Project Name: Project 1: Voting System **Team #9**

Test Stage: Unit ✓ System
Test Case ID#: FP_CreateAuditFile_1

Test Description: Verify that createAuditFile generates a file name with the correct timestamp.

Automated: yes no ✓

Test Date: November 13, 2023

Name(s) of Testers: Praful Das

Indicate where are you storing the tests (what file) and the name of the method/functions being used: Test is stored in FileProcessorTest.java which is in testing directory. The name of the function being used is testCreateAuditFile().

Results: Pass ✓ Fail_____

Preconditions for Test: The system clock is set to a known fixed time (2023-11-01T12:34:56.00Z) for consistent results.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Execute the createAuditFile	-	The createAuditFile method generates a file name with the correct time stamp	The createAuditFile method generates a file name with the correct time stamp	-

Post condition(s) for Test: File name is generated by createAuditFile method with correct timestamp.

Test Stage: Unit ✓ System
Test Case ID#: FP_ReadHeader_OPL_1

Test Description: Verify that the readHeader function initializes an OPLElection object correctly based on the input file data for OPL election.

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Praful Das

Indicate where are you storing the tests (what file) and the name of the method/functions being used: Test is stored in FileProcessorTest.java which is in testing directory. The name of the function being used is testReadHeaderOPL.

Results: Pass ✓ Fail_____

Preconditions for Test: The OPL election input file "OPLElection.csv" exists.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set the input file to "OPLTest.csv" using setInputFile.	"OPLTest.csv"	-	-	-
2	Run the readHeader function.	-	'true' (OPLElection object initialized)	'true'	-
3	Check if the returned ElectionType is an instance of OPLElection.	-	'true'	'true'	-
4	Verify OPLElection attributes: Candidates Count, Party List size, Ballot Count, and Number of Seats.	-	6, 3, 9, 0	6, 3, 9, 0	-
5	Validate the Candidate List for correct names and party assignments.	-	Pike, Foster, Deutsch, Borg, Jones, Smith	Pike, Foster, Deutsch, Borg, Jones, Smith	-

Post condition(s) for Test: The OPLElection attributes are correctly initialized, and the state of the system remains unchanged.

Test Stage: Unit ✓ System
Test Case ID#: FP_ReadHeader_IR_1

Test Description: Verify that the readHeader function initializes an IRElection object correctly based on the input file data for IR election.

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Praful Das

Indicate where are you storing the tests (what file) and the name of the method/functions being used: Test is stored in FileProcessorTest.java which is in testing directory. The name of

	the function being used is testReadHeaderOPL.
Results: Pass ✓ Fail_____	

Preconditions for Test: The IR election input file "IRElection.csv" exists.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set the input file to "IRTest.csv" using setInputFile.	"IRTest.csv"	-	-	-
2	Run the readHeader function.	-	'true' (IRElection object initialized)	'true'	-
3	Check if the returned ElectionType is an instance of IRElection.	-	'true'	'true'	-
4	Verify IRElection attributes: Candidates Count and Ballot Count.	-	4, 6	4, 6	-
5	Validate the Candidate List for correct names and party assignments.	-	Rosen, D, Kleinberg, R, Chou, I, Royce, L	Rosen, D, Kleinberg, R, Chou, I, Royce, L	-

Post condition(s) for Test: The IRElection attributes are correctly initialized, and the state of the system remains unchanged.

Test Stage: Unit ✓ System **Test Case ID#:** FP_CreateCandidatesParties_OPL_1 **Test Description:** Verify that the createCandidatesAndParties function sets up candidates and parties correctly based on the input data. **Automated: yes ✓ no ___**	**Test Date:** November 13, 2023 **Name(s) of Testers:** Praful Das **Indicate where are you storing the tests (what file) and the name of the method/functions being used:** Test is stored in FileProcessorTest.java which is in testing directory. The name of the function being used is testCreateCandidatesAndParties.

Results: Pass ✓ Fail_____

Preconditions for Test: An OPLElection object (oplelection) is instantiated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the createCandidatesAndParties function with the input line "Pike (D), Foster (D), Deutsch (R), Borg (R), Jones (R), Smith (I)".	"Pike (D), Foster (D), Deutsch (R), Borg (R), Jones (R), Smith (I)"	-	-	-
2	Check if candidates and parties are created correctly in the OPLElection object.	-	'true' (Candidates and parties are created)	'true'	-
3	Validate the number of candidates and parties created.	-	6 candidates, 3 parties	6 candidates, 3 parties	-
4	Validate details of specific candidates and parties.	-			-
5	Verify details for each candidate, including their party assignments.	-	Pike, D, Foster, D, Deutsch, R, Borg, R, Jones, R, Smith, I	Pike, D, Foster, D, Deutsch, R, Borg, R, Jones, R, Smith, I	-

Post condition(s) for Test: Candidates and parties are correctly set up, and the state of the OPLElection object remains unchanged.

Test Stage: Unit ✓ System
Test Case ID#: FP_ReadRemainingIR_1

Test Description: Verify that the readRemainingIR function processes the lines and updates the IRElection object correctly.

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Praful Das

Indicate where are you storing the tests (what file) and the name of the method/functions being used: Test is stored in FileProcessorTest.java which is in testing directory. The name of the function being used is testReadRemainingIR

Results: Pass ✓ Fail_____

Preconditions for Test: IRElection object (irelection) is instantiated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the readRemainingIR function with the input file "IRTest.csv".	"IRTest.csv"	-	-	-
2	Check if candidates and their ballots are updated correctly in the IRElection object.	-	true (Candidates and ballots are updated)	‘true’	-
3	Verify the number of ballots for each candidate.	-	Rosen: 3 ballots Kleinberg: 0 ballots (empty) Chou: 2 ballots Royce: 1 ballot	Rosen: 3 ballots Kleinberg: 0 ballots (empty) Chou: 2 ballots Royce: 1 ballot	-
4	Verify the content of each ballot for specific candidates.	-	Rosen's Ballots: "1,3,4,2" "1,2,3," ",,1,2" Chou's Ballots: "1,,2," "3,2,1,4" Royce's Ballots: ",,,1" Kleinberg's Ballots: [] (empty)	Rosen's Ballots: "1,3,4,2" "1,2,3," ",,1,2" Chou's Ballots: "1,,2," "3,2,1,4" Royce's Ballots: ",,,1" Kleinberg's Ballots: [] (empty)	-
5	Verify that the winner is correctly determined.	-	The winner is Rosen.	The winner is Rosen.	-

Post condition(s) for Test: Candidates and their ballots are correctly updated, and the state of the IRElection object remains unchanged.

Test Stage: Unit ✓ System
Test Case ID#: FP_ReadRemainingOPL_1

Test Description: Verify that the readRemainingOPL function processes the lines and updates the OPLElection object correctly.

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Praful Das

Indicate where are you storing the tests (what file) and the name of the method/functions being used: Test is stored in

FileProcessorTest.java which is in testing directory. The name of the function being used is testReadRemainingOPL

Results: Pass ✓ Fail_____

Preconditions for Test:
- An OPLElection object (oplelection) is instantiated.
- Candidates and parties are already added to the OPLElection object using readHeader (assumption).

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the readRemainingOPL function with the input file "OPLElection.csv".	"IRTest.csv"	-	-	-
2	Check if the OPLElection object is updated correctly.	-	'true'	'true'	-
3	Verify the number of candidates in the OPLElection object.	-	5 candidates	5 candidates	-
4	Verify the number of parties in the OPLElection object.	-	3 parties	3 parties	-
5	Validate the votes for each candidate in the OPLElection object.	-	Pike: 3 votes Foster: 2 votes Deutsch: 0 votes Borg: 2 votes Jones: 1 vote Smith: 1 vote	Pike: 3 votes Foster: 2 votes Deutsch: 0 votes Borg: 2 votes Jones: 1 vote Smith: 1 vote	-
6	Validate the votes for each party in the OPLElection object.	-	D: 5 votes R: 2 votes I: 2 votes	D: 5 votes R: 2 votes I: 2 votes	-

Post condition(s) for Test: Candidates and their ballots are correctly updated, and the state of the IRElection object remains unchanged.

Test Stage: Unit ✓ System **Test Date:** November 13, 2023

Test Case ID#: FP_WriteToAuditIR_1 **Name(s) of Testers:** Praful Das

Test Description: Verify if writeToAuditFileIR correctly writes IR election-related data to the audit file.

Automated: yes no ✓	**Indicate where are you storing the tests (what file) and the name of the method/functions being used:** N/A
Results: Pass ✓ Fail_____	

Preconditions for Test: - A .txt audit file exists

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call writeToAuditFileIR with specific IR election-related data. Test Data: Specific IR election data (e.g., candidate, ballot information)	Specific IR election data (e.g., candidate, ballot information)	A new line should be written to the audit file with IR election details.	Not able to write to audit correctly.	-

Post condition(s) for Test:
Audit file should have appended IR election details.

Test Stage: Unit ✓ System	**Test Date:** November 13, 2023
Test Case ID#: FP_WriteToAuditOPL_1 **Test Description:** Verify if writeToAuditFileOPL correctly writes OPL election-related data to the audit file.	**Name(s) of Testers:** Praful Das
Automated: yes no ✓	**Indicate where are you storing the tests (what file) and the name of the method/functions being used:** N/A
Results: Pass ✓ Fail____	

Preconditions for Test: - A .txt audit file exists

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

1	Call writeToAuditFileOPL with specific OPL election-related data.	OPL election data (e.g., party details, vote counts)	A new line should be written to the audit file with OPL election details.	A new line is written to the audit file with OPL election details.	-

Post condition(s) for Test:
Audit file should have appended IR election details.

Test Stage: **Unit ✓** **System**
Test Case ID#: FP_CloseAuditFile_1

Test Description: Verify that the closeAuditFile method correctly closes the writer to the audit file and sets the file to read-only.

Automated: **yes** **no ✓**

Test Date: November 13, 2023

Name(s) of Testers: Praful Das

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

Results: **Pass ✓** **Fail_____**

Preconditions for Test: An audit file should be open for writing.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	An audit file should be open for writingCall the closeAuditFile method, passing the writer to the audit file.	"audit.txt"	The writer to the audit file is successfully closed.	The writer to the audit file is successfully closed.	-
2	Attempt to write to the closed audit file.	-	Unsucessful	Unsucessful	-

Post condition(s) for Test: The audit file is closed, and its permissions are set to read-only.

Test Stage: **Unit ✓** **System**
Test Case ID#: Cand_RemoveCandidateFromElection_1

Test Description: Checks to ensure

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

removeCandidateFromElection() method works so when candidate is eliminated they can no longer be considered when ballots are being reshuffled

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

Automated: yes ✓ no

Results: Pass ✓ Fail_____

Preconditions for Test: Candidate needs to be created correctly and in the election

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create candidate object	Candidate can3 = new Candidate("can3")	-	-	-
2	call remove from election method	can3.removeCandidateFromElection()	-	-	-
3	check if still in election		false	false	

Post condition(s) for Test: candidate has been removed from election + no other attributes have changed

Test Stage: Unit ✓ System
Test Case ID#: Cand_RemoveBallot_1

Test Description: Checks to ensure removeBallot() takes out a ballot attributed to a particular candidate. This is used when candidate has been eliminated and all ballots attributed to that candidate need to be re-assigned to their next highest ranking candidate.

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

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

Automated: yes ✓ no ___

Results: Pass ✓ Fail_____

Preconditions for Test:
Candidate needs to be created correctly and ballot object needs to be created correctly. Ballot needs to be tied to candidate correctly so addToBallotList() needs to work correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create candidate object	Candidate can4 = new Candidate("can4")	-	-	-
2	create a ballot object	Ballot b1 = new Ballot()	-	-	-
3	assign ballot to candidate	can4.addToBallotList(b1);	-	-	-
4	remove ballot	can4.removeBallot();	-	-	-
5	check if ballot was removed	-	true	true	ballots will be removed sequentially (in the order they were read in and then assigned to candidate)

Post condition(s) for Test: ballot has been removed from that candidate's attributed list of ballots

Test Stage: Unit ✓ System
Test Case ID#: Cand_RemoveBallot_1

Test Description: Checks to ensure removeBallot() takes out a ballot attributed to a particular candidate. This is used when candidate has been eliminated and all ballots attributed to that candidate need to be re-assigned to their next highest ranking candidate.

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

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

Results: Pass ✓ Fail_____

Preconditions for Test:

Candidate needs to be created correctly and ballot object needs to be created correctly. Ballot needs to be tied to candidate correctly so addToBallotList() needs to work correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create candidate object	Candidate can4 = new Candidate("can4")	-	-	-
2	create a ballot object	Ballot b1 = new Ballot()	-	-	-
3	assign ballot to candidate	can4.addToBallotList(b1);	-	-	-
4	remove ballot	can4.removeBallot();	-	-	-
5	check if ballot was removed	-	true	true	ballots will be removed sequentially (in the order they were read in and then assigned to candidate)

Post condition(s) for Test: ballot has been removed from that candidate's attributed list of ballots

Test Stage: Unit ✓ System
Test Case ID#: Cand_ Constructor_1

Test Description: Check to see if Candidate object is being created correctly and default attributes are set accordingly.

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

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

Results: Pass ✓ Fail_____

Preconditions for Test:
ElectionType has to recognize Candidate object.

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

1	create an instance of candidate	Candidate can1 = new Candidate("test1");	Candiate@XXXXXXX	Candiate@XXXXXXX	-
2	check name	-	test1	test1	-
3	check ballotCount	-	0	0	-
4	check getCandidateParty	-	null	null	-
5	check getBallots()	-	true	True	-
6	check isInElection()	-	true	true	checks if list is empty with assertTrue()

Post condition(s) for Test: Candidate object is created correctly + all attributes are set

Test Stage: Unit ✓ System
Test Case ID#: Cand_AddToBallotList_1

Test Description: Checks to ensure addToBallotList() adds a ballot attributed to a particular candidate. This is used when candidate needs to be assigned all ballots that have ranked them first (or when other candidates are removed and ballots have to be reassigned)

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

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

Results: Pass ✓ Fail_____

Preconditions for Test:
Candidate needs to be created correctly and ballot object needs to be created correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create candidate object	Candidate can5 = new Candidate("can5")	-	-	-
2	create a ballot object	Ballot b1 = new Ballot()	-	-	-
3	assign ballot to candidate	can5.addToBallotList(b1);	-	-	checking the size of the list of ballots

4	check if ballot was added	-	1	1	-

Post condition(s) for Test: ballot has been tied to a candidate

Test Stage: Unit ✓ System
Test Case ID#: Cand_ AssignCandidateToParty_1

Test Description: Check to see if Candidate object is being assigned to a party object correctly

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

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

Results: Pass ✓ Fail_____

Preconditions for Test:
Candidate object is created correctly & Party object is created correctly

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create candidate object	Candidate can2 = new Candidate(“can2”);	-	-	-
2	Create Party object	Party party1 = new Party(“L”);	-	-	-
3	Assign candidate to party	can2.assignCandidateToParty(party1)	-	-	-
4	check assignment	-	L	L	-

Post condition(s) for Test: party attribution has been assigned in candidate + tied to a party object

Test Stage: Unit ✓ System
Test Case ID#: IR_ runElection_1

Test Description: Checks if correct winner is being determined from IR Election’s runElection() function. For this case, 4

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

Candidate + Party are created, along with 6 Ballots. Each of these is assigned to respectively to one another (basically doing what readHeader() and readRemainingIR() does). Then the winner of the election is checked. checkWinner1() poses an election with no ties, checkWinner2() has an election where some candidates have 0 votes, checkWinner3() tests an election with mutiple ties.

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
IRElectionTesting.java
checkWinner1(), checkWinner2(), checkWinner3()

Automated: yes ✓ no ___

Results: Pass Fail_____

Preconditions for Test:
- File has been passed in, header + ballots have been read, appropriate objects (Candidates, Ballots, Parties, IRElection) have been created.
- First stage of ballot distribution to Candidate objects has been done.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	call runElection() method	IRElection elec with variously filled in Candidates, Parties, and Ballots	Rosen	Rosen	-
2	call runElection() method	IRElection elec with variously filled in Candidates, Parties, and Ballots	Rosen	No Result - infinite loop	known bug
3	call runElection() method	IRElection elec with variously filled in Candidates, Parties, and Ballots	Klienberg	Klienberg	-

Post condition(s) for Test: Ballots attributed to the Candidate should be done correctly, dealing with tie breaker cases + any special circumstances correctly. The winner object should be passed back to the controller to determine that election has been competed and winner has been determined.

Notes: *checkWinner2() is a known bug - it doesn't pass right now and instead goes into an infinite loop because it is not hitting the tiebreaker case correctly

Test Stage: Unit ✓ System
Test Case ID#: IR_DisplayIRElectionResults_1

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

Test Description: Checks if correct winner is being determined from IR Election's runElection() function. For this case, 4 Candidate + Party are created, along with 6 Ballots. Each of these is assigned to respectively to one another (basically doing what readHeader() and readRemainingIR() does). Then the winner of the election is checked. checkWinner1() poses an election with no ties, checkWinner2() has an election where some candidates have 0 votes, checkWinner3() tests an election with mutiple ties.

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

Automated: yes ✓ no ___

Results: Pass ✓ Fail_____

Preconditions for Test:
Election needs to have run and completed successfully - runElection() had to complete correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to see if output is correctly printed and formatted.	Candidate, Party, Ballot objects configured to correctly to run an election.	R ELECTION RESULTS \nWINNER:Rosen\n Rosen:3, Klienberg:0,Chou:2,Royce:1	IR ELECTION RESULTS \nWINNER:Rosen\n Rosen:3, Klienberg:0,Chou:2,Royce:1	-

Post condition(s) for Test: No winner object or attributes should be modified/edited because of this display. Control is shifted back to controller object.

Test Stage: Unit ✓ System
Test Case ID#: IR_getInvalidatedBallots()_1

Test Date: November 13, 2023

Name(s) of Testers: Stuti Arora

Test Description: Checks to ensure ballots with all candidates who are eliminated from the election are stored in a list (which will be added to the audit file)

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

Automated: yes ✓ no ___

checkGetInvalidatedBallots()

Results: Pass Fail_____

Preconditions for Test:
- File has been passed in, header + ballots have been read, appropriate objects (Candidates, Ballots, Parties, IRElection) have been created.
- First stage of ballot distribution to Candidate objects has been done.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up a mock IR Election	Candidate, Party, Ballot objects configured to correctly to run an election.	-	-	-
2	Check to see if invalidated ballots has correct ballots		[1,2,,,],[,,,1]	[1,2,,]	known bug - last invalidated ballot is not being added to the list correctly

Post condition(s) for Test: Invalidated ballots need to be produced correctly & election should be completed.

Test Stage: Unit ✓ System
Test Case ID#: OPL_NumberOfSeats_1

Test Description: Verify the behavior of the OPLElection's setNumberOfSeats method.

Test Date: November 13, 2023

Name(s) of Testers: Neha Bhatia

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

Automated: yes ✓ no ___

Results: Pass ✓ Fail_____

Preconditions for Test:
- The OPLElection object (elec) is instantiated.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set the number of seats to 10 using setNumberOfSeats method.	Number of seats = 10	OPLElection's numberOfSeats should be set to 10.	OPLElection's numberOfSeats is 10.	-
2	Attempt to set the number of seats to -10 using setNumberOfSeats method.	Number of seats = -10	An IllegalArgumentException should be thrown with the message "Number of seats cannot be negative."	An IllegalArgumentException was thrown with the message "Number of seats cannot be negative."	-

Post condition(s) for Test:
1. After Step 1, the OPLElection's numberOfSeats attribute is set to 10.
2. After Step 2, the OPLElection's numberOfSeats remains unchanged.

Test Stage: **Unit** ✓ **System**
Test Case ID#: OPL_ runElection_1

Test Description: Test the runElection method of the OPLElection class.

Automated: **yes** ✓ **no** ___

Test Date: November 13, 2023

Name(s) of Testers: Neha Bhatia

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

Results: **Pass** ✓ **Fail**_____

Preconditions for Test:
- OPLElection, Party, and Candidate objects are instantiated.
- Each Party object has a list of Candidate objects, and each Candidate is associated with a Party.
- Vote counts for Parties and Candidates are predefined.
- The election type is set to OPL.
- The number of seats and the total ballot count for the election are set.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up OPLElection, Parties, and Candidates with predefined vote counts.	Predefined Party and Candidate data with specified vote counts.	Parties and Candidates are initialized with the following vote counts: - Party D: 4 votes - Party R: 6 votes - Party A: 9 votes - Candidate a: 2 votes - Candidate b: 2 votes - Candidate c: 3 votes - Candidate d: 1 vote - Candidate e: 1 vote - Candidate f: 5 votes	Parties and Candidates are initialized as expected..	-
2	Run the election using the runElection method.	-	Seats are assigned to Parties based on the election algorithm. Expected seat distribution: - Party D: 1 seat - Party R: 1 seat - Party A: 2 seats	Election runs successfully, and seats are assigned as expected..	-
3	Verify the election results.	-	Party seat counts match the expected results. - Party D: 1 seat - Party R: 1 seat - Party A: 2 seats	Party seat counts match the expected results.	-

Post condition(s) for Test:

- OPLElection, Party, and Candidate objects are in a consistent state after the election run.

- The runElection method has been executed successfully without errors.
- Seats have been assigned to Parties based on the election algorithm.
- Party seat counts have been updated based on the election results.
- The election results are verified, and Party seat counts match the expected results.

Test Stage: Unit ✓ System **Test Case ID#:** OPL_TieBreaker_1 **Test Description:** Test the tie-breaking mechanism for an OPL election where two parties have an equal number of votes. **Automated: yes ✓ no ___**	**Test Date:** November 13, 2023 **Name(s) of Testers:** Neha Bhatia **Indicate where are you storing the tests (what file) and the name of the method/functions being used:** OPLElectionTests.java test TieBreaker OPL()
Results: Pass ✓ Fail_____	

Preconditions for Test:
- An OPLElection object is set up with three parties: pR, pA, pD.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the election multiple times.	100 iterations	In each iteration, the election should result in pR and pA both having 2 seats and pD having 1 seat, as they have an equal number of votes.	In each iteration, the election resulted in pR and pA both having 2 seats and pD having 1 seat, as they have an equal number of votes.	-

Post condition(s) for Test:
- The election has been run multiple times to verify the tie-breaking mechanism.
- The percentages of pR and pA each getting 2 seats have been calculated.

Test Stage: Unit ✓ System

Test Case ID#: OPL_TieBreaker_Candidate

Test Description: Test the tie-breaking mechanism for an OPL election where if a party has more candidates than seats, and if two or more candidates have the same number of votes, preference should be given with a coin flip.

Automated: yes no ✓

Test Date: November 13, 2023

Name(s) of Testers: Neha Bhatia

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

Results: Pass Fail ✓

Preconditions for Test:

- An OPLElection csv file is set up so that for a party with 1 seat, there are two candidates with the same number of votes.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the election multiple times.	10 iterations	In each iteration, either candidate A or B should win (about 50% for both)	There was a skew towards preference of Candidate A.	-

Post condition(s) for Test:

- An OPLAudit file is still generated

Test Stage: Unit ✓ System

Test Case ID#: OPL_ SetUpElection_1

Test Description: Verify the setup of an OPLElection instance with candidate and party information.

Automated: yes ✓ no ___

Test Date: November 13, 2023

Name(s) of Testers: Neha Bhatia

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

	OPLElectionTests.java setUpElection()
Results: Pass ✓ Fail_____	

Preconditions for Test: Any dependencies utilized within setUpElection (such as Party, Candidate, and ElectionType classes) are functioning correctly

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	create an OPLElection instance with candidate and party information using the setUpElection method.	-	The setUpElection method should return a valid OPLElection instance with properly associated parties and candidates. The election should have: - 3 parties initialized: "D", "R", and "A". - Candidates a, b assigned to party "D". - Candidates c, d assigned to party "R". - Candidates e, f assigned to party "A".	As expected	-
2	Check the number of seats and ballot count in the OPLElection instance.	-	After setup, the OPLElection instance should have: - The number of seats set to 4. - The ballot count set to 13.	As expected	-
3	Validate the party and candidate lists within the OPLElection instance.	-	Post setup, the OPLElection	As expected	-

			instance's lists should reflect: - A list of 3 parties: "D", "R", and "A". - A list of 6 candidates: a, b, c, d, e, f. - Each candidate should be correctly assigned to their respective party. - The votes distribution as per the provided scenario: - Party "D" should have 3 votes. - Party "R" should have 8 votes. - Party "A" should have 2 votes. - Individual candidate votes should align with the provided counts: - a: 2 votes - b: 1 vote - c: 6 votes - d: 2 votes - e: 0 votes - f: 2 votes.		

Post condition(s) for Test:

- The setUpElection method should have properly initialized an OPLElection instance with accurate candidate and party information.
- OPLElection instance should have the correct number of seats and ballot count as specified in the method's setup logic.
- The OPLElection object should contain the expected party and candidate information as dictated by the provided setup within the setUpElection method. This includes correct affiliations and ballot counts for candidates within their respective parties.

Post condition(s) for Test:
- Election data is read and processed for an IR election.
- System maintains operability.

Test Stage: Unit ✓ System
Test Case ID#: Controller_Command_Processing_1
Test Description: Verify the response to an invalid input command.
Automated: yes no ✓

Test Date: November 14, 2023
Name(s) of Testers: Praful Das
Indicate where are you storing the tests (what file) and the name of the method/functions being used:
N/A

Results: Pass ✓ Fail_____

Preconditions for Test:
- The system is operational.
- The Controller class is compiled and accessible.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Entering an invalid command.	"abcd"	The system displays an error message indicating the invalid input command.	The system responds with the message "Invalid input. Please enter a valid command."	-

Post condition(s) for Test:
- System maintains operability.
- Error handling for invalid commands is confirmed.

Test Stage: Unit ✓ System
Test Case ID#: Controller_Interface_Display_1
Test Description: Verify the display of initial interface

Test Date: November 14, 2023
Name(s) of Testers: Praful Das

information upon program launch.

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

Automated: yes no ✓ N/A

Results: Pass ✓ Fail_____

Preconditions for Test:
- The system is operational.
- The Controller class is compiled and accessible.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Running the Vote Counting program.	-	The program displays welcome information and a list of available commands.	The program displays welcome information and a list of available commands.	-

Post condition(s) for Test:
- Initial interface information is correctly displayed.
- System maintains operability.

Test Stage: Unit ✓ System

Test Date: November 14, 2023

Test Case ID#: Controller_Interface_Display_1

Test Description: Verify the display of initial interface information upon program launch.

Name(s) of Testers: Praful Das

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

Automated: yes no ✓ N/A

Results: Pass ✓ Fail_____

Preconditions for Test:
- The system is operational.
- The Controller class is compiled and accessible.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Running the Vote Counting program.	-	The program displays welcome information and a list of available commands.	The program displays welcome information and a list of available commands.	-

Post condition(s) for Test:
- Initial interface information is correctly displayed.
- System maintains operability.

Test Stage: **Unit** ✓ **System**

Test Date: November 14, 2023

Test Case ID#: Controller_Exit_Command_1

Name(s) of Testers: Praful Das

Test Description: Validate the functionality of the exit command in the Vote Counting program.

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

Automated: **yes** **no** ✓

N/A

Results: **Pass** ✓ **Fail**_____

Preconditions for Test:
- The system is operational.
- The Controller class is compiled and accessible.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Entering the exit command.	"exit"	The program terminates and exits gracefully.	The program terminates upon entering the "exit" command.	-

Post condition(s) for Test:
- Program execution has ended.

Test Stage: Unit ✓ System

Test Date: November 14, 2023

Test Case ID#: Controller_Exit_Command_1

Name(s) of Testers: Praful Das

Test Description: Validate the response to an incorrect file name entered by the user (File not present in electionFiles)

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

Automated: yes no ✓

N/A

Results: Pass ✓ Fail_____

Preconditions for Test:
- The system is operational.
- The Controller class is compiled and accessible.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Entering an incorrect filename.	"NonExistentFile.csv"	The system notifies the user about the incorrect file name.	The system displays the message "The filename entered is incorrect."	-

Post condition(s) for Test:
- System maintains operability.
- Error handling for incorrect filenames is confirmed.

Test Stage: Unit ✓ System

Test Date: November 12, 2023

Test Case ID#: BallotTest_GetNextCandidate_1

Name(s) of Testers: Arpita Dev

Test Description: Test getNextCandidate() method to ensure it correctly retrieves and removes the first candidate from the ballot's formattedBallot list.

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

Automated: yes ✓ no

Test File Path: Project1/Testing/BallotTest.java

Name of Test: testGetNextCandidate

Results: Pass ✓ Fail

Preconditions for Test: An instance of ballot is created; the formattedBallot list is populated with two candidates, "Rosen" and "Kleinberg.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate a new Ballot object.	-	A new Ballot object is created.	A new Ballot object is created.	-
2	Populate the formattedBallot with candidates "Rosen" and "Kleinberg".	"Rosen", "Kleinberg"	The formattedBallot has two candidates.	The formattedBallot has two candidates.	Method should increment the digit by 1.
3	Call getNextCandidate() method on the Ballot object.	-	The method returns the first candidate, "Rosen".	The method returns the first candidate, "Rosen".	-
4	Check the size of formattedBallot after method execution.	-	The size of the list is 1.	The size of the list is 1.	Size decremented by one.
5	Ensure the removed candidate is no longer in the formattedBallot.	-	The candidate "Rosen" is not in the list.	The candidate "Rosen" is not in the list.	Candidate dequeued.

Post condition(s) for Test: The formattedBallot list size is reduced by one; the first candidate "Rosen" is removed from the formattedBallot list.

Notes: This test verifies the proper functionality of retrieving and dequeuing the next candidate from a ballot.

Test Stage: Unit ✓ System
Test Case ID#: BallotTest_IncrementRankingDigit_1

Test Description: Test the translation of a ballot with missing preferences.

Automated: yes ✓ no

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/BallotTest.java
Name of Test: testIncrementRankingDigit

Results: Pass ✓ Fail

Preconditions for Test: Ballot and Candidate classes are defined; setUp method initializes a list of Candidate objects correctly.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate a Ballot object	-	-	-	Object instantiation should be successful with ranking digit set to 0.
2	Call incrementRankingDigit method	-	ballotRankingDigit = 1	ballotRankingDigit = 1	Method should increment the digit by 1.
3	Retrieve the ranking digit	-	1	1	The getBallotRankingDigit should return the incremented value.

Post condition(s) for Test: Ballot ranking digit should be incremented by 1.
Notes: The test checks if the incrementRankingDigit method correctly increments the internal state of the ballotRankingDigit.

Test Stage: **Unit ✓** **System**
Test Case ID#: BallotTest_TranslateBallot_1

Test Description: Verify the translateBallot method correctly translates a valid string of rankings into a list of corresponding candidates.

Automated: **yes ✓** **no**

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/BallotTest.java
Name of Test: testTranslateBallotCorrectInput

Results: **Pass ✓** **Fail**

Preconditions for Test: A Ballot object must be instantiated, and a list of Candidate objects must be available.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate a new Ballot object	-	Ballot object should be created with default values	Ballot object instantiated	-
2	Prepare a list of Candidate objects	-	A list of candidates is available for ballot parsing	List prepared with "Rosen", "Kleinberg", "Chou", "Royce"	-
3	Call translateBallot with valid input	"1,2,3,4"	Translated ballot list matches input rankings	Translated ballot list correctly populated	Input reflects ordered rankings without missing preferences
4	Validate the size of the translated ballot	-	The size of the translated ballot should be 4	The size of the translated ballot is 4	Ensures that all candidate rankings are accounted for
5	Validate the content of the translated ballot	-	The translated ballot should contain candidates in the order "Rosen", "Kleinberg", "Chou", "Royce"	Translated ballot contains the correct candidates in the expected order	Validates the accuracy of translation based on rankings

Post condition(s) for Test: The Ballot object should have its formattedBallot property populated with the correct candidate rankings.

Notes: The ballot was correctly parsed, and the candidates were in the expected order. The test was automated, and the expected results matched the actual results. The test uses hard-coded candidate names and rankings to validate the translation process.

Test Stage: **Unit ✓** **System**
Test Case ID#: BallotTest_TranslateBallot_2

Test Description: Validate that translateBallot handles an empty ballot input correctly.

Automated: **yes ✓** **no**

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/BallotTest.java
Name of Test: testTranslateBallotCorrectInput

Results: **Pass** **Fail ✓**

Preconditions for Test: Ballot class is available and translateBallot method is implemented.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate Ballot object	-	-	Ballot object created	-
2	Call translateBallot with empty preferences	",,,,"	ArrayList of size 4 with all elements set to null	ArrayList of size 5 with all elements set to null	Test failed due to unexpected ArrayList size
3	Verify the size of the returned ArrayList	-	The size of the returned ArrayList should be 4	The size of the returned ArrayList is 5	Incorrect ArrayList size indicates a possible off-by-one error in implementation
4	Verify content of the returned ArrayList	-	All elements in the ArrayList should be null	Not applicable due to failure in step 2	-

Post condition(s) for Test: None.

Notes: The method should handle cases where all preferences are missing by assigning null to each preference slot.

Test Stage: **Unit ✓** **System**
Test Case ID#: BallotTest_TranslateBallot_3

Test Description: Test the translation of a ballot with missing preferences.

Automated: **yes ✓** **no**

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/BallotTest.java
Name of Test: testTranslateBallotWithMissingPreferences

Results: **Pass** **Fail ✓**

Preconditions for Test: Ballot and Candidate classes are defined; setUp method initializes a list of Candidate objects correctly.

Step	Test Step	Test	Expected	Actual	

#	Description	Data	Result	Result	Notes
1	Initialize Ballot object	-	Ballot object instantiated	Ballot object instantiated	-
2	Call translateBallot with missing preferences	"1,,3," and allCandidates	List should have the second element as null	List had non-null elements where null was expected	Test failed; the translateBallot method did not handle missing preferences as expected
3	Check size of translatedBallot	-	Size of list should be 4	Size of list was 4	-
4	Check for null where preference	-	translatedBallot.get(1) should be null	translatedBallot.get(1) was not null	Indicates a possible issue in preference parsing logic

Post condition(s) for Test: Ballot object should have a list of Candidate objects with the correct missing preferences handled.

Notes: The test checks if the translateBallot method correctly handles cases where preferences are missing (i.e., empty strings in the input).

Test Stage: **Unit** ✓ **System**
Test Case ID#: BallotTest_TranslateBallot_4

Test Description: Validate that the translateBallot method throws a NumberFormatException when given invalid numerical input.

Automated: **yes** ✓ **no**

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used
Test File Path: Project1/Testing/BallotTest.java
Name of Test: testTranslateBallotWithInvalidInput

Results: **Pass** ✓ **Fail**

Preconditions for Test: A Ballot object is created, and a list of all candidates is initialized.

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

1	Instantiate Ballot object	-	-	-	-
2	Call translateBallot with invalid input	"1,invalid,3,4"	NumberFormatException thrown	NumberFormatException thrown	The method should only accept numerical input for candidate rankings
3	Validate the exception is thrown	-	Test passes with expected exception	Test passes with expected exception	-

Post condition(s) for Test: An exception is expected to be thrown and caught by the test framework.

Notes: This test ensures the robustness of the input handling by the translateBallot method in the Ballot class.

Test Stage: **Unit** ✓ **System**
Test Case ID#: ElectionTypeTest_CreateCandidateList_1

Test Description: Verify that createCandidateList correctly parses a string into a list of Candidate objects with proper names and party associations.

Automated: **yes** ✓ **no**

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/ElectionTypeTest.java
Name of Test: testCreateCandidateList

Results: **Pass** ✓ **Fail**

Preconditions for Test: An instance of ElectionType with the anonymous implementation for runElection is available.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate ElectionType	-	Instance of ElectionType	Instance created successfully.	-

	with necessary overrides.		created.		
2	Call createCandidateList with a string of entries.	"Rosen D, Kleinberg R, Chou I, Royce L"	List populated with 4 candidates, names and parties correctly parsed.	List populated with 4 candidates, names and parties correctly parsed.	-
3	Validate the size of the candidate list.	-	Size of list should be 4.	Size of list is 4.	-
4	Validate the names and parties of the candidates.	-	First candidate should be "Rosen" from party "D".	First candidate is "Rosen" from party "D".	As expected.
5	Validate the names and parties of the candidates.	-	Second candidate should be "Kleinberg" from party "R".	Second candidate is "Rosen" from party "R".	As expected.
6	Validate the names and parties of the candidates.	-	Third candidate should be "Chou" from party "I".	Third candidate is "Chou" from party "I".	As expected.
7	Validate the party of the second candidate	-	Fourth candidate should be "Royce" from party "L".	Fourth candidate is "Royce" from party "L".	As expected.
8	Confirm that there are no additional candidates.	-	No additional candidates beyond the four specified.	No additional candidates present.	Correct number of entries.
9	Check for successful completion of test case.	-	Test case completes without errors.	Test case completed without errors.	Test case passed.

Post condition(s) for Test: ArrayList<Candidate> is populated with Candidate objects based on the input string.
Notes: The test provides a string of candidate names and expected parties, and checks if the createCandidateList method processes this string correctly.

Test Stage: Unit ✓ System
Test Case ID#: ElectionTypeTest_GettersAndSetters_1

Test Description: Validate getters and setters of the ElectionType class

Automated: yes ✓ no

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/ElectionTypeTest.java
Name of Test: testGettersAndSetters

Results: Pass ✓ Fail

Preconditions for Test: An instance of the ElectionType class must be instantiated.

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

1	Instantiate ElectionType object	-	ElectionType object created	As Expected	Object instantiation successful
2	Set ballot count	10	ballotCount property set to 10	As Expected	Setter method works as expected
3	Set candidates count	4	candidatesCount property set to 4	As Expected	Setter method works as expected
4	Set election type	"IR"	electionType property set to "IR"	As Expected	Setter method works as expected
5	Retrieve ballot count	-	Returns 10	As Expected	Getter method works as expected
6	Retrieve candidates count	-	Returns 4	As Expected	Getter method works as expected
7	Retrieve election type	-	Returns “IR”	As Expected	Getter method works as expected

Post condition(s) for Test: The ElectionType instance should have its properties set and retrieved correctly.

Notes: The test ensures that the ElectionType properties are properly accessed and modified through its getters and setters.

Test Stage: Unit ✓ System
Test Case ID#: ElectionTypeTest_TieBreaker_1

Test Description: Ensure that the tieBreaker method in ElectionType class provides an approximately equal chance of selection between two candidates over multiple invocations.

Automated: yes ✓ no

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/ElectionTypeTest.java
Name of Test: testTieBreakerEqualChance

Results: Pass ✓ Fail

Preconditions for Test: ElectionType instance must be initialized with a subclass that provides an implementation for the abstract runElection method. Two distinct Candidate objects must be instantiated and provided to the tieBreaker method.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize ElectionType with two candidates.	Candidate("Candidate1"), Candidate("Candidate2")	-	-	-

2	Call tieBreaker 1000 times with the candidates.	-	Each candidate has a 45%-55% chance of winning	-	-
3	Record the win count for each candidate.	-	Record the win count for each candidate.	-	See actual results for values
4	Assert the win count falls within expected range	-	assertTrue for the expected win count range	Passed	Test passed, win counts are within the expected range

Post condition(s) for Test: None.

Notes: The test simulates the tieBreaker method 1000 times to statistically validate that each candidate has an approximately equal chance of being selected.

Test Stage: **Unit ✓** **System**
Test Case ID#: ElectionTypeTest_TieBreaker_2

Test Description: Verify the tieBreaker method provides an approximately equal chance of selection among three candidates

Automated: **yes ✓** **no**

Test Date: November 12, 2023

Name(s) of Testers: Arpita Dev

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/ElectionTypeTest.java
Name of Test: testTieBreakerUnequalChance

Results: **Pass ✓** **Fail**

Preconditions for Test: ElectionType instance must be created and initialized with three Candidate instances.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create ElectionType instance and three Candidate objects	ElectionType, Candidate("Candidate1"), Candidate("Candidate2"), Candidate("Candidate3")	ElectionType and Candidate instances are created successfully	As Expected	-
2	Run the tieBreaker method 1000	Array of three Candidate	Each candidate is selected	As Expected	Actual percentages

	times	instances	approximately 28%-37% of the time		should be checked to ensure they fall within the expected range
3	Assert the distribution of the tieBreaker selections	1000 iterations of tieBreaker	Assert passes if each candidate's win count is within the range of 280 to 370	Each candidate's win count is within the range of 280 to 370	fails, the test should indicate an issue with the randomness of the tieBreaker

Post condition(s) for Test: None.

Notes: The test runs the tieBreaker method 1000 times and tallies the selection of each candidate to ensure a fair distribution of chances.

Test Stage: **Unit** **System ✓**

Test Date: November 15, 2023

Test Case ID#: OPLElection_1

Name(s) of Testers: Praful Das

Test Description: Verify the vote counting system accurately processes an OPLElection CSV file.

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

Automated: **yes** **no ✓**

N/A

Results: **Pass** **Fail**_____

Preconditions for Test:

- Properly formatted OPLElection CSV file available in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Enter the filename for an OPLElection CSV file.	"OPLElection.csv"	The system reads the OPLElection CSV file, correctly counts the votes for the election, determines the winner, generates audit file, and prints the results.	Works as expected for most situations but is not able to break ties correctly.	-

Post condition(s) for Test:

The system displays the OPLElection voting results.

Test Stage: **Unit** **System** ✓	**Test Date:** November 15, 2023
Test Case ID#: IRElection_1 **Test Description:** Verify the vote counting system accurately processes an IRElection CSV file.	**Name(s) of Testers:** Praful Das
Automated: **yes** **no** ✓	**Indicate where are you storing the tests (what file) and the name of the method/functions being used:** N/A
Results: **Pass** ✓ **Fail**_____	

Preconditions for Test:
- Properly formatted OPLElection CSV file available in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Enter the filename for an IRElection CSV file.	"IRElection.csv"	The system reads the IRElection CSV file, correctly counts the votes for the election, determines the winner, generates audit file, and prints the results.	Calculates winner correctly, but is not able to generate audit file.	-

Post condition(s) for Test:
The system displays the OPLElection voting results.

Test Stage: **Unit** ✓ **System** **Test Case ID#:** PartyTest_AddCandidateToList_1	**Test Date:** November 13, 2023 **Name(s) of Testers:** Arpita Dev

Test Description: Add a new candidate to the party list

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

Test File Path: Project1/Testing/PartyTest.java

Automated: yes ✓ no

Name of Test: testAddCandidateToList_NewCandidate

Results: Pass ✓ Fail

Preconditions for Test:
- `Party` object instantiated with the name "Test Party"
- Two `Candidate` objects added to the party (`candidate1` and `candidate2`)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate Party object	"Test Party"	Party object created	As Expected	Party should be initialized with no candidates
2	Add two candidates to the party list	candidate1, candidate2	List size 2	As Expected	Preconditions are met for the test
3	Add a new candidate to the party list	candidate3	List size 3	As Expected	The new candidate is added successfully

Post condition(s) for Test: Candidate` object `candidate3` should be added to the party's candidate list

Notes: This test ensures that new candidates are added successfully and that the list size increments appropriately.

Test Stage: Unit ✓ System

Test Date: November 13, 2023

Test Case ID#: PartyTest_AddCandidateToList_2

Name(s) of Testers: Arpita Dev

Test Description: Ensure that adding an existing candidate to the party's candidate list does not alter the size of the list.

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/PartyTest.java
Name of Test: testAddCandidateToList_ExistingCandidate

Automated: yes ✓ no

Results: Pass ✓ Fail

Preconditions for Test: The **Party** object must be initialized with two candidates, "Rosen" and "Kleinberg".

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate a new Party object.	"Test Party"	Party object created with empty candidate list	Party object created successfull.	**setUp** method handles this
2	Add two unique candidates to the party list	"Rosen", "Kleinberg"instances	Party list size is 2	Party list size is confirmed as 2	**setUp** method handles this
3	Attempt to add an existing candidate	"Rosen"	Party list size should remain 2	Party list size remains 2	Test verifies no duplicates are added
4	Verify the candidate list size	-	Party list size should still be 2	Party list size is 2 after adding duplicate	Confirms expected behavior of the method

Post condition(s) for Test: The candidate list size should remain unchanged after attempting to add an existing candidate.

Notes: This test verifies that the **addCandidateToList** method correctly ignores duplicate entries.

Test Stage: Unit ✓ System

Test Date: November 13, 2023

Test Case ID#: PartyTest_IncrementVoteCount_1

Name(s) of Testers: Arpita Dev

Test Description: Validate the incrementation of the vote count for a party

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/PartyTest.java
Name of Test: testIncrementVoteCount

Automated: yes ✓ no

Results: Pass ✓ Fail

Preconditions for Test: A **Party** object is created and initialized with a default vote count of 0.

A **Party** object is created and initialized with a default vote count of 0.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a **Party** object with initial conditions.	-	-	-	Party object created successfully.
2	Get the initial vote count for the party.	-	0	0	Initial vote count verified as 0
3	Call the **incrementVoteCount** method on party.	-	-	-	Method called without errors
4	Retrieve the updated vote count.	-	1	1	Vote count incremented as expected

Post condition(s) for Test: The partyVoteCount of the Party object should be increased by 1.

Notes: This test ensures that the incrementVoteCount method accurately increments the vote count of a party.

Test Stage: Unit ✓ System

Test Date: November 13, 2023

Test Case ID#: PartyTest_DecrementVoteCount_1

Name(s) of Testers: Arpita Dev

Test Description: Validate the decrementVoteCount method in Party class decreases the vote count by one when vote count is greater than zero

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

Test File Path: Project1/Testing/PartyTest.java

Automated: yes ✓ no

Name of Test: testDecrementVoteCount

Results: Pass ✓ Fail

Preconditions for Test: A Party object with an initialized vote count greater than zero.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Party object and increment vote count by one	Party object with voteCount initialized to 0	voteCount should be 1 after increment	voteCount is 1	-
2	Call decrementVoteCount on Party object	Party object with voteCount at 1	voteCount should be 0 after decrement	voteCount is 0	-
3	Verify the vote count is decremented correctly	Party object after calling decrementVoteCount	voteCount should be 0	voteCount is 0	Test passed

Post condition(s) for Test: The Party object's vote count should be decremented by one.

Notes: The test ensures that the decrementVoteCount method does not allow the vote count to go below zero.

Test Stage: Unit ✓ System

Test Date: November 13, 2023

Test Case ID#: PartyTest_DecrementVoteCount _2

Name(s) of Testers: Arpita Dev

Test Description: Verify that **decrementVoteCount** does not reduce the party's vote count below zero

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/PartyTest.java
Name of Test: testDecrementVoteCount_AtZero

Automated: yes ✓ no

Results: Pass ✓ Fail

Preconditions for Test: Party object created and no votes have been added (vote count is at initial state of zero)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate Party object with no votes	-	Party's initial vote count is 0	As Expected	Preconditions met
2	Call **decrementVoteCount** on Party object	-	Party's vote count remains at 0	As Expected	**decrementVoteCount** does not affect vote count at zero
3	Check Party's vote coun	-	Party's vote count should be 0	As Expected	Post conditions met. Test passed successfully.

Post condition(s) for Test: Party's vote count should remain zero after calling **decrementVoteCount**.

Notes: Ensures the vote count cannot go into negative values which is critical for accurate vote tallying.

Test Stage: Unit ✓ System
Test Case ID#: PartyTest_GetCandidateCount _1

Test Date: November 13, 2023

Name(s) of Testers: Arpita Dev

Test Description: Test the getCandidateCount method to ensure it returns the correct number of candidates added to the party

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/PartyTest.java
Name of Test: testGetCandidateCount

Automated: yes ✓ no

Results: Pass ✓ Fail

Preconditions for Test: A `Party` object must be instantiated and two `Candidate` objects must be added to this party

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate a new Party object	"Test Party"	Party's initial vote count is 0	As Expected	-
2	Add two Candidate objects to the Party object	Candidate: "Rosen", "Kleinberg"	Two candidates added	As Expected	-
3	Call `getCandidateCount` method	-	Count: 2	As Expected	Test passes if counts match
4	Verify the count of candidates	-	Count should match expected	Count matches expected	-

Post condition(s) for Test: None.

Notes: The test checks if the getCandidateCount method correctly reports the number of candidates after they are added to the party's candidate list.

Test Stage: Unit ✓ System

Test Date: November 13, 2023

Test Case ID#: PartyTest_GetPartySeatCount_1

Name(s) of Testers: Arpita Dev

Test Description: Validate initial party seat count is zero before any seats are added

Indicate where are you storing the tests (what file) and the name of the method/functions being used:
Test File Path: Project1/Testing/PartyTest.java
Name of Test: testGetPartySeatCount

Automated: yes ✓ no

Results: Pass ✓ Fail

Preconditions for Test: Party object must be instantiated with no seats assigned

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate a Party object with the name "Test Party"	"Test Party"	-	-	Initialization step
2	Check the initial seat count of the party	-	0	0	Verification step

Post condition(s) for Test: Party seat count should remain unchanged.

Notes: This test ensures that the seat count property is properly initialized and that no seats are counted before any election process begins.

Test Stage: Unit ✓ System
Test Case ID#: PartyTest_AddToPartySeatCount_1

Test Description: Verify that addToPartySeatCount() **correctly adds seats to a party's total seat count**

Test Date: November 13, 2023

Name(s) of Testers: Arpita Dev

Automated: yes ✓ no	**Indicate where are you storing the tests (what file) and the name of the method/functions being used:** **Test File Path:** Project1/Testing/PartyTest.java **Name of Test:** testAddToPartySeatCount
Results: Pass ✓ Fail	

Preconditions for Test: A `Party` object must be instantiated with a name "Test Party" and at least one candidate added to the party's candidate list

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate a new Party object with the name “Test Party”	“Test Party”	Party object created with the specified name	As Expected	-
2	Add a candidate to the party’s candidate list	Candidate object “Rosen”	Candidate list size is 1	As Expected	-
3	Call addToPartySeatCount on the Party ohbject	3	Party's seatCount should increase by 3	As Expected	-
4	Asser that the seat count matches the expected result	-	Assert passes with the expected seat count of 3	As Expected	

Post condition(s) for Test: The Party object's seatCount should be incremented by the number of seats specified in the test data.

Notes: The method should only accept positive integers and should increase the seat count accordingly.

Test Stage: Unit ✓ System **Test Case ID#:** PartyTest_ DecrementVoteCount _2 **Test Description:** Verify that **`decrementVoteCount`** does not reduce the party's vote count below zero	**Test Date:** November 13, 2023 **Name(s) of Testers:** Arpita Dev
Automated: yes ✓ no	**Indicate where are you storing the tests (what file) and the**

name of the method/functions being used:
Test File Path: Project1/Testing/PartyTest.java
Name of Test: testDecrementVoteCount_AtZero

Results: Pass ✓ Fail

Preconditions for Test: Party object created and no votes have been added (vote count is at initial state of zero)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate Party object with no votes	-	Party's initial vote count is 0	As Expected	Preconditions met
2	Call **decrementVoteCount** on Party object	-	Party's vote count remains at 0	As Expected	**decrementVoteCount** does not affect vote count at zero
3	Check Party's vote coun	-	Party's vote count should be 0	As Expected	Post conditions met. Test passed successfully.

Post condition(s) for Test: Party's vote count should remain zero after calling **decrementVoteCount**.

Notes: Ensures the vote count cannot go into negative values which is critical for accurate vote tallying.

Test Stage: Unit ✓ System
Test Case ID#: PartyTest_ DecrementVoteCount _2

Test Description: Verify that **decrementVoteCount** does not reduce the party's vote count below zero

Test Date: November 13, 2023

Name(s) of Testers: Arpita Dev

Automated: yes ✓ no	**Indicate where are you storing the tests (what file) and the name of the method/functions being used:** **Test File Path:** Project1/Testing/PartyTest.java **Name of Test:** testGetPartyName
Results: Pass ✓ Fail	

Preconditions for Test: Party object created and no votes have been added (vote count is at initial state of zero)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate Party object with no votes	-	Party's initial vote count is 0	As Expected	Preconditions met
2	Call **decrementVoteCount** on Party object	-	Party's vote count remains at 0	As Expected	**decrementVoteCount** does not affect vote count at zero
3	Check Party's vote coun	-	Party's vote count should be 0	As Expected	Post conditions met. Test passed successfully.

Post condition(s) for Test: Party's vote count should remain zero after calling **decrementVoteCount**.

Notes: Ensures the vote count cannot go into negative values which is critical for accurate vote tallying.