Project Name: Project 1: Voting System **Team# 4**

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 001

Name(s) of Testers: Varun Reddy

Test Description: isGreaterThanTests tests the function isGreaterThan(). This test is segmented into 3 junit tests.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail _____

Preconditions for Test:
Two Candidate objects must be instantalized with cand1 having more Ballots than cand2.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	use assertEquals on test data	cand1.isGreaterThan(cand2, true)	1	1	
2	use assertEquals on test data	cand2.isGreaterThan(cand1, true)	-1	-1	
3	add ballots to cand2 so that cand1 and cand2 have equal number of ballots, use assertEquals on test data	cand1.isGreaterThan(cand2, true)	0	0	

Post condition(s) for Test:

Test Stage: Unit _+_ System __

Test Date:

Test Case ID#: 002

Name(s) of Testers: Varun Reddy

Test Description: incrementVoteTest() tests the function incrementTest(). This test is split into 3 junit tests.

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

Automated: yes + no __

Results: Pass + Fail

Preconditions for Test:
A Ballot and a Candidate object must be instantailized. cand1.incrementVote(ballot1) is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data to see if the numVotes attribute is updated	cand1.getNumVotes()	1	1	
2	Call assertEquals on test data to see if the ballot is added to the candidate's ballot array	cand1.getCurrentBallots()	[OPLBallot ballot1]	[OPLBallot ballot1]	
3	Create new Candidate object, and call cand2.incrementVote(null). Call assertEquals on tests data to see if there is proper null handling.	cand2.getNumVotes()	0	0	

Post condition(s) for Test:
No post conditions for test

Test Stage: **Unit _+_** **System __**

Test Date:

Test Case ID#: 003

Name(s) of Testers: Varun Reddy

Test Description: Tests if attributes are initialized correctly inside of the Candidate class constructor for an OPL election.

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

Automated: yes + no

Results: Pass + Fail

Preconditions for Test:
A candidate object is initialized with new Candidate("Bob", "J").

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data to see if candidate name is initialized properly	testCandidate.getName()	“Bob”	“Bob”	
2	Call assertEquals on test data to see if candidate party is initialized properly	testCandidate.getParty()	“J”	“J”	

Post condition(s) for Test:

Test Stage: Unit ___ System _+_ **Test Date:** 04/29/21

Test Case ID#: 004 **Name(s) of Testers: Jonathan Leibovich**

Test Description: Testing the system for an IR election with 20 ballots and redistributing votes needed to determine the outcome

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

Automated: yes _+__ no ___

Results: Pass + Fail

Preconditions for Test: A correctly formatted csv file containing the election information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on the candidate array at the end of the election to see if the correct candidate won	electionTest.getCandidates()	Kleinberg (R)	Kleinberg (R)	

Post condition(s) for Test:

An audit and media file are made for the election

Test Stage: Unit ___ System _+__ **Test Date: 04/29/2021**

Test Case ID#: 005 **Name(s) of Testers: Jonathan Leibovich**

Test Description: Testing the system for an IR election with 20 ballots and the winner is determined by a tie breaker

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
ElectionTest.java
testSmallIR2()

Automated: yes + no ___

Results: Pass + Fail

Preconditions for Test: A correctly formatted csv file containing the election information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on the candidate array at the end of the election to see if either of the correct candidates won	electionTest.getCandidates()	Chou (I) or Royce (L)	Chou (I) or Royce (L)	
2	Call assertEquals on numWinners to ensure that there was only one candidate that won	electionTest.getCandidates()	1	1	

Post condition(s) for Test:

An audit and media file are made for the election

Test Stage: Unit ___ System _+__ **Test Date:** 04/29/21
Test Case ID#: 006
Test Description: Testing the system for an IR election with 20 ballots

Name(s) of Testers: Jonathan Leibovich

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
ElectionTest.java
testSmallIR3()

Automated: yes + no

Results: Pass + Fail

Preconditions for Test: A correctly formatted csv file containing the election information

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on the candidate array at the end of the election to see if the correct candidate won	electionTest.getCandidates()	Rosen (D)	Rosen (D)	

Post condition(s) for Test:

An audit and media file are made for the election

Test Stage: Unit _+_ System __

Test Date: 04/29/21

Test Case ID#: 007

Name(s) of Testers: Varun Reddy

Test Description: Tests if attributes are initialized correctly inside of the Candidate class constructor for an IR election.

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

Automated: yes + no

CandidateTest.java
CandidateContsructorIRTest()

Results: Pass + Fail

Preconditions for Test: A candidate object is initialized with new Candidate("Bob").

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data to see if candidate name is initialized properly	testCandidate.getName()	"Bob"	"Bob"	

Post condition(s) for Test:

No post conditions since this is a unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 008

Name(s) of Testers: Varun Reddy

Test Description: Tests if the rank index for the first choice for a ballot is initialized properly in the IRBallot constructor. Using the getCurrentRankIndex() method, we make sure the currentRankIndex is set to the index where the first choice is.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: An IRBallot is initialized with new IRBallot("1,3,4,2", 123).

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on the test data.	testBallot.getCurrentRankIndex()	0	0	

Post condition(s) for Test:

No post conditions since this is a unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 009

Name(s) of Testers: Varun Reddy

Test Description: Tests the functionality of the method updateCandidateIndex() which looks through the IR ballot's choice array for the next rank choice's index.

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

Automated: yes__+_ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: A IRBallot is initialized with new IRBallot("1,3,4,2", 123). Call updateCandidateIndex() on the IRBallot.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEqual on test data.	testBallot.getCurrentRankIndex()	3	3	The result is 3 because the next rank choice for the ballot is 2, which is located at the index 3.

Post condition(s) for Test:

No post conditions since this is a unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 010

Name(s) of Testers: Varun Reddy

Test Description: Tests an edge case of the method updateCandidateIndex() when there is no next rank choice on an IR ballot.

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

Automated: yes _+_ no ___

Results: Pass _+_ Fail ____

Preconditions for Test: A IRBallot is initialized with new IRBallot(",,,1", 123). Call updateCandidateIndex() on the IRBallot.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEqual on test data.	testBallot.getCurrentRankIndex()	-1	-1	The result is -1 representing that there is no next choice and the ballot is dead.

Post condition(s) for Test:
No post conditions since this is a unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 011

Name(s) of Testers: Varun Reddy

Test Description: Tests functionality of updateCandidateIndex() if called multiple times on an IRBallot object.

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

Automated: yes + no

Results: Pass __+___ Fail_____

Preconditions for Test: A IRBallot is initialized with new IRBallot("1,3,4,2", 123).

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEqual on test data.	testBallot.getCurrentRankIndex()	0	0	Expect 0 since the first choice is at index 0.
2	Call updateCandidateIndex() on the same IR ballot. Call assertEqual on test data.	testBallot.getCurrentRankIndex()	3	3	Expect 3 since the second choice is at index 3
3	Call updateCandidateIndex() on the same IR ballot. Call	testBallot.getCurrentRankIndex()	1	1	Expect 2 since the third choice

	assertEqual on test data.				is at index 2
4	Call updateCandidateIndex() on the same IR ballot. Call assertEqual on test data.	testBallot.getCurrentRankIndex()	2	2	Expect 1 since the fourth choice is at index 1
5	Call updateCandidateIndex() on the same IR ballot. Call assertEqual on test data.	testBallot.getCurrentRankIndex()	-1	-1	Expect -1 since there are no more choices left.

Post condition(s) for Test: No post conditions since this is a unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 012

Name(s) of Testers: Varun Reddy

Test Description: Tests the functionality of the method updateCandidateIndex() when there are indexes that have no choice specified.

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

Automated: yes__+_ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: A IRBallot is initialized with new IRBallot("1,,,2", 123).

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEqual on test data.	testBallot.getCurrentRankIndex()	0	0	Expect 0 since the first choice is at index 0.
2	Call updateCandidateIndex() on the same IR ballot. Call	testBallot.getCurrentRankIndex()	3	3	Expect 3 since the second

	assertEqual on test data.				choice is at index 3
3	Call updateCandidateIndex() on the same IR ballot. Call assertEqual on test data.	testBallot.getCurrentRankIndex()	-1	-1	Expect -1 since there are no more choices left.

Post condition(s) for Test:

No post conditions since this is a unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 013

Name(s) of Testers: Varun Reddy

Test Description: Tests the OPLBallot constructor to make sure the choice is initialized properly and the index of that choice is initialized properly.

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

Automated: yes__+_ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: A OPLBallot is initialized with OPL testBallot = new OPLBallot("1,,,,", 123). Another OPLBallot is initialized with OPL testBallot = new OPLBallot(",,1,,", 123).

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEqual on test data.	testBallot.getCandidateIndex()	0	0	Expect 0 since the choice is at index 0
2	Call assertEqual on test data.	testBallot2.getCandidateIndex()	2	2	Expect 2 since the choice is at index 2

Post condition(s) for Test:

No post conditions since this is a unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 014

Name(s) of Testers: Varun Reddy

Test Description: Tests the constructor of the Party class and checks to see if the relevant Party class attributes are initialized properly.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: A Party object is initialized with Party testParty = new Party("test"). Where test is the name of the party.

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

1	Call assertEquals on test data	testParty.getName()	“test”	“test”	
2	Call assertEquals on test data	testParty.getNumVotes()	0	0	
3	Call assertEquals on test data	testParty.getRemainder	0	0	
4	Call assertEquals on test data	testParty.getSeatsAllocated()	0	0	
5	Call assertEquals on test data	testParty.getCandidates.size()	0	0	each party has a candidate array representing the candidates in the party

Post condition(s) for Test:

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 015

Name(s) of Testers: Varun Reddy

Test Description: Tests functionality of the sumTotalPartyVotes() method, which is a function that adds up all the votes for a party.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: A Party object is initialized with Party testParty = new Party("test"). Where test is the name of the party. Candidate candidate1 and Candidate candidate2 are initialized with new Candidate("test1") and new Candidate("test2"). Two OPLBallot objects are added to candidate 1's ballot array, and one OPLBallot object is added to candidate 2's ballot. Candidate candidate1 and Candidate candidate2 is added to the party. sumTotalPartyVotes() is called on the Party object and stored in the int partyVoteTotal

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data	partyVoteTotal	3	3	

Post condition(s) for Test: no post condition since it is an individual unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 016

Name(s) of Testers: Varun Reddy

Test Description: Tests functionality of the sumTotalPartyVotes() method, which is a function that adds up all the votes for a party, when there are no candidates added.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Party testParty is initialized with new Party("test").

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data	testParty.sumTotalParty Votes()	0	0	

Post condition(s) for Test: no post condition since it is an individual unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 017

Name(s) of Testers: Varun Reddy

Test Description: Tests functionality of the addSeats() method which adds seats to the seatsAllocated attribute for the party object.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Party testParty is initialized with new Party("test"). testParty.addSeats(5) is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data	testParty.getSeatsAllocated()	5	5	

Post condition(s) for Test: no post condition since it is an individual unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 018

Name(s) of Testers: Varun Reddy

Test Description: Tests functionality of the addSeats() method which adds seats to the seatsAllocated attribute for the party object when it is called multiple times on the same party.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Party testParty is initialized with new Party("test"). testParty.addSeats(5) is called. Afterwards, testParty.addSeats(6) is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data	testParty.getSeatsAllocated()	11	11	

Post condition(s) for Test: no post condition since it is an individual unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2001

Test Case ID#: 019

Name(s) of Testers: Varun Reddy

Test Description: Tests functionality of the addCandidate() method in Party class which adds a candidate object to the candidate arrayList attribute in each party.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Party testParty is initialized with new Party("test"). Candidate candidate1 and Candidate candidate2 are initialized with new Candidate("test1") and new Candidate("test2"). candidate1 and candidate2 are added to testParty's candidate arrayList attribute using testParty.addCandidates(candidate1) and testParty.addCandidates(candidate2). An arrayList of Candidates is instantiated with ArrayList<Candidate> candidates = testParty.getCandidates(). testParty.getCandidates() returns the arrayList of candidates for the party.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data	candidates.size()	2	2	
2	Call assertEquals on test data	candidates.get(0).getName()	"test1"	"test1"	
3	Call assertEquals on test data	candidates.get(1).getName()	"test2"	"test2"	

Post condition(s) for Test:
no post condition since it is an individual unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 020

Name(s) of Testers: Varun Reddy

Test Description: Tests the functionality of the testSetRemainder() function.

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

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Party testParty is initialized with new Party("test"). testParty.setRemainder(4) is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertEquals on test data	testParty.getRemainder()	4	4	

Post condition(s) for Test:
no post condition since it is an individual unit test

Project Name: Project 1: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 021

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with an OPL election with 10 votes

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

ElectionTest.java
testSmallOPL()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Correctly formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This ensures that only the correct candidates got elected. It does this by going through each party and seeing who got nominated.	electionTest.getParties()	PikeWinner && JonesWinner && SmithWinner && valid	PikeWinner && JonesWinner && SmithWinner && valid	the "Valid" check is set to false if anyone other than Pike, Jones, or Smith was elected

2	**This test ensures that the correct amount of candidates were elected for this election**	**electionTest.getParties()**	**3**	**3**	

Post condition(s) for Test:

An audit and media file are made for the election

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 022

Name(s) of Testers: Varun Reddy

Test Description: Tests the rankCandidates() method for the Party class which sorts the ArrayList of Candidates for the party based on the number of votes of each candidate.

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

Automated: yes__+_ no ___

Results: Pass _+____ Fail_____

Preconditions for Test: Party test is initialized with new Party("test"). Candidate candidate1, Candidate candidate3 and Candidate candidate2 are initialized with new Candidate("test1"), new Candidate("test3") and new Candidate("test2"). 2 OPL ballots are added to candidate1 and 1 OPL ballot is added to candidate 3. party.rankCandidate() is called. Finally, the sorted ArrayList is stored in ArrayList<Candidates> candidates using party.getCandidates().

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	assertEquals is called on test data	candidates.size()	3	3	test is run to check if all the candidates are added correctly
2	assertEquals is called on test data	candidates.get(0).getName()	"test1"	"test1"	the 0th index would be the candidate with the name test1 because the candidate has 2 votes
3	assertEquals is called on test data	candidates.get(1).getName()	"test3"	"test3"	the first index would be the candidate with the name test3 because the candidate has 1 vote
4	assertEquals is called on test data	candidates.get(2).getName()	"test2"	"test2"	the second index would be the candidate with the name test2 because the candidate has 0 votes

Post condition(s) for Test:
no post conditions

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 023

Name(s) of Testers: Varun Reddy

Test Description: Tests functionality of the greaterRemain() function in the Party class which compares the remainder attribute of two different party classes.

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

Automated: yes__+_ no ___

Results: Pass _+____ Fail_____

Preconditions for Test:
Two Party objects are created with names "Test1" and "Test2". Party1.setRemainder(3) is called, and Party2.setRemainder(4) is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	assertEquals is called on the test data	party2.greaterRemain(party1, true)	1	1	
2	assertEquals is called on the test data	party1.greaterRemain(party2, true)	-1	-1	

Post condition(s) for Test: no postconditions

Project Name: Project 1: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 024

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with an OPL election with 377 votes

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

ElectionTest.java
testMediumOPL()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Correctly formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This ensures that only the correct candidates got elected. It does this by going through each party and seeing who got nominated.	electionTest.getParties()	PikeWinner && JonesWinner && FosterWinner && BorgWinner && valid	PikeWinner && JonesWinner && FosterWinner && BorgWinner && valid	the "Valid" check is set to false if anyone other than Pike, Jones, Smit, or Borg was elected
2	This test ensures that the correct amount of candidates were elected for this election	electionTest.getParties()	4	4	

Post condition(s) for Test:

An audit and media file are made for the election

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 025

Name(s) of Testers: Varun Reddy

Test Description: Tests functionality of the greaterRemain() function in the Party class which compares the remainder attribute of two different party classes when there is a tie in the remain

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

Automated: yes_+__ no ___

Results: Pass _+____ Fail_____

Preconditions for Test:
Two Party objects are created with names "Test1" and "Test2". Party1.setRemainder(3) is called, and Party2.setRemainder(3) is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	assertEquals is called on the test data	party2.greaterRemain(party1, true)	0	0	

Post condition(s) for Test: No postconditions

Test Stage: Unit _+_ System __ **Test Date:** 04/29/2021

Test Case ID#: 026 **Name(s) of Testers:** Varun Reddy

Test Description: Tests the openFile() method in Main

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

Automated: yes__+_ no ___

Results: Pass ___+__ Fail_____

Preconditions for Test: no preconditions since Main is static

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call assertNull on test data	Main.openFile("Doesnt Exist")	Null	Null	Assert looks if Null is returned when the filename given doesn't have .csv at the end

2	Call assertNull on test data	Main.openFile("Doesn't Exist.csv")	Null	Null	Assert looks if Null is returned when the file is not found
3	Call assertNull on test data	Main.openFile("csv")	Null	Null	Assert looks if Null is returned when the file name given is too short
4	Call assertNotEquals on test data	Main.openFile("testing/testCSVs/smallIR.csv")	!Null	!Null	Assert looks if !Null is returned when the file name is correct.

Post condition(s) for Test:

Project Name: Project 1: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 027

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with an OPL election with 100 votes and a tie within a party ranking

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

ElectionTest.java
testMediumTieOPL()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Correctly formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it. For this test specifically, the election should contain a tie within the party rankings.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This ensures that only the correct candidates got elected. It does this by going through each party and seeing who got nominated. For this test either Foster or Johnson will get elected depending on the random output.	electionTest.getParties()	JonesWinner && FosterOrJohnsonWinner && BorgWinner && SmithWinner && valid	JonesWinner && FosterOrJohnsonWinner && BorgWinner && SmithWinner && valid	the "Valid" check is set to false if anyone other than Borg, Jones, Smith, Foster or Johnson was elected
2	This test ensures that the correct amount of candidates were elected for this election	electionTest.getParties()	4	4	

Post condition(s) for Test:

An audit and media file are made for the election

Project Name: Project 1: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 028

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with an OPL election with 10000 votes

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

ElectionTest.java
testLargeOPL()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Correctly formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it.

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

1	This ensures that only the correct candidates got elected. It does this by going through each party and seeing who got nominated.	electionTest.getParties()	JohnsonWinner && JonesWinner && FosterWinner && DeutschWinner && valid	JohnsonWinner && JonesWinner && FosterWinner && DeutschWinner && valid	the "Valid" check is set to false if anyone other than Johnson, Jones, Foster, or Deutsch was elected
2	This test ensures that the correct amount of candidates were elected for this election	electionTest.getParties()	4	4	

Post condition(s) for Test:

An audit and media file are made for the election

Project Name: Project 1: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 029

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with an OPL election with 100000 votes

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

ElectionTest.java
testHugeOPL()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Correctly formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This ensures that only the correct candidates got elected. It does this by going through each party and seeing who got nominated.	electionTest.getParties()	SmithWinner && FosterWinner && DeutschWinner && valid	SmithWinner && FosterWinner && DeutschWinner && valid	the "Valid" check is set to false if anyone other than Smith, Foster, or Deutsch was elected
2	This test ensures that the correct amount of candidates were elected for this election	electionTest.getParties()	3	3	

Post condition(s) for Test:

An audit and media file are made for the election

Project Name: Project 1: Voting System

Test Stage: Unit __ System _+_ Test Date: 04/29/21

Test Case ID#: 030

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with an OPL election with 100000 votes

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

ElectionTest.java
testPartyTieOPL()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: Correctly formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it. This specific test has a tie between two parties.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This ensures that only the correct candidates got elected. It does this by going through each party and seeing who got nominated. This test has a tie between two parties so either Pike or Deutsche will be elected.	electionTest.getParties()	SmithWinner && (PikeWinner || DeutschWinner) && valid	SmithWinner && (PikeWinner || DeutschWinner) && valid	the "Valid" check is set to false if anyone other than Smith, Pike, or Deutsch was elected

2	**This test ensures that the correct amount of candidates were elected for this election**	**electionTest.getParties()**	**2**	**2**	

Post condition(s) for Test:

An audit and media file are made for the election

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 031

Name(s) of Testers: Varun

Test Description: Tests the eliminate() function in the Candidate class which eliminates a candidate from the election.

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

Automated: yes_+__ no ___

Results: Pass ___+__ Fail_____

Preconditions for Test: A candidate object is initialized with new Candidate("Bob"). testCandidate.eliminate() is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	assertEquals is called on test data	testCandidate.getNumVotes()	-1	-1	In an IR election once a candidate gets eliminated their vote count is set to -1.
2	assertTrue is called on test data	testCandidate.isEliminated()	true	true	Checks if the eliminated attribute is set to true

Post condition(s) for Test:

Test Stage: Unit ___ System _+__ **Test Date:** 04/29/21

Test Case ID#: 032

Test Description: Testing the system for an IR election with 9 ballots. In this test the three candidates all have the same amount of votes and therefore any one of them can win.

Name(s) of Testers: Jonathan Leibovich

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

Automated: yes _+__ no ___

Results: Pass ___+___ Fail _____

Preconditions for Test: A correctly formatted csv file containing the election information

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

1	Call assertEquals on the candidate array at the end of the election to see if the correct candidate won. In this case, any one of the candidates could possibly win since they all have the same amount of votes.	electionTest.getCandidates()	(RosenWinner||KleinbergWinner||ChouWinner) &&valid	(RosenWinner||KleinbergWinner||ChouWinner)&&valid	Valid ensures that only Rosen, Kleinberger or Chou won the election
2	Ensures that only one winner was was selected	electionTest.getCandidates()	1	1	

Post condition(s) for Test:

An audit and media file are made for the election

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 033

Name(s) of Testers: Varun Reddy

Test Description: Tests the setRemainder() method which sets the remainder attribute for the party class.

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

Automated: yes_+__ no ___

Results: Pass ___+__ Fail_____

Preconditions for Test: Party testParty is initialized with new Party("test"). testParty.setRemainder(3) is called; next testParty.setRemainder(4) is called.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	assertEquals is called on the test data	testParty.getRemainder( )	4	4	

Post condition(s) for Test: no post conditions for unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 034

Name(s) of Testers: Jonathan Leibovich

Test Description: Tests the POBallot() Constructor to ensure that the candidate index is initialized correctly

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

Automated: yes_+__ no ___

Results: Pass ___+__ Fail_____

Preconditions for Test: Two PO ballots were initialized with ID's of 123 and 456 respectively and candidate indexes of 0 and 2 respectively.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	assertEquals is called on the first ballots candidate index	testBallot1.getCandidate Index()	0	0	
2	assertEquals is called on the first ballots ID	testBallot1.getID()	123	123	
3	assertEquals is called on the first ballots candidate index	testBallot2.getCandidate Index()	2	2	
4	assertEquals is called on the second ballots ID	testBallot2.getID()	456	456	

Post condition(s) for Test: no post conditions for unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 035

Name(s) of Testers: Jonathan Leibovich

Test Description: Tests the Invalidate() method works correctly for IR Ballots to ensure that the candidate index is initialized correctly

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

Automated: yes_+__ no ___

Results: Pass ___+__ Fail_____

Preconditions for Test: Two IR ballots were initialized with to have initial candidate indexes of 0 and 1 respectively.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	assertEquals is called on the first ballots candidate index	testBallot.getCandidateIndex()	0	0	
2	assertEquals is called on the first ballots candidate index after it has been invalidated	testBallot.getCandidateIndex()	-1	-1	invalidate() was called in between steps 1 and 2
3	assertEquals is called on the second ballots candidate index after it has been invalidated	testBallot2.getCandidateIndex()	-1	-1	

Post condition(s) for Test: no post condition since it is an individual unit test

Test Stage: Unit _+_ System __

Test Date: 04/29/2021

Test Case ID#: 036

Name(s) of Testers: Jonathan Leibovich

Test Description: Tests the getNumCandidatesRanked() function in IR ballot to ensure it returns the correct number of candidates that were ranked in a given ballot

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

Automated: yes_+__ no ___

Results: Pass ___+__ Fail_____

Preconditions for Test: Two IR ballots were initialized with to have initial candidate indexes of 4 and 1 candidates ranked respectively.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	assertEquals is called on the first ballots number of candidates ranked	testBallot.getNumCandidatesRanked()	4	4	

2	assertEquals is called on the second ballots number of candidates ranked	testBallot.getNumCandidatesRanked()	1	1	

Post condition(s) for Test: no post condition since it is an individual unit test

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 037

Name(s) of Testers: Tanner Skluzacek

Test Description: Testing the system with a 3 small IR csv files being passed in

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

ElectionTest.java
testMultipleSmallIR()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: 3 correctly formatted csv file containing the IR ballot information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step goes through all the candidates and that the name matches the expected winner of a candidate that is not eliminated	electionTest.getCandidates()	name.equals("Kleinberg (R)) is true	name.equals("Kleinberg (R)) is true	
2	This test ensures that only one candidate is not eliminated	electionTest.getCandidates()	1	1	

Post condition(s) for Test:

An audit and media file are made for the election

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 038

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with a small PO csv file passed in (10 votes)

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

ElectionTest.java
testSmallPO()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: 1 correctly formatted csv file containing the PO ballot information

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step ensures that all of the candidates from the PO file were loaded into the election	electionTest.getCandidates()	candidateKennedy && candidateTruman && candidateBush && candidateRubio && valid	candidateKennedy && candidateTruman && candidateBush && candidateRubio && valid	the "Valid" check is set to false if anyone other than Kennedy, Truman, or Rubio, or Bush was in the candidate array
2	This step ensures that all of the ballots from the PO file were loaded into the election	electionTest.getBallots().size	10	10	

Post condition(s) for Test: All of the candidates have been initialized and the ballots have been initialized correctly

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 039

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with another small PO csv file passed in (15 votes). This test creates a three way tie

if the algorithm to calculate the result of PO elections is put into place

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

ElectionTest.java
testSmallPO2()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: 1 correctly formatted csv file containing the PO ballot information

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step ensures that all of the candidates from the PO file were loaded into the election	electionTest.getCandidates()	candidateKennedy && candidateTruman && candidateBush && candidateRubio && valid	candidateKennedy && candidateTruman && candidateBush && candidateRubio && valid	the "Valid" check is set to false if anyone other than Kennedy, Truman, or Rubio, or Bush was in the candidate array
2	This step ensures that all of the ballots from the PO file were loaded into the election	electionTest.getBallots().size	15	15	

Post condition(s) for Test: All of the candidates have been initialized and the ballots have been initialized correctly

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 040

Name(s) of Testers: Tanner Skluzacek

Test Description: Testing the system with a small IR file and a medium IR file together

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

ElectionTest.java
testMultipleSmallandMediumIR()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: both files are formatted csv file containing the IR ballot information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step goes through all the candidates and that the name matches the expected winner of a candidate that is not eliminated	electionTest.getCandidates()	name.equals("Kleinberg (R)) or name.equals("Chou (I)") is true	name.equals("Kleinberg (R)) or name.equals("Chou (I)") is true	

2	**This test ensures that only one candidate is not eliminated**	**electionTest.getCandidates()**	**1**	**1**	

Post condition(s) for Test:

An audit and media file are made for the election

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 041

Name(s) of Testers: Tanner Skluzacek

Test Description: Testing the system with two medium OPL files together

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

ElectionTest.java
testMultipleMediumOPL()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: both files are formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step goes through all parties and seats allocated to check names of who is elected	electionTest.getParties()	FosterWinner && JohnsonWinner && JonesWinner && BorgWinner && valid is true	FosterWinner && JohnsonWinner && JonesWinner && BorgWinner && valid is true	valid is to make sure that only the expected candidates are present
2	This test ensures that the correct number of candidates were elected for the OPL election	electionTest.getParties()	4	4	

Post condition(s) for Test:

An audit and media file are made for the election

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 042

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with a medium PO csv file passed in (300 votes).

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

ElectionTest.java
testMediumPO()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: 1 correctly formatted csv file containing the PO ballot information

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step ensures that all of the candidates from the PO file were loaded into the election	electionTest.getCandidates()	candidateKennedy && candidateTruman && candidateBush && candidateRubio && candidateSmith && valid	candidateKennedy && candidateTruman && candidateBush && candidateRubio && candidateSmith && valid	the "Valid" check is set to false if anyone other than Kennedy, Truman, or Rubio, Smith, or Bush was in the candidate array
2	This step ensures that all of the ballots from the PO file were loaded into the election	electionTest.getBallots().size	300	300	

Post condition(s) for Test: All of the candidates have been initialized and the ballots have been initialized correctly

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 043

Name(s) of Testers: Tanner Skluzacek

Test Description: Testing the system with one small and one very large OPL file

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

ElectionTest.java
testMultipleHugeSmallOPL()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: both files are formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step goes through all parties and seats allocated to check names of who is elected	electionTest.getParties()	FosterWinner && SmithWinner && (JonesWinner || DeutschWinner) && valid is true	FosterWinner && SmithWinner && (JonesWinner || DeutschWinner) && valid is true	valid is to make sure that only the expected candidates are present
2	This test ensures that the correct number of candidates were elected for the OPL election	electionTest.getParties()	3	3	

Post condition(s) for Test:

An audit and media file are made for the election

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 044

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with a large PO csv file passed in (1000 votes).

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

ElectionTest.java
testLargePO()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: 1 correctly formatted csv file containing the PO ballot information

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step ensures that all of the candidates from the PO file were loaded into the election	electionTest.getCandidates()	candidateKennedy && candidateTruman && candidateBush && candidateRubio && candidateSmith && valid	candidateKennedy && candidateTruman && candidateBush && candidateRubio && candidateSmith && valid	the "Valid" check is set to false if anyone other than Kennedy, Truman, or Rubio, Smith, or Bush was in the candidate array
2	This step ensures that all of the ballots from the PO	electionTest.getBallots().size	10000	10000	

	file were loaded into the election				

Post condition(s) for Test: All of the candidates have been initialized and the ballots have been initialized correctly

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 045

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with a huge PO csv file passed in (100000 votes).

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

ElectionTest.java
testHugePO()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: 1 correctly formatted csv file containing the PO ballot information

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

1	This step ensures that all of the candidates from the PO file were loaded into the election	electionTest.getCandidates()	candidateKennedy && candidateTruman && candidateBush && candidateRubio && candidateSmith && valid	candidateKennedy && candidateTruman && candidateBush && candidateRubio && candidateSmith && valid	the "Valid" check is set to false if anyone other than Kennedy, Truman, or Rubio, Smith, or Bush was in the candidate array
2	This step ensures that all of the ballots from the PO file were loaded into the election	electionTest.getBallots().size	100000	100000	

Post condition(s) for Test: All of the candidates have been initialized and the ballots have been initialized correctly

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 046

Name(s) of Testers: Tanner Skluzacek

Test Description: Testing the system with 5 different OPL election files

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

ElectionTest.java
testMultipleElection1()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: all 5 files are formatted csv file containing the OPL ballot information and correctly instantiated Election object. This object must have runElection called upon it.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step goes through all parties and seats allocated to check names of who is elected	electionTest.getParties()	KennedyWinner && BushWinner && RubioWinner && valid is true	KennedyWinner && BushWinner && RubioWinner && valid is true	valid is to make sure that only the expected candidates are present
2	This test ensures that the correct number of candidates were elected for the OPL election	electionTest.getParties()	3	3	

Post condition(s) for Test:

An audit and media file are made for the election

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 047

Name(s) of Testers: Jonathan Leibovich

Test Description: Testing the system with 2 different PO election files

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

ElectionTest.java
testMultiplePO()

Automated: yes_+__ no ___

Results: Pass __+___ Fail_____

Preconditions for Test: 2 correctly formatted csv file containing the PO ballot information

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	This step ensures that all of the candidates from the PO file were loaded into the election	electionTest.getCandidates()	candidateKennedy && candidateTruman && candidateBush && candidateRubio && valid	candidateKennedy && candidateTruman && candidateBush && candidateRubio && valid	the "Valid" check is set to false if anyone other than Kennedy, Truman, or Rubio, or Bush was in the candidate array
2	This step ensures that all of the ballots from the PO file were loaded into the election	electionTest.getBallots().size	25	25	

Post condition(s) for Test: All of the candidates have been initialized and the ballots have been initialized correctly

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 048

Name(s) of Testers: Varun Reddy

Test Description: Testing the system when there is a 3 way tie for elimination in an IR election to make sure the probability is even. This test was run using a program that runs an OPL election 1000 times. Since the coin flip is based on probability, this test has a small chance of failing, therefore, it cannot be automated. The program used is commented out at the bottom of ElectionTest.java.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
This test was run manually by giving the system the file multiTieIR.csv

Automated: yes___ no _+__

Results: Pass __+___ Fail_____

Preconditions for Test:
1 correctly formatted .csv file containing more than a two way tie for elimination.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call AssertTrue on test data,	.2<(RosenWinner)&& (RosenWinner)<.4&& .2<(ChouWinner)&& (ChouWinner)<.4&&	true	True	RosenWinner, KleinbergWinner, and ChouWinner represent the percentage of wins each

		.2<(KleinbergWinner)& & (KleinbergWinner)<.4			candidate had over 1000 elections

Post condition(s) for Test:

Project Name: Project 2: Voting System

Test Stage: Unit __ System _+_

Test Date: 04/29/21

Test Case ID#: 048

Name(s) of Testers: Varun Reddy

Test Description: Testing the system when there is more than a 2-way tie for the winner of a seat between multiple parties in a OPL. This test was run using a program that runs an OPL election 1000 times. Since the coin flip is based on probability, this test has a small chance of failing, therefore, it cannot be automated. The program used is commented out at the bottom of ElectionTest.java.

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

This test was run manually by giving the system the file multiPartyTieOPL.csv. A test was used to run the election multiple times.

Automated: yes___ no _+__

Results: Pass __+___ Fail_____

Preconditions for Test:

1 correctly formatted .csv file containing more than a two way tie for elimination.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call AssertTrue on test data,	.2<(DeutschWinner)&& (DeutschWinner)<.4&& .2<(DeutschWinner)&& (PikeWinner)<.4&& .2<(SmithWinner)&& (SmithWinner)<.4	true	True	DeutschWinner, DeutschWinner, and PikeWinner represent the percentage of wins each candidate had over 1000 elections

Post condition(s) for Test: