For our unit testing, we did not think it was necessary to write individual tests for things as getters or setters, so we instead opted to test those when testing larger overall functions (i.e. If they didn't work, these tests wouldn't work) as we were under a heavy time constraint and it's pretty impossible to test all possible cases and write the code and document everything. Also, certain methods we used did not lend themselves to unit testing. We have overarching test classes with lots of asserts in them, so we will detail those as the steps. For example, when writing an audit file, this is a protected method, but we need to run the whole program to actually produce a file. It seemed pretty impossible to test every single one of our methods, or at least we couldn't think of ways to do some of them so we tested what we felt we could, especially given the time constraints.

Unit Testing:

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit _X_ System __ **Test Date: 03/13/21**

Test Case ID#: testBallotPreferences() **Name(s) of Testers: Sean Carter**

Test Description: Tests if ballot preferences are able to be retrieved and manipulated properly. Also tests getPreferredCandidate and eliminatePreferredCandidate and their associated getters.

- **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
    - **BallotTest.java**
    - **We use the constructor, getPreferredCandiate, and eliminatePreferredCandidate**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

We initialized a list of candidates in order to test our various methods on.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check if preferred candidate for ballot1 is equal to ballot3	ballot1.getPreferredCandidate().getName()	They should not be equal	They are not equal	This part passes
2	Check that preferred candidate for ballot1 is Joe Biden	ballot1.getPreferredCandidate.getName()	Should be Joe Biden	It is Joe Biden	Test passes
3	Eliminate preferred candidate from ballot 1 and then check if the new preferred	ballot1.eliminatePreferredCandidate(), ballot1.getPreferredCandidate.getName())	The new preferred candidate should be Donald Trump	It should be Donald Trump	Test passes

	candidate is Donald Trump				
4	Get the names of the eliminated preferred candidates from ballot2	ballot2.eliminatePreferredCandidate().getName()	Should get Donald Trump, Kanye West, then Alexander Hamilton	It gets them in the correct order	Test passes
5	Eliminates ballot3's one preferred candidate, and then makes sure that getPreferredCandidate is null	ballot3.eliminatePreferredCandidate.getName(), ballot3.getPreferredCandidate()	Should get Kanye West and then null after these operations are performed in the order specified to the left.	It does get Kanye West and then null as expected	Test passes

Post condition(s) for Test:

Ballot 3 should have no preferred candidate left, as well as the effects of the manipulations detailed above. Overall, since this is just a small test file, the system itself shouldn't be affected.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit _X_ System __ **Test Date: 03/13/21**

Test Case ID#: testBallotLine() **Name(s) of Testers: Sean Carter**

Test Description: Tests the storing and retrieval of ballot data

BallotTest.java

- **testBallotLine()**
- **Ballot()**
- **getBallotInfo()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

We initialized a list of candidates in order to test our various methods on. Also, we initialized 3 ballot objects with ballot info passed in to test.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check that ballot1's ballotInfo has	String "1,2,3,4" and "1,2,3,4" passed into the ballot constructor to	Should be equal to one another "1,2,3,4"	They are equal "1,2,3,4"	Test Passed

	stored the correct string	initialize ballotInfo			
2	Check that ballot2's ballot info has stored the correct string	String "4,1,3,2" and "4,1,3,2" passed into the ballot constructor to initialize ballotInfo	Should be equal to one another "4,1,3,2"	They are equal "4,1,3,2"	Test Passed
3	Check that ballot2's ballotInfo stored the correct string	String ",,,1" and ",,,1" passed into the ballot constructor to initialize ballotInfo	Should be equal to one another ",,,1"	They are equal ",,,1"	Test Passed

Post condition(s) for Test:

ballot1, ballot2 and ballot3 will all have the correct balllotInfo stored after the lines are passed to the constructor.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit _X_ System __ **Test Date: 03/13/21**

Test Case ID#: testVoteCount() **Name(s) of Testers: Conor Brown**

Test Description: Tests if getVoteCount() and if increment vote count will work.

CandidateTest.java

- **testVoteCount()**
- **Candidate()**
- **Party()**
- **getVoteCount()**
- **incrementVoteCount()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

Initialize party in order to initialize candidates. Then initialize a candidate to run tests upon.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Confirm that candidate has correct initial vote count	0 is what the initial vote count should be and compared to kanyeWest.getVoteCount()	Should be equal to 0	The values are equal 0	Test Passed

2	Increment candidate vote count with incrementVoteCount and test that getVoteCount returns the correct incremented count	IncrementVoteCount() increments kanyeWests's votes to 1 and kanyeWest.getVoteCount()	Should be equal to 1	The values are equal 1	Test Passed
3	Increment candidate vote count again and ensure that it sets the vote count correctly	Increment kayneWest's vote count again and compare it to kanyeWest.getVoteCount()	Should be equal to 2	The values are equal 2	Test Passed
4	Call incrementVoteCount with parameter that will add the passed in value to the candidates current vote count	Call kanyeWest.incrementVoteCount(2018) then compared kanyeWest.getVoteCount() to 2020	Should be equal 2020	The values are equal 2020	Test Passed

Post condition(s) for Test:

kayneWest's vote count will be equal to 2020 and kanyeWest.getVoteCount() returns 2020.

Test Stage: Unit _X_ System __ **Test Date: 03/13/21**

Test Case ID#: testCompareTo() **Name(s) of Testers: Conor Brown**

Test Description: Test the testCompareTo() method to ensure it correctly compares two candidates

- **CandidateTest.java**
    - **testCompareTo()**
    - **Candidate()**
    - **Party()**
    - **compareTo()**
    - **incrementVoteCount()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

We initialized a party to initialize two candidates to compare to each other.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Increment candidate kanyeWest's vote count and compare it to candidate northWests	Will check to see if -1 is equal to kanyeWest.compareTo(nothWest) because kanyeWest has a greater number of votes	The values should be equal -1	The values are equal -1	Test Passed
2	Increment candidate nothWests's vote count to 1 and then compare to kanyeWestVote count	Check to see when kanyeWest.compareTo(northWest) returns 0 meaning their vote count is equal	The values should be equal 0	The values are equal 0	Test Passed
3	Increment northWest's vote count again and compare to kanyeWest's	kanyeWest.compareTo(northWest) should return 1 now that northWest's vote count is greater than kanyeWest's	The values should be equal 1	The values are equal 1	Test Passed

Post condition(s) for Test:

northWest's vote count will be equal to 2 and kanyeWest's vote count will be equal to 1.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit _X_ System __ **Test Date: 03/13/21**

Test Case ID#: testSetParty **Name(s) of Testers: Conor Brown**

Test Description: Test the setParty() function in Candidate class

- **CandidateTest.java**
  - **testSetParty()**
  - **Party()**
  - **Candidate()**
  - **getPart()**
  - **setParty()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

Initialize two parties patriots and buccaneers to test that the constructor sets te correct party when passed in and then tests that setParty() correctly changes that party on the candidate tomBrady.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to see that the constructor correct assign tomBrady's party at patriots	tomBrady.getParty() is compared to patriots the Party object	Should be equal. patriots	The values are equal patriots	Test Passed
2	Change tomBrady's party to buccaneers and use tomBrady.getParty to see if the party has been updated	tomBrady.getParty() is compared to buccaneers the party	The values should be equal buccaneers	The values are equal buccaneers	Test Passed

Post condition(s) for Test:

Candidate tomBrady's party will be equal to buccaneers.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit _X_ System __ **Test Date: 03/13/21**

Test Case ID#: testInitValues() **Name(s) of Testers: Conor Brown**

Test Description: Test that the Party class constructor assigns the correct initial values.

- **PartyTest.java**
    - **getTotalPartyVotes()**
    - **getPartyMembers()**
    - **getSeatsAllocated()**
    - **getRemainder()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

Create a Party presidential to create an object of class Party with name “Presidential”.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Check to see if the correct party name has been initialized by calling	“Presidential” is compared to presidential.getPartyName()	The values should be equal. “Presidential”	The values are equal. “Presidential”	Test passed

	presidential.getPartyName()				
2	Check to see if the Party's total vote count has been initialized correctly. Call presidential.getTotalPartyVotes()	Initial party vote count should be equal to 0. presidential.getTotalPartyVotes() is compared to 0.	The values should be equal. 0	The values are equal. 0	Test passed
3	Check if affiliatedPartyMemebers has been initialized but with no Candidates within it yet. Call presidential.getPartyMembers().size()	The size of partyMembers should initially be equal to 0. presidential.getPartyMembers().size() should be equal to 0	The values should be equal. 0	The values are equal. 0	Test passed
4	Check to see that seatsAllocated is initialized to 0. Call presidential.getSeatsAllocated().	presidential.getSeatsAllocated() should return 0 after initialization, compare this to 0	The values should be equal. 0	The values are equal. 0	Test passed
5	Check to see that presidential's remainder is equal to 0. Call presidential.getRemainder()	presidential.getRemainder() should return 0. Compare that to 0	The values should be equal. 0	The values are equal. 0	Tests passed

Post condition(s) for Test:

A Party presidential will be initialized with all the correct values after calling the constructor.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit _X_ System __ **Test Date: 03/13/21**

Test Case ID#: testPartMembers() **Name(s) of Testers: Conor Brown**

Test Description: Test that the Party class's affiliatedPartyMembers works as intended.

- **PartyTest.java**
  - **Party()**
  - **Candidate()**
  - **addCandidate()**
  - **getPartyMemebers()**
  - **incrementVoteCount()**
  - **sortPartyMembersByVote()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

Create a Party presidential to create an object of class Party with name “Presidential”. Create four Candidate objects that are affiliated with Party presidential.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test if the size of affiliatedPartyMembers for Party presidential is equal to 4 after 4 candidates have been created with the presidential Party affiliation	4 and getPartyMemebers().size()	The values should be equal. 4	The values are equal. 4	Test passed
2	Test that the affiliatedPartyMembers are in the correct initial position.	Compares the Candidate abeLicoln to presidential.getPartyMembers().get(3) which will return the Candidate at position 3 in	The values should be equal. abeLincoln	The values are equal abeLincoln	Test passed
3	Test that sorPartyMembersByVotes() works as intended	Check that Candidate teddyRoosevelt is now in last with the least amount of votes. PartyMembers(	The values should be equal. teddyRoosevelt	The values are equal. teddyRoosevelt	Test passed

		).get(3) should now return teddyRoosevelt			

Post condition(s) for Test:

Party presidential will have 4 affiliated Candidates and the affiliatedPartyMembers list will be sorted by each candidate's number of votes.

OPL System Tests:

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ **Test Date: 03/13/21**

Test Case ID#: OPLTest1() **Name(s) of Testers: Jack Soderwall**

Test Description: Runs an Open Party Listing election on a file called OPLTest1.csv. This test contained a tie, so we tested our ability to break ties.

- **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
  - **OPLTest.java**
  - **Tests our running of the OPL election on a small file, thus testing all of the functionality of the**

Automated: yes_X_no ___

OpenPartyListingElection() class of our program

Results: Pass __X__ Fail________

Preconditions for Test:

We initialized an OpenPartyListingElection class using a scanner and then ran an OPL election. We are then checking if the results are what we expect.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that 2 candidates are seated	opl.getSeatedCandidates() aka a list of the candidates that were seated in the election	2	2	Test passed
2	Check that the first candidate in the seated candidates list is Billy and he is of party D	seatedCandidates.get(0).getName(), seatedCandidates.get(0).getParty().getPartyName()	Should be Billy and D if ran in the order specified on the left	Billy, D	Test passed

3	Check that the two parties get allocated 1 seat	getParty().getSeatsAllocated()	1 and 1 since we perform 1 check for each of the parties in this election	1, 1	Test passed
4	Check that the number of seated candidates is equal to the number of seats up for grabs at the start of the election	opl.getSeats()	True	True	Test passed

Post condition(s) for Test:

Audit and media files will be placed into the directory that the code was run from that details all of the steps in the given election. Relevant results will be displayed to the terminal.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ Test Date: 03/13/21

Test Case ID#: OPLTest2() **Name(s) of Testers: Jack Soderwall**

Test Description: Runs an Open Party Listing election on a file called OPLTest2.csv

- **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
    - **OPLTest.java**
    - **Tests our running of the OPL election on a small file, thus testing all of the functionality of the OpenPartyListingElection() class of our program**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

We initialized an OpenPartyListingElection class using a scanner and then ran an OPL election. We are then checking if the results are what we expect.

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

1	Check that 3 candidates are seated	opl.getSeatedCandidates() aka a list of the candidates that were seated in the election	3	3	Test passed
2	Check that Pike's party got 5 total votes	seatedCandidates.get(0).getParty().getTotalPartyVote()	5	5	Test passed
3	Check that the seated candidate at position 0 is Pike	getName() aka the name of the seated candidate at position 0	Pike	Pike	Test passed
4	Check that the party of the seated candidate at index 1 has 3 votes	seatedCandidates.get(1).getParty().getTotalPartyVote()	3	3	Test passed
5	Check that Jones and Foster are the seated candidates at indices 1 and 2	getName() aka the names of the candidates	Jones, Foster	Jones, Foster	Test passed

Post condition(s) for Test:

Audit and media files will be placed into the directory that the code was run from that details all of the steps in the given election.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ **Test Date: 03/13/21**

Test Case ID#: OPLTestSpecial() **Name(s) of Testers: Jack Soderwall**

Test Description: Runs an Open Party Listing election on a file called OPLTestSpecial.csv. This is a very special case where a party wins more seats than it has candidates, so we tested our ability to handle that.

- **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
  - **OPLTest.java**
  - **Tests our running of the OPL election on a small file, thus testing all of the functionality of the OpenPartyListingElection() class of our program**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

We initialized an OpenPartyListingElection class using a scanner and then ran an OPL election. We are then checking if the results are what we expect.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that 3 candidates are seated	opl.getSeatedCandidates() aka a list of the candidates that were seated in the election	3	3	Test passed
2	Check that the seated candidate at index 0 got 4 votes	getTotalPartyVote aka the total number of votes cast to the party	4	4	Test passed
3	Check that the seated candidate at position 0 is Sean	getName() aka the name of the seated candidate at position 0	Sean	Sean	Test passed
4	Check that the parties of the candidates	getTotalPartyVote() AKA the			

	seated at indices 1 and 2 each have 1 vote	vote total for the party these candidates are in	1	1	Test passed
5	Check that the number of seated candidates equals the number of seats up for grabs at the start of the election	getSeats aka retrieving the number of seats that were open at the start of this election	True	True	Test passed

Post condition(s) for Test:

Audit and media files will be placed into the directory that the code was run from that details all of the steps in the given election.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ Test Date: 03/13/21

Test Case ID#: testLargeOPLFile() **Name(s) of Testers: Jack Soderwall**

Test Description: Tests the functionality of our OPL Election on a file with 100,000 ballots. Uses the file "hundredk_ballots_opl.csv"

- **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**
  - **OPLTest.java**
  - **Tests our running of the OPL election on a very large file, thus testing all of the functionality of the OpenPartyListingElection() class of our program**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

We initialized an OpenPartyListingElection class using a scanner and then ran an OPL election. We are then checking if the results are what we expect.

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

1	Check that 3 candidates are seated	opl.getSeatedCandidates() aka a list of the candidates that were seated in the election	3	3	Test passed
2	Check that the seated candidate at index 0 got 99,999 votes	getTotalPartyVote aka the total number of votes cast to the party	99,999	99,999	Test passed
3	Check that the seated candidate at position 0 is named Pike and his party is D	getName() aka the name of the seated candidate at position 0 and getParty() aka the party of the seated candidate at position 0	Pike, D	Pike, D	Test passed
4	Check that the seated candidate at position 1 is Foster and his party is D	getName() aka the name of the seated candidate at position 1 and getParty() aka the party of the seated candidate at position 1	Foster	D	Test passed
5	Check that the seated candidate at position 1 is Smith and his party is I	getName() aka the name of the seated candidate at position 2 and getParty() aka	Smith	I	Test passed

		the party of the seated candidate at position 2			

Post condition(s) for Test:

Audit and media files will be placed into the directory that the code was run from that details all of the steps in the given election.

System Testing for Instant Runoff:

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ Test Date: 03/13/21

Test Case ID#: testFixedFile() **Name(s) of Testers: Conor Brown**

Test Description: Test that the Instant Runoff Election works when a candidate has an initial clear majority.

- **IRElectionTest.java**
    - **InstantRunoffElection()**
    - **getElectionWinner()**
    - **getPartyName()**
    - **getVoteCount()**
    - **getParty()**
    - **getPartyMemebers()**
    - **getParticipatingParties()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

A specified ballot file has been created to test this scenario. The tests opens the file and a scanner to read it. An InstantRunoffElection object is created to run the election and the winner is stored.

Step #	**Test Step Description**	**Test Data**	**Expected Result**	**Actual Result**	**Notes**
1	Confirm the correct winner was found	Compare “Watters” to winner.getName()	Should be equal. Watters	They are equal Watters	Test passed
2	Confirm the winner has the correct party name	Compare “P” to winner.getParty().getaArtyName()	The values should be equal. “P”	The values are equal “P”	Test Passed
3	Confirm the correct number of votes was	Compare 3, the expect votes, to	The values should be equal	The values are equal. 3	Test passed

	found for the winner.	winner.getVoteCount()	3		
4	Ensure that all members were assigned to the correct party	Compare 4, the number of Candidates in the winner's party to winner.getParty.getPartyMembers.size()	The values should be equal. 4	The values are equal. 4	Test Passed
5	Ensure that the number of participatingParties is correct	Compare 1, the expect number of participating parties with ir.getParticipatingParties.size()	The values should be equal. 1	The values are equal 1	Test passed

Post condition(s) for Test:

An IR election will have run with the specified file and a winner declared.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ Test Date: 03/13/21

Test Case ID#: testComeback() **Name(s) of Testers: Conor Brown**

Test Description: Test that the Instant Runoff Election works when a candidate has a comeback to win after runoff.

- **IRElectionTest.java**
    - **InstantRunoffElection()**
    - **getElectionWinner()**
    - **getParty()**
    - **getPartyName()**
    - **getVoteCount()**
    - **getElectionWinner()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

A specified ballot file has been created to test this scenario. The test opens the file and a scanner to read it. An InstantRunoffElection object is created to run the election and the winner is stored.

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

1	Tests that the correct winner is found.	“Kleinberg”, the expected winner, is compared to winner.getName()	The values should be equal. “Kleinberg”	The values are equal. “Kleinberg”	Test Passed
2	Confirm that the winner is assigned the correct party	“R”, the expected party name of the winner, is compared to winner.getParty.getPartyName()	The values should be equal. “R”	The values are equal. “R”	Test Passes
3	Confirm that the winner has the correct number of votes	4, the expected number of votes for winner, is compared to winner.getVoteCount	The values should be equal. 4	The values are equal. 4	Test Passes

Post condition(s) for Test:

An IR election will have run with the specified file and a winner declared.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ Test Date: 03/13/21

Test Case ID#: testFourWayTie() **Name(s) of Testers: Conor Brown**

Test Description: Test that the Instant Runoff Election works when there is a four way tie.

- **IRElectionTest.java**
    - **InstantRunoffElection()**
    - **getEliminatedCandidates()**
    - **getParticipatingParties()**
    - **getNumBallots()**
    - **getCandidates()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

A specified ballot file has been created to test this scenario. The test opens the file and a scanner to read it. An InstantRunoffElection object is created to run the election.

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

1	Ensure that there are 3 eliminated Candidates	Compare 3, the expect number of eliminated Candidates to ir.getElimintedCandidates.getsize()	The values should be equal. 3	The values are equal. 3	Test Passed
2	Ensure that the correct number of participating parties was found and stored	Compare 4, the expected number of participating parties to ir.getParticipatingParties().size()	The values should be equal. 4	The values are equal. 4	Test Passes
3	Ensure that only one ballot remains following the election due to tie breaking and eliminations	Compare 1, the expected number of remaining ballots, to ir.getNumBallots()	The values should be equal. 1	The values are equal. 1	Test Passes
4	Ensure that only one candidate remains following the election due to tie breaking and eliminations	Compare 1, the expected number of remaining ballots to ir.getCandidates.size()	The values should be equal. 1	The values are equal 1	Test Passes

Post condition(s) for Test:

An IR election will have run with the specified file and a winner declared.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ Test Date: 03/13/21

Test Case ID#: testCandidatesWithLastNames()

Name(s) of Testers: Conor Brown

Test Description: Test that the Instant Runoff Election works when there is a four way tie.

- **IRElectionTest.java**
  - **InstantRunoffElection()**
  - **getName()**
  - **getParty()**
  - **getPartyName()**

Automated: yes_X_no ___

Results: Pass ____ Fail____X____

Preconditions for Test:

A specified ballot file has been created to test this scenario. The test opens the file and a scanner to read it. An InstantRunoffElection object is created to run the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks if the name of the winner is equal to “Mike Douglas”	winner.getName() or the name of the winner of the IR election	The values should be equal.	The values are not equal. The space is missing in between Mike and Douglas. This is addressed in the bug list.	Test Fails
2	Checks if the winner is of the party D.	winner.getParty().getPartyName() or the name of the party of the election winner	The values should be equal.	The values are equal.	Test Fails due to the above test not passing, but the values will be equal.

Post condition(s) for Test:

An IR election will have run with the specified file and a winner declared.

Project Name: Project 1: Voting System **Team# 21**

Test Stage: Unit __ System _X_ Test Date: 03/13/21

Test Case ID#: fileNotFound()

Name(s) of Testers: Conor Brown

Test Description: Test that the proper exception is thrown when an incorrect file path is given to ElectionDriver.

- **IRElectionTest.java**
  - **ElectionDriver.main()**

Automated: yes_X_no ___

Results: Pass __X__ Fail________

Preconditions for Test:

Enter in a non-existent file path which will throw an exception when passed to ElectionDriver.main().

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

1	Check to see if ElectionDriver.main() will throw the correct exception when provided with an incorrect file path	Compare an illegalArgument Exception with what exception ElectionDriver. main() throws when passed an incorrect file name	The values should be equal. IllegalArgumentException	The values are equal. IllegalArgumentException	Test Passed

Post condition(s) for Test:

The specified path will not be found and an exception should be thrown.