Maria Zavala - Zaval054
Aaron Raines - Raine124
Ethan Johnson - joh18919

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_1	**Name(s) of Testers: Maria Zavala, Cam**
Test Description: Run an election file that has a clear majority winner with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionClearWinner1()
Results: Pass ___X__ Fail_____	
Preconditions for Test: - The election file must have a clear majority winner without any redistribution of votes needed. - Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTestFile1.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Assert that the expected winners name matches return from runElection().	“Rosen” and winner.getName()	“Rosen”	“Rosen”	

Post condition(s) for Test:
Must have the name of the winner who has the majority of votes.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_2	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner in the middle of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionClearWinner2()
Results: Pass ___X__ Fail_____	

Preconditions for Test:

- The election file must have a clear majority winner without any redistribution of votes needed.
- Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTestFile2.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Assert that the expected winners name matches return from runElection().	"Kleinberg" and winner.getName()	"Kleinberg"	"Kleinberg"	

Post condition(s) for Test:

Must have the name of the winner who has the majority of votes.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_3	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner in the end of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionClearWinner3()
Results: Pass __X__ Fail_____	
Preconditions for Test: - The election file must have a clear majority winner without any redistribution of votes needed. - Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTestFile3.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Assert that the expected winners name matches return from runElection().	"Royce" and winner.getName()	"Royce"	"Royce"	

Post condition(s) for Test:
Must have the name of the winner who has the majority of votes.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_4	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear winner after multiple redistributions of votes between tied dropped candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionTieLowest1()
Results: Pass __X___ Fail_____	

Preconditions for Test:

- The election file must have a clear majority winner after a many redistributions of votes from dropping tied candidates.
- Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTieTestFile3.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Assert that the expected winners name matches return from runElection().	"Kleinberg" and winner.getName()	"Kleinberg"	"Kleinberg"	

Post condition(s) for Test:

Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_5	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear winner after a tie and the dropped candidate is determined by a coin flip, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionTieLowest2()
Results: Pass ___X__ Fail_____	

Preconditions for Test:

- The election file must have a clear majority winner after a tie and a coin flip determines what candidate gets dropped.
- Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTieTestFile4.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Make an if statement to see which of two candidates gets chosen as winner after a tie.	winner.getName()	"Kleinberg" or not “Kleinberg”	"Kleinberg" or not “Kleinberg”	
6	Assert that the expected winners name matches return from runElection().	name and winner.getName()	"Kleinberg" or “Rosen”	"Kleinberg" or “Rosen”	

Post condition(s) for Test:

Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_6	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner at the end of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionRedistributeWinner1()
Results: Pass __X___ Fail_____	

Preconditions for Test:

- The election file must have tied candidates to redistribute ballots.
- Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTieTestFile1.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Assert that the expected winners name matches return from runElection().	"Kleinberg" and winner.getName()	"Kleinberg"	"Kleinberg"	

Post condition(s) for Test:

Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_7	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner at the end of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionRedistributeWinner2()
Results: Pass __X___ Fail_____	
Preconditions for Test: - The election file must have tied candidates to redistribute ballots. - Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTieTestFile2.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Assert that the expected winners name matches return from runElection().	"Chou" and winner.getName()	"Chou"	"Chou"	

Post condition(s) for Test:
Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_8	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner at the end of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionRedistributeWinner3()
Results: Pass __X___ Fail_____	
Preconditions for Test: - The election file must have tied candidates to redistribute ballots. - Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTieTestFile6.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Assert that the expected winners name matches return from runElection().	"Chou" and winner.getName()	"Chou"	java.lang.IndexOutOfBoundsException: Index 0 out of bounds for length 0	This is not passing because lowestCandidates List is empty

Post condition(s) for Test:
Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/23/2023**
Test Case ID#: IR_Test_9	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner at the end of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void runElectionTieWinners1()
Results: Pass ___X__ Fail____	
Preconditions for Test: - The election file must have tied candidates to redistribute ballots of two winning candidates. - Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTieTestFile5.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	Assert that the expected winners name matches return from runElection().	"Chou" and winner.getName()	"Chou" or “Kleinberg”	java.lang.NullPointerException: Cannot read the array length because "<local4>" is null	This is not passing because Every candidate must have atleast 1 vote at beginning of run

Post condition(s) for Test:
Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: 3/23/2023**
Test Case ID#: Drop_Candidate_1	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner at the end of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	**Indicate where are you storing the tests (what file) and the name of the method/functions being used.** **File: test/**InstantRunoffTest **Method:** void dropCandidate1()
Results: Pass __X__ Fail_____	
Preconditions for Test: - We must have election that has a tie to drop a candidate. - Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTieTestFile1.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	run the getDroppedCanddiates() method and save the name of the dropped candidate.	ir object	Recieve name of candidates who where dropped.	Recieve name of candidates who where dropped.	
6	Assert that the expected winners name matches return from runElection().	"Rosen" and winner.getName()	"Rosen"	“Rosen”	

Post condition(s) for Test:
Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: 3/23/2023**
Test Case ID#: Distribute_Ballot_1	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner at the end of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void distrbuteBallot1()
Results: Pass __X__ Fail_____	
Preconditions for Test: - We must have an auditFile with information of an election. - Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	CoinFlip.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the getCandidates() method and save into candidates	auditFile Object	Receive a Candidate array	Recieved a Candidate array	
5	Get the ballots that candidate has currently.	candidate object	Recieve array of ballots that candidate has	Recieve array of ballots that candidate has	
6	Run the distributeBallot() method.	ir Object	Distribute ballots to a candidate.	Distribute ballots to a candidate.	
7	Assert that the ballots befofre distribution does not equal the distribution of ballots after running the previous method.	beforeDistribution and afterDistrbution ballots	Not equal to each other	Not equal to each other	

Post condition(s) for Test:
Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: 3/23/2023**
Test Case ID#: Coin_Flip_1	**Name(s) of Testers: Maria Zavala**
Test Description: Run coinflip method and make sure it does not select same candidate 100% of time.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** coinFlip1()
Results: Pass ___X__ Fail_____	

Preconditions for Test:

- There must be more than 1 candidate to pick from.
- Must be run at least 100 times to test randomization.
- Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	CoinFlip.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the getCandidates() method and save into candidates	auditFile Object	Receive a Candidate List	Recieved a Candidate List	
5	Run a the coinFilp() method in a for loop 100 times.	candidateList	Recieve a candidate	Recieve a candidate	
6	If candidate name equals to “Rosen” increment count variable	candidate.getName()	count incrememts	count incrememts	
7	Assert that count does not queal 100.	100 and count	Not equal to each other	Not equal to each other	

Post condition(s) for Test:

Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: 3/23/2023**
Test Case ID#: Get_Dropped_Candidate_1	**Name(s) of Testers: Maria Zavala**
Test Description: Run an election file that has a clear majority winner at the end of all possible candidates, run with the IR algorithm and determine if the correct winner is determined.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**InstantRunoffTest **Method:** void getDroppedCandidates1()
Results: Pass __X__ Fail_____	

Preconditions for Test:

- The election file to run must have a tie in order to drop a candidate.
- Work by uncommenting line in AuidFile… name = System.getProperty("user.dir") + "/testing/IRtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR IR

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	IRTieTestFile1.txt	Recieve a AuditFile object	Recieved a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create InstantRunoff object	auditFile object	Recieve a InstantRunoff obejct	Recieved a InstantRunoff obejct	
4	run the runElection() method and save result into winner.	ir object	Receive a Candidate	Recieved a Candidate	
5	run the getDroppedCanddiates() method and save the name of the dropped candidate.	ir object	Recieve name of candidates who where dropped.	Recieve name of candidates who where dropped.	
6	Assert that the expected winners name matches return from runElection().	"Rosen" and winner.getName()	"Rosen"	“Rosen”	

Post condition(s) for Test:

Must have the name of the winner who has the majority of votes after a Tie.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: March 23, 2023**
Test Case ID#: Ballot_Test_1	**Name(s) of Testers: Aaron Raines**
Test Description: Test that the Ballot class correctly stores and retrieves a list of votes.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**BallotTest **Method:** testGetVotes(), testIndexOf()
Results: Pass __X__ Fail_____	
Preconditions for Test: - Ballot class has been properly implemented and compiled.	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Test the getVotes() method of the Ballot class.	An ArrayList of integers (1, 2, 3) is passed to the Ballot constructor.	The getVotes() method should return an ArrayList containing the values passed to the constructor.	The getVotes() method returned the expected ArrayList..	
2	Test the indexOf() method of the Ballot class with a valid input value.	An ArrayList of integers (1, 2, 3) is passed to the Ballot constructor, and the indexOf() method is called with a value of 1.	The indexOf() method should return 0	The indexOf() method returned the expected value of 0.	
3	Test the indexOf() method of the Ballot class with an invalid input value.	An ArrayList of integers (1, 2, 3) is passed to the Ballot constructor, and the indexOf() method is called with a value of 4..	The indexOf() method should return -1.	The indexOf() method returned the expected value of -1.	

Post condition(s) for Test: None

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: March 23, 2023**
Test Case ID#: Candidate_Test_1	**Name(s) of Testers: Aaron Raines**
Test Description: This test is designed to verify the correctness of the Candidate class methods.	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/**CandidateTest **Method:** testGettersAndSetters, testPassBallot
Results: Pass __X___ Fail_____	
Preconditions for Test: The Candidate class must be implemented and imported correctly.	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Party object.	Name: "Test Party"	The Party object to be created.	The Party object is created.	
2	Create a Candidate object.	Name: "Test Candidate"; Party: party from step #1.	The Candidate object is created.	The Candidate object is created.	
3	Set the CandidateID of the Candidate object.	CandidateID: 123	The CandidateID to be set to 123.	The CandidateID is set to 123.	
4	Set the NumVotes of the Candidate object.	NumVotes: 456	The NumVotes to be set to 456.	The NumVotes is set to 456.	
5	Verify the correctness of getName() method of the Candidate object.	None.	The getName() method should return "Test Candidate".	The getName() method returns "Test Candidate".	
6	Verify the correctness of getParty() method of the Candidate object.	None.	The getParty() method should return the Party object from step #1.	The getParty() method returns the Party object from step #1.	
7	Verify the correctness of getCandidateID() method of the Candidate object.	None.	The getCandidateID() method should return 123.	The getCandidateID() method returns 123.	

8	Verify the correctness of getNumVotes() method of the Candidate object.	None.	The getNumVotes() method should return 456.	The getNumVotes() method returns 456.	
9	Create a Ballot object.	Votes: 1, 2, 3	The Ballot object to be created.	The Ballot object is created.	
10	Create another Ballot object.	Votes: 4, 5, 6	The Ballot object to be created.	The Ballot object is created.	
11	Pass the Ballot object from step #9 to the Candidate object.	None.	The Ballot object should be passed to the Candidate object.	The Ballot object is passed to the Candidate object.	
12	Pass the Ballot object from step #10 to the Candidate object.	None.	The Ballot object should be passed to the Candidate object.	The Ballot object is passed to the Candidate object.	
13	Verify the correctness of getBallots() method of the Candidate object.	None.	The getBallots() method should return an array of Ballot objects, which contains ballots from step #11 and #12.	The getBallots() method returns an array of Ballot objects, which contains ballots from step #11 and #12.	
14	Verify the correctness of getNumVotes() method of the Candidate object after passing the ballots.	None.	The getNumVotes() method should return the total number of votes received by the Candidate object.	The getNumVotes() method returns the total number of votes received by the Candidate object, which is 6 in this case.	
15	Verify the correctness of toString() method of the Candidate object.	None.	The toString() method returns a string representation of the Candidate object.	The toString() method returns a string representation of the Candidate object, which includes the name, party, candidateID, and number of votes received.	

Post condition(s) for Test: The state of the Candidate object may be changed, but the system state remains the same.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: March 23, 2023**
Test Case ID#: Party_Test_1	**Name(s) of Testers: Aaron Raines**
Test Description: This test is used to check the getters and setters of the Party class.	
Automated: yes_X_ no ___	**Indicate where are you storing the tests (what file) and the name of the method/functions being used.** **File: test**/PartyTest **Method:** testGettersAndSetters()
Results: Pass __X___ Fail_____	
Preconditions for Test: The Party class and its dependencies should be compiled without any errors.	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a Party object with the name "Test Party".	Name: "Test Party"	The Party object with the name "Test Party" should be] created.	The Party object with the name "Test Party" is created.	
2	Create two Candidate objects with names "Test Candidate 1" and "Test Candidate 2", respectively, and assign them to the Party object created in Step #1.	Candidate 1 Name: "Test Candidate 1"; Candidate 2 Name: "Test Candidate 2"; Party: party from step #1.	Two Candidate objects with the assigned names should be created and added to the Party object.	Two Candidate objects with the assigned names are created and added to the Party object.	
3	Create two Ballot objects with votes {1, 2, 3} and {4, 5, 6}, respectively, and assign them to the Party object created in Step #1.	Ballot 1 Votes: {1, 2, 3}; Ballot 2 Votes: {4, 5, 6}; Party: party from step #1.	Two Ballot objects with the assigned votes are created and added to the Party object.	Two Ballot objects with the assigned votes are created and added to the Party object.	
4	Set the array of Candidate objects from Step #2 to the Party object created in Step #1.	Candidates: candidates array from Step #2; Party: party from Step #1.	The array of Candidate objects should be set to the Party object.	The array of Candidate objects is set to the Party object.	

5	Set the array of Ballot objects from Step #3 to the Party object created in Step #1.	Ballots: ballots array from Step #3; Party: party from Step #1	The array of Ballot objects should be set to the Party object.	The array of Ballot objects is set to the Party object.	
6	Set the number of PVotes to 100 for the Party object created in Step #1.	Number of PVotes: 100; Party: party from Step #1.	The number of PVotes should be set to 100 for the Party object.	The number of PVotes is set to 100 for the Party object.	
7	Set the number of PSeats to 2 for the Party object created in Step #1.	Number of PSeats: 2; Party: party from Step #1.	The number of PSeats should be set to 2 for the Party object.	The number of PSeats is set to 2 for the Party object.	
8	Set the remaining votes to 50 for the Party object created in Step #1.	Remaining votes: 50; Party: party from Step #1.	The remaining number of votes should be set to 50 for the Party object.	The remaining number of votes is set to 50 for the Party object.	
9	Verify the correctness of the getpName() method of the Party object.	None.	The getpName() method should return "Test Party".	The getpName() method returns "Test Party".	
10	Verify the correctness of the getpCandidates() method of the Party object.	None.	The getpCandidates() method should return an array containing the two Candidate objects created in Step #2.	The getpCandidates() method returns an array containing the two Candidate objects created in Step #2.	

Post condition(s) for Test: The getters and setters of the Party class should be functioning properly without any errors.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/24/2023**
Test Case ID#: CPL_test1	**Name(s) of Testers: Ethan Johnson**
Test Description: Tests that a CPL election with no remaining votes after first allocation works	
Automated: yes_X_ no ___	**Indicate where are you storing the tests (what file) and the name of the method/functions being used.** **File: test/ClosedPartyListTest** **Method:** - runElection() - runCplElectionClearWinner1
Results: Pass __X___ Fail_____	

Preconditions for Test:

- Have CPL file with ballots that will produce a winner with no remaining votes for any party and all seats will be allocated in the first allocation
- **Have "**name = System.getProperty("user.dir") + "/testing/CPLtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR CPL" **line in auditFile.java uncommented**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	CPLTestFile1.csv	Receive a AuditFile object	Received a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create ClosedPartyList object	auditFile object	Receive a ClosedPartyList object	Received a ClosedPartyList object	
4	run the runElection() method and save result into winners.	Cpl object	Receive a party list	Received a party list	
5	Assert that the expected party winners name matches return from runElection().	"Reform" and winners.getpName()	"Reform"	"Reform"	

Post condition(s) for Test:

Produce Party who won the seats of the election.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/24/2023**
Test Case ID#: CPL_test2	**Name(s) of Testers: Ethan Johnson**
Test Description: Run CPL election with remaining seats after first allocation of votes and then those seats are allocated during second allocation	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/ClosedPartyListTest** **Method:** - runElection() - RunCplElectionClearWinner2()
Results: Pass __X___ Fail_____	
Preconditions for Test: - **Have CPL file with ballots set so that there will be remaining seats after first allocation so there will be a second allocation** - **Have "**name = System.getProperty("user.dir") + "/testing/CPLtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR CPL" **line in auditFile.java uncommented**	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	CPLTestFile2.csv	Receive a AuditFile object	Received a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create ClosedPartyList object	auditFile object	Receive a ClosedPartyList object	Received a ClosedPartyList object	
4	run the runElection() method and save result into winners.	Cpl object	Receive a party list	Received a party list	
5	Assert that the expected party winners name matches return from runElection().	"Republican" and "Independent" and winners.getpName()	"Republican" And "Independent"	"Republican" And "Independent"	

Post condition(s) for Test:

Produce Parties who won the seats of the election.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit __ System _X_	**Test Date: 3/24/2023**
Test Case ID#: CPL_test3	**Name(s) of Testers: Ethan Johnson**
Test Description: Run CPL election with remaining seats after first allocation of votes and then those seats are allocated during second allocation but there is a tie and the winner is randomly decided	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/ClosedPartyListTest** **Method:** - runElection() - RunCplElectionClearWinner3
Results: Pass _____ Fail____X____	
Preconditions for Test: - **Have CPL file with ballots set so that there will be a tie** - **Have "**name = System.getProperty("user.dir") + "/testing/CPLtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR CPL" **line in auditFile.java uncommented**	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	CPLTestFile3.csv	Receive a AuditFile object	Received a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create ClosedPartyList object	auditFile object	Receive a ClosedPartyList object	Received a ClosedPartyList object	
4	run the runElection() method and save result into winners.	Cpl object	Receive a party list	Received a party list	
5	Assert that the expected party winners name matches return from runElection().	"Democratic" "New Wave" "Reform" "Independent" and winners.getpName()	"Democratic" And one of the following: "New Wave" "Reform" "Independent"	"Democratic" "New Wave"	This is noted on the bug list. CPL ties do not work currently.

Post condition(s) for Test:

Produce Parties who won the seats of the election.

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: 3/24/2023**
Test Case ID#: CPL_test4	**Name(s) of Testers: Ethan Johnson**
Test Description: Run assignSeats method to test if seats are being allocated to parties with no bugs	
Automated: yes_X_ no ___	Indicate where are you storing the tests (what file) and the name of the method/functions being used. **File: test/ClosedPartyListTest** **Method:** - runElection() - assignSeats()
Results: Pass __X___ Fail_____	

Preconditions for Test:
- **Have CPL file with no errors**
- **Have "**name = System.getProperty("user.dir") + "/testing/CPLtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR CPL" **line in auditFile.java uncommented**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	CPLTestFile3.csv	Receive a AuditFile object	Received a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create ClosedPartyList object	auditFile object	Receive a ClosedPartyList object	Received a ClosedPartyList object	
4	run the runElection() method and save result into winners.	Cpl object	Receive a party list	Received a party list	
5	Assert that the expected parties with assigned seats names match return from assignSeats().	"Reform" and winners.getpName()	"Reform"	"Reform"	

Post condition(s) for Test:
assignSeats returns leading party

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: 3/24/2023**
Test Case ID#: CPL_test5	**Name(s) of Testers: Ethan Johnson**
Test Description: Run remainingSeats method and have it return parties with their seats assigned	
Automated: yes_X_ no ___	**Indicate where are you storing the tests (what file) and the name of the method/functions being used.** **File: test/ClosedPartyListTest** **Method:** - runElection() - remainingSeats()
Results: Pass __X___ Fail_____	

Preconditions for Test:
- **Have CPL file with no errors**
- **Have "**name = System.getProperty("user.dir") + "/testing/CPLtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR CPL" **line in auditFile.java uncommented**

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	CPLTestFile3.csv	Receive a AuditFile object	Received a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create ClosedPartyList object	auditFile object	Receive a ClosedPartyList object	Received a ClosedPartyList object	
4	run the runElection() method and save result into winners.	Cpl object	Receive a party list	Received a party list	
5	Assert that the expected parties with assigned seats names match return from remainingSeats()	"Democratic" "New Wave" "Reform" "Independent" and winners.getpName()	"Democratic" And one of the following: "New Wave" "Reform" "Independent"	"Democratic" And one of the following: "New Wave" "Reform" "Independent"	

Post condition(s) for Test:
remainingSeats() returns list of parties with assigned seats

ProjectName: Project1:Voting System **Team# 5**

Test Stage: Unit _X_ System __	**Test Date: 3/24/2023**
Test Case ID#: CPL_test6	**Name(s) of Testers: Ethan Johnson**
Test Description: Run coinflip and make sure it is random by making sure return value is not always the same	
Automated: yes_X_ no ___	**Indicate where are you storing the tests (what file) and the name of the method/functions being used.** **File: test/ClosedPartyListTest** **Method:** - runElection() - coinFlip()
Results: Pass __X___ Fail_____	
Preconditions for Test: - **Have 2 different numbers for args** - **Have "**name = System.getProperty("user.dir") + "/testing/CPLtestFiles/" + name; //ONLY WORKS IN test\java FILES FOR CPL" **line in auditFile.java uncommented**	

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create AuditFile object	CPLTestFile3.csv	Receive a AuditFile object	Received a AuditFile object	
2	Run the collectFileInfo() method for auditfile object	auditFile object	auditFile has collected all info about the csv file	auditFile has collected all info about the csv file	
3	Create ClosedPartyList object	auditFile object	Receive a ClosedPartyList object	Received a ClosedPartyList object	
4	run the runElection() method and save result into winners.	Cpl object	Receive a party list	Received a party list	
5	Assert that the results of the coinflip function were not always the same	Ints 1 and 2	Count != 100	Count != 100	

Post condition(s) for Test:
Results from coinflip are not always the same therefore it is random.