correct candidate was moved to eliminatedCandidates first. This is tested by calling election.getEliminatedCandidates().get(0).getcName().	Kleinberg	Kleinberg	Makes sure that Kleinberg was the first candidate eliminated.
11	Make sure that the correct candidate was moved to eliminatedCandidates second. This is tested by calling election.getEliminatedCandidates().get(1).getcName().	Royce	Royce	Makes sure that Royce was the second candidate eliminated.
12	Make sure that the correct candidate was moved to eliminatedCandidates third. This is tested by calling	Chou	Chou	Makes sure that Chou was the third candidate eliminated.

	election.getEliminatedCandidates().get(2).getcName().			
13	Make sure Kleinberg's ballots were redistributed correctly. This is tested by calling election.getEliminatedCandidates().get(0).getcNumBallots()	0	0	Makes sure that Kleinberg still has zero ballots.
14	Make sure Royce's ballots were redistributed correctly. This is tested by calling election.getEliminatedCandidates().get(1).getcNumBallots()	1	1	Make sures Royce's one ballot was not redistributed.
15	Make sure Chou's ballots were redistributed correctly. This is tested by calling election.getEliminatedCandidates().get(2).getcNumBallots()	1	1	Make sure one of Chou's ballots was redistributed and the other was not.

Post Condition(s) for Test: There is only one candidate left in the currCandidates array and that is the winner. All other candidates are in the eliminatedCandidates array and all ballots are redistributed correctly. All variables set by run election are tested and set to the expected values.

TestOPLElection

Test Stage: Unit_X_ System___ **Test Date:** 3-13-2021
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___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: Party objects can be successfully created. Candidate objects can be successfully created. Ballot objects can be successfully created.

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	Create and distribute party, candidate, and ballot objects	N/A	N/A	
2	okayestParty has zero candidates and zero ballots	0	0	
3	bestParty has 7 ballots	7	7	7 ballots were assigned to candidates that are in the bestParty
4	partyRock has 6 ballots	6	6	6 ballots were assigned to candidates that are in partyRock
5	bestParty candidates are sorted by popularity	candidate@*id*	candidate@*id*	The 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.
	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 sorted by popularity.

Test Stage: Unit___ System_X__ **Test Date:** 3-14-2021
Test Case ID#: 009 **Name(s) of Testers:**

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

Automated: Yes___ No_X_ **Results:** Pass_X__ Fail___

Preconditions for Test: The file OPLTest.txt 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 the D party	2	2	
8	Correct number of ballots associated with the D party	5	5	
9	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#: 010 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the readOPLCSV function properly handles when the input file is has completely valid input
→ This test is stored in the TestOPLElection.java file (called testReadOPLCSVValidInput)
→ This test calls the readOPLCSV method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: The "OPLTest.txt" file must be in the correct processing folder

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	User input is simulated, and the methods promptCSV, and readOPLCSV are called	N/A	N/A	Sends in "OPLTest" as user input
2	Checks that totalNumBallots is set to the expected number	9	9	
3	Checks that totalNumSeats is set to the expected Number	3	3	
4	Checs that numSeatsLeft is set to the expected number	3	3	
5	Checks that quota is set to the expected number	3	3	
6	Checks that the size of the party ArrayList is the expected number	3	3	
7	Checks that the size of the candidates ArrayList is the expected number	6	6	
8	Checks that the first Candidate in the candidate ArrayList has the expected name	"Pike"	"Pike	
9	Checks that the first Party in the party ArrayList has the expected	"D"	"D"	

	name			

Post Condition(s) for Test: A VotingSystem with the totalNumBallots, totalNumSeats, numSeatsLeft, and quota variables initialized is created. The party and candidates ArrayLists should also be set.

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#:011 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the readOPLCSV function properly handles when the input file is has an invalid line
→ This test is stored in the TestOPLElection.java file (uses testReadOPLCSVInvalidSecondLine, testReadOPLCSVInvalidThirdLine, testReadOPLCSVInvalidFourthLine, and testReadOPLCSVInvalidFifthLine)
→ This test calls the readOPLCSV method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: The "OPLInvTestX.txt" files must be in the correct processing folder

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	User input is simulated, and the methods promptCSV, and readOPLCSV are called	N/A	N/A	Sends in "OPLInvTestX" as user input, where X corresponds to the line that is invalid
2	Checks that totalNumBallots is set to the expected number	0	0	
3	Checks that totalNumSeats is set to the expected Number	0	0	
4	Checs that numSeatsLeft is set to the expected number	0	0	
5	Checks that quota is set to the expected number	0	0	
6	Checks that the size of the party ArrayList is the expected number	3	3	
7	Checks that the size of the candidates ArrayList is the expected	6	6	

	number			
8	Checks that the first Candidate in the candidate ArrayList has the expected name	“Pike”	“Pike	
9	Checks that the first Party in the party ArrayList has the expected name	“D”	“D”	

Post Condition(s) for Test: A VotingSystem with some of the variables totalNumBallots, totalNumSeats, numSeatsLeft, and quota variables initialized is created. The party and candidates ArrayLists should also be set.

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#:012 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the readOPLCSV function properly handles when the input file requires that the calculation of quota includes a divide by 0
→ This test is stored in the TestOPLElection.java file (called testReadOPLCSVQuotaDivByZero)
→ This test calls the readOPLCSV method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: The “OPLInvTestDivByZero.txt” files must be in the correct processing folder

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	User input is simulated, and the methods promptCSV, and readOPLCSV are called	N/A	N/A	Sends in “OPLInvTestDivByZero” as user input
2	Checks that readOPLCSV correctly responds when the second line is invalid	True	True	
3	Checks that readOPLCSV correctly responds when the third line is invalid	True	True	

4	Checks that readOPLCSV correctly responds when the fourth line is invalid	True	True	
5	Checks that readOPLCSV correctly responds when the fifth line is invalid	True	True	

Post Condition(s) for Test: A VotingSystem with the variables totalNumBallots, totalNumSeats, numSeatsLeft, and quota variables initialized is created. The party and candidates ArrayLists should also be set.

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#:013 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the readBallots function properly handles both valid and invalid ballot lines
→ This test is stored in the TestOPLElection.java file (uses testReadBallotsValidInput and testReadBallotsInvalidInput)
→ This test calls the readBallots method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: The “OPLTest.txt” files must be in the correct processing folder

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	User input is simulated, and the methods promptCSV, and readOPLCSV, and readBallots are called	N/A	N/A	Sends in “OPLTest” as user input. Assumes that readOPLCSV sets all variables correctly
2	Checks that readBallots correctly assigned the totalNumBallots variable	9	9	
3	Checks that readBallots correctly assigned all ballots	9 assigned	9 assigned	

Post Condition(s) for Test: A VotingSystem with the variables totalNumBallots, totalNumSeats, numSeatsLeft, and quota variables initialized is created. The party and candidates ArrayLists should also be set.

The variable totalNumBallots is set by this test specifically and this test should also assign all 9 ballots to a specific candidate

TestVotingSystem

Test Stage: Unit_X_ System___ **Test Date:** 3-14-2021
Test Case ID#: 014 **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___ No_X_ **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 ___ System X **Test Date:** 3-13-21
Test Case ID#: 015 IR Test(for main) **Name(s) of Testers:** Hazel Dunn

Test Description: This test 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.txt 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___ No X **Results:** Pass X Fail ___

Preconditions for Test: promptCSV, promptMedia and promptAudit have to complete successfully. The files have to exist and be created.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Create new voting system	N/A	N/A	Creates new voting system
2	Check that CSV name is correct (what was inputted)	IRTest.txt	IRTest.txt	
3	Check that election type is correct	IR	IR	
4	Check that total num ballots is correct	6	6	
5	Check that candidates array is correct at index 0	Rosen	Rosen	
6	Check that candidates array is correct at index 1	Kleinberg	Kleinberg	
7	Check that candidates array is correct at index 2	Chou	Chou	
8	Check that candidates array is correct at index 3	Royce	Royce	
9	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.txt file.

Test Stage: Unit ___ System X **Test Date:** 3-13-21
Test Case ID#:017 OPL Test (for main) **Name(s) of Testers:** Hazel Dunn

Test Description: This test 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.txt 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___ No X **Results:** Pass X Fail___

Preconditions for Test: promptCSV, promptMedia and promptAudit have to complete successfully. The files have to exist and be 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.txt file.

Test Stage: Unit X System___ **Test Date:** 3-13-21
Test Case ID#: 018 allocate by quota **Name(s) of Testers:** Hazel Dunn

Test Description: This test 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.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	set the total number of seats, set num seats left to total number of seats, set party array list to parties from file, set 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	N/A - just setup before testing	N/A	A lot of setters, in order to use the proper data for this test.
2	Check that there is at least one party in the current election	True	True	Can't allocate any seats if there are no parties
3	Check that the	True	True	A quota of zero would not allow for

	current quota is greater than 0			any seat allocation
4	Call election allocateByQuota	N/A	N/A	
5	Check that party at index 0 (D) has correct number of seats allocated	1	1	
6	Check that party at index 1 (R) has correct number of seats allocated	1	1	
7	Check that party at index 2 (I) has correct number of seats allocated	0	0	
8	Check that party at index 0 (D) has correct remainder	2	2	
9	Check that party at index 1 (R) has correct remainder	0	0	
10	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#: 019 allocate by remainder **Name(s) of Testers:** Hazel Dunn

Test Description: This test 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___ No X **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.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	Create a new OPL election, set total num seats to 3, set num seats left to 3, set array list of parties to {“D”, “R”, “I”}, set the parties, set total num ballots to 9, calculate and set the quota, set party num ballots for each party	N/A	N/A	Setters before testing the function itself. Setting to all sample values that are desired.
2	Check that the remainder for party at index 0 is not null	True	True	
3	Check that the remainder for party at index 1 is not null	True	True	
4	Check that the remainder for party at index 2 is not null	True	True	
5	Check that totalNumSeats is not equal to 0	True	True	Don’t want to allocate by remainder if there are no seats remaining
6	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
7	Check that the number of seats for party at index 0 is correct	2	2	
8	Check that the number of seats for party at index 1 is correct	1	1	
9	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:** 3-14-2021
Test Case ID#: 020 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptCSV function properly handles when the first line of the input csv matches the string "IR".
→ This test is stored in the TestVotingSystem.java file (called testPromptCSVFirstLineIR)
→ This test calls the promptCSV method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: The file "IRTest.txt" must be in the correct folder to be processed

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	User input is simulated, and the static method promptCSV is called	N/A	N/A	Sends in "IRTest" as user input
2	Check that returned VotingSystem object is not null	True	True	
3	Check that an object of type IRElection is returned	True	True	
4	Check that the CSVFile Scanner is initialized	True	True	
5	Check that ElectionType is set correctly	"IR"	"IR"	

Post Condition(s) for Test: A VotingSystem of type IRElection is returned. The instance variables ElectionType and CsvFile are set.

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 021 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptCSV function properly handles when the first line of the input csv matches the string "OPL".
→ This test is stored in the TestVotingSystem.java file (called

testPromptCSVFirstLineOPL)
→ This test calls the promptCSV method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: The file “OPLTest.txt” must be in the correct folder to be processed

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	User input is simulated, and the static method promptCSV is called	N/A	N/A	Sends in “OPLTest” as user input
2	Check that returned VotingSystem object is not null	True	True	
3	Check that an object of type IRElection is returned	True	True	
4	Check that the CSVFile Scanner is initialized	True	True	
5	Check that ElectionType is set correctly	“OPL”	“OPL”	

Post Condition(s) for Test: A VotingSystem of type OPLElection is returned. The instance variables ElectionType and CsvFile are set.

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 022 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptCSV function properly handles when the first line of the input csv does not match “IR” or “OPL”
→ This test is stored in the TestVotingSystem.java file (called testPromptCSVFirstLineInvalid)
→ This test calls the promptCSV method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: The file “InvalidTest.txt” must be in the correct folder to be processed

Step	Test Step	Expected	Actual	Notes

#	Description	Result	Result	
1	User input is simulated, and the static method promptCSV is called	N/A	N/A	Sends in “InvalidTest” as user input
2	Check that returned VotingSystem object is null	True	True	
3	Check that the expected error output is displayed	True	True	Should display: “Invalid election type, exiting”

Post Condition(s) for Test: A null VotingSystem is returned.

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptCSV function properly handles when the input CSV is not present.
→ This test is stored in the TestVotingSystem.java file (called testPromptCSVInvalidFile)
→ This test calls the promptCSV method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: The file “blah.txt” must not be in the processing folder.

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	User input is simulated, and the static method promptCSV is called	N/A	N/A	Sends in “blah” as user input
2	Check that returned VotingSystem object is null	True	True	
3	Check that the expected error output is displayed	True	True	

Post Condition(s) for Test: A null VotingSystem is returned.

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 023 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptAudit function properly handles when the user decides to accept their first input.
→ This test is stored in the TestVotingSystem.java file (called testPromptAuditFirstInput)
→ This test calls the promptAudit method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: None

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	User input is simulated, and the method promptAudit is called	N/A	N/A	Sends in “testAudit” and “Y” as user input
2	Check that the auditFile PrintWriter is initialized	True	True	

Post Condition(s) for Test: A VotingSystem with the auditFile variable initialized is created

---

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 024 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptAudit function properly handles when the user decides to reject their first input.
→ This test is stored in the TestVotingSystem.java file (called testPromptAuditSecondInput)
→ This test calls the promptAudit method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: None

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	User input is simulated, and the	N/A	N/A	Sends in “testAudit1” , “N”, and

	method promptAudit is called			“testAudit1” as user input
2	Check that the auditFile PrintWriter is initialized	True	True	

Post Condition(s) for Test: A VotingSystem with the auditFile variable initialized is created

---

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 025 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptAudit function properly handles when the user decides to use the default naming conventions.
→ This test is stored in the TestVotingSystem.java file (called testPromptAuditDefault)
→ This test calls the promptAudit method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: None

Step #	**Test Step Description**	**Expected Result**	**Actual Result**	**Notes**
1	User input is simulated, and the method promptAudit is called	N/A	N/A	Sends in “D” as user input
2	Check that the auditFile PrintWriter is initialized	True	True	

Post Condition(s) for Test: A VotingSystem with the auditFile variable initialized is created

---

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 026 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptAudit function properly handles when the user inputs an invalid file name.
→ This test is stored in the TestVotingSystem.java file (called

testPromptAuditInvalidInput)
→ This test calls the promptAudit method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: None

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	User input is simulated, and the method promptAudit is called	N/A	N/A	Sends in “thelp?” as user input
2	Check that the auditFile PrintWriter is not initialized	True	True	

Post Condition(s) for Test: A VotingSystem with the auditFile variable not initialized is created

---

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 027 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptMedia function properly handles when the user decides to accept their first input.
→ This test is stored in the TestVotingSystem.java file (called testPromptMediaFirstInput)
→ This test calls the promptMedia method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: None

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	User input is simulated, and the method promptMedia is called	N/A	N/A	Sends in “testMedia1” and “Y” as user input
2	Check that the auditMedia PrintWriter is initialized	True	True	

Post Condition(s) for Test: A VotingSystem with the MediaFile variable initialized is created

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 028 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptMedia function properly handles when the user decides to reject their first input.
→ This test is stored in the TestVotingSystem.java file (called testPromptMediaSecondInput)
→ This test calls the promptMedia method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: None

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	User input is simulated, and the method promptMedia is called	N/A	N/A	Sends in “testMedia1”, “N”, and “testMedia1” as user input
2	Check that the auditMedia PrintWriter is initialized	True	True	

Post Condition(s) for Test: A VotingSystem with the MediaFile variable initialized is created

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 029 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptMedia function properly handles when the user decides to use the default naming conventions.
→ This test is stored in the TestVotingSystem.java file (called testPromptMediaDefault)
→ This test calls the promptMedia method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: None

Step	Test Step	Expected	Actual	Notes

#	Description	Result	Result	
1	User input is simulated, and the method promptMedia is called	N/A	N/A	Sends in “D” as user input
2	Check that the auditMedia PrintWriter is initialized	True	True	

Post Condition(s) for Test: A VotingSystem with the MediaFile variable initialized is created

---

Test Stage: Unit_X__ System___ **Test Date:** 3-14-2021
Test Case ID#: 030 **Name(s) of Testers:** Maranda Donaldson

Test Description: This test will make sure that the promptMedia function properly handles when the user decides to accept their first input.
→ This test is stored in the TestVotingSystem.java file (called testPromptMediaInvalidInput)
→ This test calls the promptMedia method

Automated: Yes___ No_X_ **Results:** Pass_X_ Fail___

Preconditions for Test: None

Step #	Test Step Description	Expected Result	Actual Result	Notes
1	User input is simulated, and the method promptMedia is called	N/A	N/A	Sends in “help?” as user input
2	Check that the auditMedia PrintWriter is not initialized	True	True	

Post Condition(s) for Test: A VotingSystem with the MediaFile variable not initialized is created

---