Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: getNumObjsTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the method getNumObjs correctly gets the number of objects from the file.
Automated: yes
Results: Pass
Preconditions for Test: OPLManager is correctly implemented. "OPL_test.txt" exists in the right directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading and skip to the line for number of objects	File: "OPL_test.txt"	File is read in	As expected	
2	Read the second line to get the number of objects.	File: “OPL_test.txt	Line is read in	As expected	
3	Compare the returned value with the expected number of objects.	File: “OPL_test.txt”	Number of objects = 6	As expected	

Post condition(s) for Test: The method getNumObjs is correctly parses and return the number of objects from the OPL_test.txt file.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: getNumBallotsTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the method getNumBallots correctly gets the number of ballots from the file.
Automated: yes
Results: Pass
Preconditions for Test: OPLManager is correctly implemented. "OPL_test.txt" exists in the right directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading and skip to the line where the number of ballots is stored	File: "OPL_test.txt"	File is read in	As expected	
2	Read the line to get the number of ballots.	File: “OPL_test.txt	Line is read in	As expected	
3	Compare the returned value with the expected number of ballots.	File: “OPL_test.txt”	Number of ballots= 9	As expected	

Post condition(s) for Test: The method getNumBallots is correctly parses and return the number of ballots from the OPL_test.txt file.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: getNumSeatsTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the method getNumSeats correctly gets the number of objects from the file.
Automated: yes
Results: Pass
Preconditions for Test: OPLManager is correctly implemented. "OPL_test.txt" exists in the right directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading and skip to the line where number of seats is stored	File: "OPL_test.txt"	File is read in	As expected	

2	Read the line to get the number of seats.	File: “OPL_test.txt	Line is read in	As expected	
3	Compare the returned value with the expected number of seats.	File: “OPL_test.txt”	Number of seats= 2	As expected	

Post condition(s) for Test: The method getNumSeats is correctly parses and return the number of seats from the OPL_test.txt file.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading	"OPL_test.txt"	File is accessible and can be read	As expected	
2	Skip to the line in file where party information begins	(N/A)	The parser correctly navigates to the party information in the file	As expected	
3	Parse party information using parseParty method	(Content from file)	An OPLParty object with name "Democrat" and candidate "Pike" is created	As expected	

Project Name: Project 1: Voting System Team#14
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: partsePartyTestOPL
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: This tests that the parseParty method in OPLManager correctly parses a party's information from a given file and creates an OPLParty object with the correct party name and candidates list.
Automated: yes
Results: Pass
Preconditions for Test: "OPL_test.txt" is in the right directory.

Post condition(s) for Test: The OPLParty object named "Democrat" with candidate "Pike" is created and matches the expected result.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parsePartyTestOPL2
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the party parsing correctly creates a party object based on input string.
Automated: yes
Results: Pass
Preconditions for Test: Voting and Party classes are correctly implemented

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Split input line to extract party name.	Line: "Democrat, Pike"	Party name "Democrat" is extracted.	As expected	
2	Check if party exists in the party dictionary.	Party name: "Democrat"	Retrieve existing party object or create a new one if it does not exist.	As expected	
3	Verify the party object's name.	Party name: "Democrat"	Party object with the name "Democrat" is either found or created.	As expected	

Post condition(s) for Test: A party object with the name "Democrat" is successfully retrieved from or added to the party dictionary.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parsePartyTestOPL3
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests if the system correctly parses a line to create a party and candidate, and adds them to the party and candidate dictionaries.
Automated: yes
Results: Pass
Preconditions for Test: OPLManager, OPLCandidate, and Voting are correctly implemented.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Split input line to extract party name and candidate name.	Line: "Democrat, Pike"	Party name "Democrat" and candidate name "Pike" are extracted.	As Expected	

2	Check if the party exists in the party dictionary, and create or retrieve the party object.	Party name: "Democrat"	Party object with name "Democrat" is either found or created.	As Expecte d	
3	Add candidate to the party and candidate dictionary.	Candidate name: "Pike"	Candidate "Pike" is added to the party "Democrat" and to the vote's candidate dictionary.	As Expecte d	
4	Verify the number of candidates in the candidate dictionary.	Candidate dictionary	The candidate dictionary size is 1	As Expecte d	

Post condition(s) for Test: The party and candidate dictionaries in the Voting object are correctly populated.

Project Name: Project 1: Voting System Team# 14
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseBallotsTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: This tests that the parseBallots method in OPLManager accurately reads ballot information from a file and assigns votes correctly to both parties and individual candidates.
Automated: Yes
Results: Pass
Preconditions for Test: “OPL_test.txt" is in the correct directory.

Test Step Table:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note s
1	Open "OPL_test.txt" for reading	"OPL_test.txt"	File is read in	As expected	
2	Read and set the number of seats from the file	"OPL_test.txt"	Correct number of seats is set in vote object	As expected	

3	Read and set the number of ballots from the file	"OPL_test.txt"	Correct number of ballots is set in vote	As expected	
4	Read the number of parties/objects from the file	"OPL_test.txt"	Correct number of parties is parsed	As expected	
5	Parse party information and add parties to the vote	"OPL_test.txt"	Parties are correctly added with initial data	As expected	
6	Parse ballots and distribute votes among parties and candidates	"OPL_test.txt"	Votes are correctly assigned to parties and candidates	As expected	
7	Verify the total votes for each party are correct	"OPL_test.txt"	Party votes match expected values	As expected	
8	Verify the total votes for each candidate are correct	"OPL_test.txt"	Candidate votes match expected values	As expected	

Post condition(s) for Test: The vote object correctly reflects the vote distribution among parties and candidates as specified in "OPL_test.txt".

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseBallotsTestOPL2
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Verifies that ballot parsing in an OPL system correctly processes and validates the presence of each ballot line from the input file.
Automated: yes
Results: Pass
Preconditions for Test: "OPL_test.txt" is in the correct directory. The OPLManager and related classes are correctly implemented.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading and skip header.	File: "OPL_test.txt"	File is read in	As expected	

2	Get number of seats, ballots, and candidates from the file.	File: "OPL_test.txt"	Correct values for num_seats, num_ballots, and num_objs based on the file	As expecte d	
3	Set the gotten values in the Voting object.	num_seats, num_ballots	Voting object is updated	As expecte d	
4	Verify each ballot line's presence and format.	File: "OPL_test.txt"	Each line for ballots is present and correctly formatted.	As expecte d	

Post condition(s) for Test: Each ballot line from "OPL_test.txt" is successfully read.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseBallotsTestOPL3
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the ballot format is read in correctly.
Automated: yes
Results: Pass
Preconditions for Test: "OPL_test.txt" is in the correct directory. OPLManager is implemented correctly

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note s
1	Open "OPL_test.txt" and bypass the header to begin ballot parsing.	File: "OPL_test.txt"	File is read in	As expecte d	
2	Extract data for seats, ballots, and candidates.	File: "OPL_test.txt"	Correctly determine numbers for seats, total ballots, and candidate objects based on the file	As expecte d	

3	Update the Voting object with the seat and ballot numbers.	Values of num_seats and num_ballots	Voting object has correct number of seats and ballots	As expected	
4	Read and verify each ballot line.		Each ballot line is non null	As expected	

Post condition(s) for Test: Each ballot is verified and not null.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseBallotsTestOPL4
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Verifies that ballot parsing for an OPL vote correctly identifies ballots with at least one vote marked.
Automated: yes
Results: Pass
Preconditions for Test: "OPL_test.txt" file in the correct directory. OPLManager is correctly implemented.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading and skip the header.	File: "OPL_test.txt"	File is read in	As expected	
2	Extract number of seats, ballots, and candidates from the file.	File: "OPL_test.txt"	Correct num_seats, num_ballots, and num_objs values are gotten from the file	As expected	
3	Set the extracted values in the Voting object.	num_seats, num_ballots	Voting object is updated with correct settings.	As expected	

4	Read each ballot line, ensuring at least one '1' per ballot.	File: "OPL_test.txt"	Each ballot contains at least one vote ('1').	As expected	

Post condition(s) for Test: Each ballot has a vote.

Project Name: Project 1: Voting System Team#14
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: makeVoteTestOPL
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the makeVote method in OPLManager class successfully creates and returns a non-null Voting object.
Automated: Yes
Results: Pass
Preconditions for Test: OPLManager is initialized

Test Step Table:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate an OPLManager object.	N/A	OPLManager object is created.	As expected	
2	Call the makeVote method on the OPLManager object.	N/A	Method returns a non-null Voting object.	As expected	
3	Assert that the returned Voting object is not null.	N/A	Test passes if object is not null, indicating successful object creation.	As expected	

Post condition(s) for Test: A new instance of the Voting object is created.

Project Name: Project 1: Voting System Team#14
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: makePartyTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Verifies that the OPLManager can correctly create an OPLParty object with a specified name.
Automated: yes
Results: Pass
Preconditions for Test: OPLManager is initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create an instance of OPLManager.	N/A	Instance created.	As expected	
2	Use makeParty method on OPLManager with party name "qwerty".	"qwerty"	OPLParty object with name "qwerty" is created.	As expected	

Post condition(s) for Test: A new OPLParty object with the name "qwerty" exists.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: makeCandidateTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests the creation of an OPLCandidate object with specified name and party attributes using OPLManager.
Automated: yes
Results: Pass
Preconditions for Test: OPLManager is initialized

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

1	Initialize OPLManager object.	N/A	OPLManager instance created.	As expecte d	
2	Create an OPLCandidate object using makeCandidate method.	N/A	OPLCandidate object is created with specified name and party.	As expecte d	
3	Verify candidate's name is correctly set.	Created OPLCandidat e	Candidate's name matches expected name.	As expecte d	
4	Verify candidate's party is correctly set.	Created OPLCandidat e	Candidate's party matches expected party.	As expecte d	

Post condition(s) for Test: An OPLCandidate object with the specified name and party exists.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseCSVTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests parsing of an OPL CSV file to correctly initialize a Voting object with the expected number of ballots, seat numbers, candidates, and parties.
Automated: yes
Results: Pass
Preconditions for Test: "OPL_test.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note s

1	Open "OPL_test.txt" for reading.	"testing/OPL_test.txt"	File is read in	As expected	
2	Parse the CSV file to create a Voting object.	"testing/OPL_test.txt"	Voting object is created with data from file.	As expected	
3	Verify the number of ballots is correctly set.	Created Voting object	Voting object has 9 ballots.	As expected	
4	Verify the number of seat numbers is correctly set.	Created Voting object	Voting object has 2 seats.	As expected	
5	Verify the number of candidates is correct.	Created Voting object	Voting object has 6 candidates.	As expected	
6	Verify the number of parties is correct.	Created Voting object	Voting object has 3 parties.	As expected	

Post condition(s) for Test: A Voting object with the data in "OPL_test.txt" exists with accurate attributes for ballots, seat numbers, candidates, and parties.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseCSVTestOPL3
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that parsing an OPL CSV file correctly populates the party dictionary with expected parties.
Automated: yes
Results: Pass
Preconditions for Test: "OPL_test.txt" in the right directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading.	"testing/OPL_test.txt"	File is read in	As expected	
2	Parse the CSV file to create a Voting object.	"testing/OPL_test.txt"	Voting object is created with data from file.	As expected	
3	Verify the number of parties is correctly set.	Created Voting object	Voting object has 3 parties.	As expected	
4	Confirm presence of "Democrat" in party dictionary.	Created Voting object	"Democrat" entry exists in the dictionary.	As expected	
5	Confirm presence of "Republican" in party dictionary.	Created Voting object	"Republican" entry exists in the dictionary.	As expected	
6	Confirm presence of "Independent1" in party dictionary.	Created Voting object	"Independent1" entry exists in the dictionary.	As expected	

Post condition(s) for Test: A Voting object reflecting the data in "OPL_test.txt" exists with a correctly populated party dictionary.

---

separate

Project Name: Project 1: Voting System Team#14
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parsePartyTestCPL

Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: This tests that the parseParty method in CPLManager correctly parses a party's information from a given file and creates a CPLParty object with the correct party name and candidates list.
Automated: yes
Results: Pass
Preconditions for Test: "CPL_test.txt" is in the right directory.

Post condition(s) for Test: The CPLParty object named "Democratic" containing candidates "Joe", “Sally”, and “Ahmed” is created

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test.txt" for reading	"CPL_test.txt"	File is read	As expected	
2	Skip to the line in file where party information begins	(N/A)	The parser navigates to the relevant information in the file	As expected	
3	Parse party information using parseParty method	"CPL_test.txt"	A CPLParty object with name "Democratic" and candidates "Joe", “Sally”, and “Ahmed” is created	As expected	

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: partyPartyTestCPL2
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests if parsing a correctly formatted String identifies and creates a party with the specified name.
Automated: yes
Results: Pass
Preconditions for Test: CPLManager is correctly implemented. The String is valid.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Parse a line representing a party and its candidates.	Line: "Democrat, Pike, Etta, Alawa"	A CPLParty object is created or fetched from the party dictionary with the name "Democrat".	As Expected	

2	Verify the party name is correctly set.	Expected: "Democrat"	CPLParty object has the name "Democrat".	As Expecte d	

Post condition(s) for Test: A CPLParty object with the name "Democrat" exists.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parsePartyTestCPL3
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that parsing a line from a CPL file correctly populates a party with candidates.
Automated: yes
Results: Pass
Preconditions for Test: CPLManager is correctly implemented. The String is correctly formatted

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note s
1	Parse a line representing a party and its candidates.	Line: "Democrat, Pike, Etta, Alawa"	A CPLParty object with candidates Pike, Etta, and Alawa is created or updated.	As expecte d	
2	Verify the number of candidates in the party.	The CPL party object	The CPLParty object has 3 candidates.	As expecte d	

Post condition(s) for Test: A CPLParty object with the name "Democrat" and three candidates exists.

Project Name: Project 1: Voting System Team# 14
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseBallotsTestCPL
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: This tests that the parseBallots method in CPLManager accurately reads ballot information from a file and assigns votes correctly to both parties and individual candidates.
Automated: Yes
Results: Pass
Preconditions for Test: "CPL_test.txt" is in the right directory.

Test Step Table:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test.txt" for reading	"CPL_test.txt"	File is accessible and readable	As expected	
2	Read and set the number of seats from the file	"CPL_test.txt"	Correct number of seats is set in vote object	As expected	
3	Read and set the number of ballots from the file	"CPL_test.txt"	Correct number of ballots is set in vote	As expected	
4	Read the number of parties/objects from the file	"CPL_test.txt"	Correct number of parties is parsed	As expected	
5	Parse party information and add parties to the vote	"CPL_test.txt"	Parties are correctly added with initial data	As expected	
6	Parse ballots and distribute votes among parties and candidates	"CPL_test.txt"	Votes are correctly assigned to parties and candidates	As expected	
7	Verify the total votes for each party are correct	"CPL_test.txt"	Party votes match expected values	As expected	
8	Verify the total votes for each candidate are correct	"CPL_test.txt"	Candidate votes match expected values	As expected	

Post condition(s) for Test: The vote object correctly reflects the votes among parties and candidates as specified in "CPL_test.txt".

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseBallotsTestCPL2
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that ballots in a CPL file are not null.
Automated: yes
Results: Pass
Preconditions for Test: CPLManager is correctly implemented. CPL_test.txt exists in the right directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note s
1	Open "CPL_test.txt" for reading.	File: "CPL_test.txt"	File is read in	As expected	
2	Read the file line by line to process each ballot.	File: "CPL_test.txt"	Each line (ballot) is not null.	As expected	

Post condition(s) for Test: All ballots from "CPL_test.txt" are not null

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/24
Test Case ID#: CPL_ParseBallotsTest3_1
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that each character in a ballot line is not null.
Automated: yes
Results: Pass
Preconditions for Test: CPLManager is implemented. "CPL_test.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note s
1	Open "CPL_test.txt" for reading.	File: "CPL_test.txt"	File is read in	As expected	
2	Process each ballot line.	File: "CPL_test.txt"	Each character in the ballot line is not null.	As expected	

Post condition(s) for Test: Each ballot line from "CPL_test.txt" has non null characters for however many parties there are.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseBallotsTestCPL4
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Verifies that each ballot line contains a '1'.
Automated: yes
Results: Pass
Preconditions for Test: CPLManager is correctly implemented. "CPL_test.txt" is in the right directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test.txt" for reading.	File: "CPL_test.txt"	File is read in	As expected	
2	Read and process each ballot line.	File: “CPL_test.txt”	Each ballot line has a ‘1’.	As expected	

Post condition(s) for Test: Each ballot line has a ‘1’.

Project Name: Project 1: Voting System Team#14
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: makeVoteTestCPL
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the makeVote method in CPLManager class successfully creates and returns a non-null Voting object.
Automated: Yes
Results: Pass
Preconditions for Test: CPLManager object is initialized

Test Step Table:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Instantiate an CPLManager object.	N/A	CPLManager object is created.	As expected	
2	Call the makeVote method on the CPLManager object.	N/A	Method returns a non-null Voting object.	As expected	
3	Assert that the returned Voting object is not null.	N/A	Test passes if object is not null, indicating successful object creation.	As expected	

Post condition(s) for Test: A new instance of the Voting object is created.

Project Name: Project 1: Voting System Team#14
Test Stage: Unit
Test Date: 3/24
Test Case ID#: makePartyTestCPL
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the CPLManager can correctly create an CPLParty object with a specified name.
Automated: yes
Results: Pass
Preconditions for Test: CPLManager is initialized.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create an instance of CPLManager.	N/A	Instance created.	As expected	

2	Use makeParty method on CPLManager with party name "qwerty".	"qwerty"	CPLParty object with name "qwerty" is created.	As expected	

Post condition(s) for Test: A new PLParty object with the name "qwerty" exists.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: makeCandidateTestCPL
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests the creation of an CPLCandidate object with specified name and party attributes using CPLManager.
Automated: yes
Results: Pass
Preconditions for Test: CPLManager is initialized

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize CPLManager object.	N/A	CPLManager instance created.	As expected	
2	Create an CPLCandidate object using makeCandidate method.	N/A	CPLCandidate object is created with specified name and party.	As expected	
3	Verify candidate's name is correctly set.	Created CPLCandidate	Candidate's name matches expected name.	As expected	
4	Verify candidate's party is correctly set.	Created CPLCandidate	Candidate's party matches expected party.	As expected	

Post condition(s) for Test: A CPLCandidate object with the specified name and party exists.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/24
Test Case ID#: parseCSVTestCPL
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that parsing of an CPL file correctly initializes a Voting object with the expected number of ballots, seat numbers, candidates, and parties.
Automated: yes
Results: Pass
Preconditions for Test:"CPL_test.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test.txt" for reading.	"CPL_test.txt"	File is read in	As expected	
2	Parse the CSV file to create a Voting object.	"CPL_test.txt"	Voting object is created with data from file.	As expected	
3	Verify the number of ballots is correctly set.	Created Voting object	Voting object has 9 ballots.	As expected	
4	Verify the number of seat numbers is correctly set.	Created Voting object	Voting object has 3 seats.	As expected	
5	Verify the number of parties is correct.	Created Voting object	Voting object has 6 parties.	As expected	

6	Verify the number of seats is correct.	Created Voting object	Voting object has 3 seats.	As expected	

Post condition(s) for Test: A Voting object with the data in "CPL_test.txt" exists with accurate attributes for ballots, seat numbers, candidates, and parties.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: parseCSVTestCPL2
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that parsing an CPL CSV file correctly populates the party dictionary with expected parties.
Automated: yes
Results: Pass
Preconditions for Test: "CPL_test.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test.txt" for reading.	"CPL_test.txt"	File is read in	As expected	
2	Parse the CSV file to create a Voting object.	“CPL_test.txt"	Voting object is created with data from file.	As expected	
3	Verify the number of parties is correctly set.	Created Voting object	Voting object has 3 parties.	As expected	
4	Confirm presence of "Democrat" in party dictionary.	Created Voting object	"Democrat" entry exists in the dictionary.	As expected	

5	Confirm presence of "Republican" in party dictionary.	Created Voting object	"Republican" entry exists in the dictionary.	As expected	
6	Confirm presence of "Independent1" in party dictionary.	Created Voting object	"Independent1" entry exists in the dictionary.	As expected	
7	Confirm presence of "New Wave" in party dictionary.	Created Voting object	"New Wave" entry exists in the dictionary.	As expected	
8	Confirm presence of "Reform" in party dictionary.	Created Voting object	“Reform" entry exists in the dictionary.	As expected	
9	Confirm presence of "Green" in party dictionary.	Created Voting object	“Green" entry exists in the dictionary.	As expected	

Post condition(s) for Test: A Voting object reflecting the data in "CPL_test.txt" exists with a correctly populated party dictionary.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `getSeatPercentageTest()`
Name(s) of Testers: Michael Diep
Test Description: Tests the percentage of the total seats a party acquire.
Automated: yes
Results: Pass
Preconditions for Test: Prior getters and setters must be correct

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

1	Create a new OPL voting object	Created OPL Voting object	new object created	As expected	
2	Created a new OPL party object	Created OPL party object	new object created	As expected	
3	Set that OPL party's allocated seat to 2	Created OPL party object	party has 2 allocated seats	As expected	
4	Set Voting object's available seats to 10	Created OPL Voting object	Voting object has 10 seats to give	As expected	
5	Called and set variable result to getSeatPercentage	Voting and Party object used	result is equal 20%	As expected	
6	Verify the result percentage with 20%	String variable result	Result verified at 20%	As expected	

Post condition(s) for Test: A Voting object with one OPL Party object exists with 10% of the votings ballot votes.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `getVotePercentageTest()`
Name(s) of Testers: Michael Diep
Test Description: Tests the percentage of the total votes a party acquires.
Automated: yes
Results: Pass
Preconditions for Test: Prior getters and setters must be correct

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

1	Create a new OPL voting object	Created OPL voting object	new object created	As expected	
2	Created a new OPL party object	Created OPL party object	new object created	As expected	
3	Added one vote to that OPL party	Created Party object	party has 1 ballot vote	As expected	
4	Set Voting object’s number of ballots to 10	Created Voting object	Voting object has 10 seats.	As expected	
5	Called and set variable results to getSeatPercentage	Voting and Party object used	result is equal 10%	As expected	
6	Verify the result percentage with 10%	String variable result	Result verified at 10%	As expected	

Post condition(s) for Test: A Voting object with one OPL Party object exists with 20% of the votings ballot existing;

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `OPLgiveSeatsRoutineTest()`
Name(s) of Testers: Michael Diep
Test Description: Give seats to the OPL candidates.
Automated: yes
Results: Pass
Preconditions for Test: Opl test file must be a correct file, and parsing the data must be correct

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

1	Create a new scanner to take in a file	Created Scanner object	new scanner object created	As expected	
2	Created a new OPL manager object	Created OPL manager object	new manager object created	As expected	
3	Set Voting object to the manager object that parses the scanner file	Created scanner file object, manager, and voting object	Voting parses the passed in file	As expected	
4	Verify each of the Candidates gotSeats boolean	Created Voting object	Voting Candidates getGotSeats were verified	As expected	

Post condition(s) for Test: A Voting object with the OPL test file data is created. Candidate Pike and Etta or Alawa got a seat.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `OPLaddPartyTest()`
Name(s) of Testers: Michael Diep
Test Description: Adds OPL party to the parties hashmap in the Voting object.
Automated: yes
Results: Pass
Preconditions for Test: Prior getters and setters must be correct

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new Voting OPL object	Created Voting object	new Voting object created	As expected	
2	Created 3 new OPL Party objects and set their name	Created OPL Voting and Party object	3 new Part objects created	As expected	
3	Called add Party object on Voting object	Created Voting object and Party object	2 Party objects added to the parties hashmap	As expected	
4	Verify the size of the Parties hashmap	Voting Parties hashmap	Verified to 2 parties added to the Parties hashmap	As expected	

Post condition(s) for Test: A Voting object with 2 parties is added to the parties hashmap

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `OPLaddCandidateTest()`
Name(s) of Testers: Michael Diep
Test Description: Adds OPL candidates to the candidates hashmap in the Voting object.
Automated: yes
Results: Pass
Preconditions for Test: Prior getters and setters must be correct

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new Voting OPL object	Created Voting object	new Voting object created	As expected	
2	Created and added a new OPL Party objects and set their name	Created OPL Voting and Party object	new Part objects created	As expected	
3	Created new OPL candidate with name cand1 and called add candidate	Created Voting object and Candidate object	Candidate was added to candidates hashmap in Voting	As expected	
4	Created another OPL candidate with same name cand1 and called add candidate	Created Voting object and Candidate object	Candidate was not added to candidates hashmap in Voting, because of its same name	As expected	
5	Verify candidates hashmap size	Created Voting object and candidates	Candidates size matches expected size	As expected	

Post condition(s) for Test: A Voting object with 1 candidate is added to the candidates hashmap

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `CPLgiveSeatsRoutineTest()`
Name(s) of Testers: Michael Diep
Test Description: Gives seats to the CPL candidates.
Automated: yes
Results: Pass
Preconditions for Test: Cpl test file must be a correct file, and parsing the data must be correct

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new scanner to take in a file	Created Scanner object	new scanner object created	As expected	
2	Created a new CPL manager object	Created CPL manager object	new manager object created	As expected	
3	Set Voting object to the manager object that parses the scanner file	Created scanner file object, manager, and voting object	Voting parses the passed in file	As expected	
4	Verify each of the Candidates gotSeats boolean	Created Voting object	Voting Candidates getGotSeats were verified	As expected	

Post condition(s) for Test: A Voting object with the CPL test file data is created. Candidate Joe, Allen, and Xinyue got a seat.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `CPLaddPartyTest()`
Name(s) of Testers: Michael Diep
Test Description: Adds CPL party to the parties hashmap in Voting object.
Automated: yes
Results: Pass
Preconditions for Test: Prior getters and setters must be correct

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new Voting CPL object	Created Voting object	new Voting object created	As expected	
2	Created 3 new CPL Party objects and set their name	Created OPL Voting and Party object	3 new Part objects created	As expected	
3	Called add Party object on Voting object	Created Voting object and Party object	2 Party objects added to the parties hashmap	As expected	

4	Verify the size of the Parties hashmap	Voting Parties hashmap	Verified to 2 parties added to the Parties hashmap	As expected	

Post condition(s) for Test: A Voting object with 2 parties is added to the parties hashmap

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `CPLaddCandidateTest()`
Name(s) of Testers: Michael Diep
Test Description: Adds CPL candidate to the candidates hashmap in Voting object.
Automated: yes
Results: Pass
Preconditions for Test: Prior getters and setters must be correct

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new Voting CPL object	Created Voting object	new Voting object created	As expected	
2	Created and added a new OPL Party objects and set their name	Created CPL Voting and Party object	new Part objects created	As expected	
3	Created new CPL candidate with name cand1 and called add candidate	Created Voting object and Candidate object	Candidate was added to candidates hashmap in Voting	As expected	
4	Created another CPL candidate with same name cand1 and called add candidate	Created Voting object and Candidate object	Candidate was not added to candidates hashmap in Voting, because of its same name	As expected	
5	Verify candidates hashmap size	Created Voting object and candidates	Candidates size matches expected size	As expected	

Post condition(s) for Test: A Voting object with 1 candidate is added to the candidates hashmap

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `testBreakTieTest()`
Name(s) of Testers: Michael Diep

Test Description: Breaks a tie between two components passed in.
Automated: yes
Results: Pass
Preconditions for Test: Voting object must be properly implemented

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new Voting OPL object	Created Voting object	new Voting object created	As expected	
2	Declare string result and call breakTie	Created OPL Voting object	String result set to either “A” or “B”	As expected	
3	Verify whether results is “A” or “B” and not random result “C”	Created OPL Voting object and string result	Verified String result to either “A” or “B”	As expected	

Post condition(s) for Test: A string is returned to Voting object with a 50% chance of return either or of your inputs

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `allocateSeatsRoutineTest()`
Name(s) of Testers: Michael Diep
Test Description: Allocate final seats to each party in the Voting object.
Automated: yes
Results: Does not Pass
Preconditions for Test: Prior getters and setters must be correct

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new Voting OPL object	Created Voting object	new Voting object created	As expected	
2	Set Voting object’s ballots to 10 and seat numbers to 3	Created Voting object, integer 3 and 10	Voting object has 10 ballots and 3 seats	As expected	
3	Created 3 new OPL Party objects and set their name	Created OPL Voting and Party object	3 new Part objects created	As expected	

4	Called add Party object on Voting object	Created Voting object and Party object	3 Party objects added to the parties hashmap	As expected	
5	Added votes to each Party object	Voting object and Party object	Party1 has 6 votes, Party2 has 3 votes, and party3 has 1 vote	As expected	
6	Call allocateSeats() on Voting object	Voting object	Party object is awarded seats	As expected	
7	Verify each party has the correct amount of seats	Party object	Party1 has 2 seats, Party2 has 1 seat, Party3 has 0 seats	As expected	Final call to giveSeats at the end of the method will result in a Null pointer exception because candidates have not been created. But it should allocate seats properly to each party

Post condition(s) for Test: A Voting object with 3 parties is added to the parties hashmap, and 3 of those parties has their allocated seats

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `firstAllocRoutineTest()`
Name(s) of Testers: Michael Diep
Test Description: Give each party their first round allocation of seats.
Automated: yes
Results: Pass 50% of times, depending on tie breaker
Preconditions for Test: Cpl test file must be a correct file, and parsing the data must be correct

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new scanner to take in a file	Created Scanner object	new scanner object created	As expected	
2	Created a new CPL manager object	Created CPL manager object	new manager object created	As expected	
3	Set Voting object to the manager object that parses the scanner file	Created scanner file object, manager, and voting object	Voting parses the passed in file	As expected	
4	Set integer seatsGiven to the calling of firstAlloc	Created Voting object, integer quota=2, and integer seatsLeft=4	Gives the parties their first round allocation, which ends up being their final seats because no seats are left and their remainders are negligible	As expected	
4	Verify each of the Parties FirstAllocation numbers	Created Voting object, parties hashmap, and party objects	Parties Dem. received either 1 or 2 first round allocated seats, Repub. got 1 or 2 seats, and Green gets 1 or 0 seats	As expected	

Post condition(s) for Test: A Voting object with the CPL test file data is created. A Voting object with 3 parties is added to the parties hashmap, and 3 of those parties has their first round allocated seats

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/25
Test Case ID#: `secondAllocRoutineTest()`
Name(s) of Testers: Michael Diep
Test Description: Give each party their second round allocation seats.
Automated: yes
Results: Pass
Preconditions for Test: Cpl test file must be a correct file, and parsing the data must be correct, and method calls prior to this function is correct

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new scanner to take in a file	Created Scanner object	new scanner object created	As expected	

2	Created a new CPL manager object	Created CPL manager object	new manager object created	As expected	
3	Set Voting object to the manager object that parses the scanner file	Created scanner file object, manager, and voting object	Voting parses the passed in file	As expected	
4	Call firstAlloc	Created Voting object, integer quota=3, and integer seatsLeft=2	Gives the parties their first round allocation, remainders are taken account in the secondAlloc	As expected	
5	Call secondAlloc	Created Voting object, integer quota=3, and integer seatsLeft=1	Gives the parties their second round allocation based on their remainders	As expected	
6	Verify each of the Parties SecondAllocation numbers	Created Voting object, parties hashmap, and party objects	Parties Repub. received either 1 second round allocated seats, Dem. and New Wave did not receive any second allocation seats	As expected	

Post condition(s) for Test: A Voting object with the CPL test file data is created. A Voting object with 3 parties is added to the parties hashmap, and 3 of those parties has their first and second round allocated seats

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_CPL1
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an CPL election.
Automated: No
Results: Pass
Preconditions for Test:"CPL_test.txt" is in the correct directory.

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

1	Open "CPL_test.txt" for reading.	"CPL_test.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"CPL_test.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"CPL_test.txt"	9 ballots.	9 ballots	
4	Verify the number of seats is correct.	"CPL_test.txt"	3 seats.	3 seats	
5	Verify the number of parties is correct.	"CPL_test.txt"	6 parties.	6 parties	
6	Verify the winners of the election is correct in both the terminal and “output.txt”	"CPL_test.txt"	Seat Winners: Joe: Democratic Xinyue: Reform Allen: Republican	Joe: Democratic Xinyue: Reform Allen: Republican	

Post condition(s) for Test: The results of the election given the CPL_test.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_CPL2
Name(s) of Testers: Riandy Setiadi

Test Description: Testing that the system displays the correct output given a .csv file for an CPL election.
Automated: No
Results: Pass
Preconditions for Test:"CPL_test2.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test2.txt" for reading.	"CPL_test2.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"CPL_test2.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"CPL_test2.txt"	9 ballots.	9 ballots	
4	Verify the number of seats is correct.	"CPL_test2.txt"	4 seats.	4 seats	
5	Verify the number of parties is correct.	"CPL_test2.txt"	3 parties.	3 parties	
6	Verify the winners of the election is correct in both the terminal and "output.txt"	"CPL_test2.txt"	Seat Winners: Brian: New Wave Poyo: Republican Sally: Democratic Joey: Republican	Seat Winners: Brian: New Wave Poyo: Republican Sally: Democratic Joey: Republican	

Post condition(s) for Test: The results of the election given the CPL_test2.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_CPL3
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an CPL election.
Automated: No
Results: Pass
Preconditions for Test:"CPL_test3.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test3.txt" for reading.	"CPL_test3.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"CPL_test3.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"CPL_test3.txt"	10 ballots.	10 ballots.	
4	Verify the number of seats is correct.	"CPL_test3.txt"	2 seats.	2 seats.	
5	Verify the number of parties is correct.	"CPL_test3.txt"	3 parties	3 parties	
6	Verify the winners of the election is correct in both the terminal and “output.txt”	"CPL_test3.txt"	Seat Winners: Sophie: Democratic Jeffrey: New Wave	Seat Winners: Sophie: Democratic Jeffrey: New Wave	

Post condition(s) for Test: The results of the election given the CPL_test3.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_CPL4
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an CPL election.
Automated: No
Results: Pass
Preconditions for Test:"CPL_test4.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test4.txt" for reading.	"CPL_test4.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"CPL_test4.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"CPL_test4.txt"	6 ballots	6 ballots	
4	Verify the number of seats is correct.	"CPL_test4.txt"	2 seats.	2 seats	
5	Verify the number of parties is correct.	"CPL_test4.txt"	4 parties.	4 parties	
6	Verify the winners of the election is correct in both the terminal and “output.txt”	"CPL_test4.txt"	Seat Winners: Mike: Republican	Seat Winners: Mike: Republican	

			Candice: Reform	Candice: Reform	

Post condition(s) for Test: The results of the election given the CPL_test4.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_CPL_Tiebreaker1
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an CPL election with a tie.
Automated: No
Results: Pass
Preconditions for Test:" "CPL_testing_tiebreaker1.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_testing_tiebreaker1.txt" for reading.	"CPL_testing_tiebreaker1.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"CPL_testing_tiebreaker1.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"CPL_testing_tiebreaker1.txt"	10 ballots	10 ballots	
4	Verify the number of seats is correct.	"CPL_testing_tiebreaker1.txt"	4 seats.	4 seats	

5	Verify the number of parties is correct.	"CPL_testing_tiebreaker1.txt"	3 parties.	3 parties	
6	Verify the winners of the election is correct in both the terminal and "output.txt"	"CPL_testing_tiebreaker1.txt"	Seat Winners: 2 Republican, 1 Democrat, 1 Green OR 2 Democrat, 1 Republican, 1 Green OR 2 Republican 2 Democrat	As expected	

Post condition(s) for Test: The results of the election given the CPL_testing_tiebreaker1.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_CPL_Tiebreaker2
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an CPL election with a tie.
Automated: No
Results: Pass
Preconditions for Test:" "CPL_testing_tiebreaker2.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_testing_tiebreaker2.txt" for reading.	"CPL_testing_tiebreaker2.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"CPL_testing_tiebreaker2.txt"	Results printed to the terminal	As expected	

3	Verify the number of ballots is correctly set.	"CPL_testing_tiebreaker2.txt”	16 ballots	16 ballots	
4	Verify the number of seats is correct.	"CPL_testing_tiebreaker2.txt”	6 seats.	6 seats	
5	Verify the number of parties is correct.	"CPL_testing_tiebreaker2.txt”	6 parties.	6 parties	
6	Verify the winners of the election is correct in both the terminal and “output.txt”	"CPL_testing_tiebreaker2.txt”	Seat Winners: 2 Independent, 2 Republican, 1 New Wave/Reform/Green AND 0 to 1 of Wave/Reform/Green OR 2 Independent/2 Republican 1 Independent/1 Republican 1 New Wave/Reform/Green	As expected	

Post condition(s) for Test: The results of the election given the CPL_test.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_CPL_Tiebreaker3
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an CPL election with a tie.
Automated: No
Results: Not Passed
Preconditions for Test:" "CPL_testing_tiebreaker3.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_testing_tiebreaker3.txt" for reading.	"CPL_testing_tiebreaker3.txt”	File is read in	As expected	
2	Parse the CSV file to get an output	"CPL_testing_tiebreaker3.txt”	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"CPL_testing_tiebreaker3.txt”	14 ballots	14 ballots	
4	Verify the number of seats is correct.	"CPL_testing_tiebreaker3.txt”	9 seats.	9 seats	
5	Verify the number of parties is correct.	"CPL_testing_tiebreaker3.txt”	4 parties.	4 parties	
6	Verify the winners of the election is correct in both the terminal and “output.txt”	"CPL_testing_tiebreaker3.txt”	All parties meet the quota, some of them several times over, many results possible	10 seats assigned and results that should be random seem to be consistent	

Post condition(s) for Test: The results of the election given the CPL_testing_tiebreaker3.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_CPL_Tiebreaker4
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an CPL election with a tie.
Automated: No
Results: Pass
Preconditions for Test:" "CPL_testing_tiebreaker4.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_testing_tiebreaker4.txt" for reading.	"CPL_testing_tiebreaker4.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"CPL_testing_tiebreaker4.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"CPL_testing_tiebreaker4.txt"	11 ballots	11 ballots	
4	Verify the number of seats is correct.	"CPL_testing_tiebreaker4.txt"	7 seats.	7 seats	
5	Verify the number of parties is correct.	"CPL_testing_tiebreaker4.txt"	5 parties.	5 parties	

6	Verify the winners of the election is correct in both the terminal and “output.txt”	"CPL_testing_tiebreaker4.txt”	Seat Winners: Since all parties meet the quota of 1 there should be 7 seats assigned with random results every time	As expected	

Post condition(s) for Test: The results of the election given the CPL_testing_tiebreaker4.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_OPL1
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an OPL election
Automated: No
Results: Pass
Preconditions for Test:" "OPL_test.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading.	"OPL_test.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"OPL_test.txt”	Results printed to the terminal	As expected	

3	Verify the number of ballots is correctly set.	"OPL_test.txt"	9 ballots	9 ballots	
4	Verify the number of seats is correct.	"OPL_test.txt"	2 seats.	2 seats	
5	Verify the number of parties is correct.	"OPL_test.txt"	3 parties.	3 parties	
6	Verify the winners of the election is correct in both the terminal and “output.txt”	"OPL_test.txt"	Seat Winners: Pike: Democrat received 2 Votes Alawa: Republican received 2 Votes	As expected	

Post condition(s) for Test: The results of the election given the OPL_test.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_OPL2
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an OPL election
Automated: No
Results: Pass
Preconditions for Test:" "OPL_test2.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test2.txt" for reading.	"OPL_test2.txt"	File is read in	As expected	

2	Parse the CSV file to get an output	"OPL_test2.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"OPL_test2.txt"	9 ballots	9 ballots	
4	Verify the number of seats is correct.	"OPL_test2.txt"	3 seats.	3 seats	
5	Verify the number of parties is correct.	"OPL_test2.txt"	3 parties.	3 parties	
6	Verify the winners of the election is correct in both the terminal and "output.txt"	"OPL_test2.txt"	Seat Winners: Kevin: Republican received 5 Votes John: Republican received 1 Votes Jim: New Wave received 3 Votes	As expected	

Post condition(s) for Test: The results of the election given the OPL_test2.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_OPL3
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an OPL election
Automated: No
Results: Pass
Preconditions for Test:" "OPL_test3.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Note s
1	Open "OPL_test4.txt" for reading.	"OPL_test3.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"OPL_test3.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"OPL_test3.txt"	9 ballots	9 ballots	
4	Verify the number of seats is correct.	"OPL_test3.txt"	3 seats.	3 seats	
5	Verify the number of parties is correct.	"OPL_test3.txt"	3 parties.	3 parties	
6	Verify the winners of the election is correct in both the terminal and "output.txt"	"OPL_test3.txt"	Seat Winners: Billy: New Wave received 2 Votes Mike: Democratic received 3 Votes Vivi: Republican received 4 Votes	As expected	

Post condition(s) for Test: The results of the election given the OPL_test3.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System

Test Date: 3/24
Test Case ID#: System_Test_OPL4
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an OPL election
Automated: No
Results: Pass
Preconditions for Test:" "OPL_test4.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test4.txt" for reading.	"OPL_test4.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"OPL_test4.txt”	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"OPL_test4.txt"	8 ballots	8 ballots	
4	Verify the number of seats is correct.	"OPL_test4.txt"	4 seats.	4 seats	
5	Verify the number of parties is correct.	"OPL_test4.txt"	2 parties.	2 parties	

6	Verify the winners of the election is correct in both the terminal and "output.txt"	"OPL_test4.txt"	Seat Winners: Zirui: Republican received 1 Votes Nick: Republican received 3 Votes Shana: Democratic received 3 Votes Sam: Democratic received 1 Votes	As expected	

Post condition(s) for Test: The results of the election given the OPL_test4.txt csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_OPL_Tiebreaker1
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an OPL election with a tie.
Automated: No
Results: Pass
Preconditions for Test: "OPL_testing_tiebreaker1.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_testing_tiebreaker1.txt" for reading.	"OPL_testing_tiebreaker1.txt"	File is read in	As expected	

2	Parse the CSV file to get an output	"OPL_testing_tiebreaker1.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"OPL_testing_tiebreaker1.txt"	10 ballots	10 ballots	
4	Verify the number of seats is correct.	"OPL_testing_tiebreaker1.txt"	2 seats.	2 seats	
5	Verify the number of parties is correct.	"OPL_testing_tiebreaker1.txt"	3 parties.	3 parties	
6	Verify the winners of the election is correct in both the terminal and "output.txt"	"OPL_testing_tiebreaker1.txt"	Seat Winners: Vivi/Sally get a seat Chen/Candice get a seat	As expected	

Post condition(s) for Test: The results of the election given the "OPL_testing_tiebreaker1.txt" csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_OPL_Tiebreaker2
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an OPL election with a tie.
Automated: No
Results: Pass
Preconditions for Test: "OPL_testing_tiebreaker2.txt" is in the correct directory.

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

1	Open "OPL_testing_tiebreaker2.txt" for reading.	"OPL_testing_tiebreaker2.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"OPL_testing_tiebreaker2.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"OPL_testing_tiebreaker2.txt"	7 ballots	10 ballots	
4	Verify the number of seats is correct.	"OPL_testing_tiebreaker2.txt"	3 seats.	2 seats	
5	Verify the number of parties is correct.	"OPL_testing_tiebreaker2.txt"	3 parties.	3 parties	
6	Verify the winners of the election is correct in both the terminal and “output.txt”	"OPL_testing_tiebreaker2.txt"	Seat Winners: Maggie gets a seat Tyson, Kevin, Corry, Samantha, or Emily get a seat	As expected	

Post condition(s) for Test: The results of the election given the "OPL_testing_tiebreaker2.txt" csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_OPL_Tiebreaker3
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an OPL election with a tie.
Automated: No
Results: Pass
Preconditions for Test: "OPL_testing_tiebreaker3.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_testing_tiebreaker3.txt" for reading.	"OPL_testing_tiebreaker3.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"OPL_testing_tiebreaker3.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"OPL_testing_tiebreaker3.txt"	10 ballots	10 ballots	
4	Verify the number of seats is correct.	"OPL_testing_tiebreaker3.txt"	2 seats.	2 seats	
5	Verify the number of parties is correct.	"OPL_testing_tiebreaker3.txt"	3 parties.	3 parties	
6	Verify the winners of the election is correct in both the terminal and "output.txt"	"OPL_testing_tiebreaker3.txt"	Seat Winners: Beth gets a seat always Joe/Andrea/Alice get a seat	As expected	

Post condition(s) for Test: The results of the election given the "OPL_testing_tiebreaker3.txt" csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: System
Test Date: 3/24
Test Case ID#: System_Test_OPL_Tiebreaker4
Name(s) of Testers: Riandy Setiadi
Test Description: Testing that the system displays the correct output given a .csv file for an OPL election with a tie.
Automated: No
Results: Pass
Preconditions for Test: "OPL_testing_tiebreaker4.txt" is in the correct directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_testing_tiebreaker4.txt" for reading.	"OPL_testing_tiebreaker4.txt"	File is read in	As expected	
2	Parse the CSV file to get an output	"OPL_testing_tiebreaker4.txt"	Results printed to the terminal	As expected	
3	Verify the number of ballots is correctly set.	"OPL_testing_tiebreaker4.txt"	9 ballots	10 ballots	
4	Verify the number of seats is correct.	"OPL_testing_tiebreaker4.txt"	2 seats.	2 seats	
5	Verify the number of parties is correct.	"OPL_testing_tiebreaker4.txt"	3 parties.	3 parties	

6	Verify the winners of the election is correct in both the terminal and “output.txt”	"OPL_testing_tiebreaker4.txt"	Seat Winners: John gets a seat Eric/Penny/Yusef gets a seat	As expected	

Post condition(s) for Test: The results of the election given the "OPL_testing_tiebreaker4.txt" csv has been printed out onto the terminal to be displayed and output.txt has been created and contains the same results

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: forLoopAllocateSeatsTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the for loops in AllocateSeats work properly
Automated: yes
Results: Pass
Preconditions for Test: File is read in correctly and a vote object was created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test4.txt" for reading.	File: "CPL_test4.txt"	File is read in	As expected	
2	For loop goes through, iterates four times and increments iterations four times and partyGot twice	File: “CPL_test4.txt	For loop does not give an error	As expected	
3	Compare the returned value with the expected number of objects.	File: “CPL_test4.txt”	Iterations - 4 partyGot - 2	As expected	

Post condition(s) for Test: The for loop has gone through the parties in the dictionary the correct number of times and allocated 2 seats to the parties.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: forLoopFirstAllocTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the for loops in AllocateSeats work properly
Automated: yes
Results: Pass
Preconditions for Test: File is read in correctly and a vote object was created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test4.txt" for reading.	File: "CPL_test4.txt"	File is read in	As expected	
2	For loop goes through, iterates four times and increments iterations four times and partyGot once and then checks to see if the Republican party's remainder votes got properly allocated	File: “CPL_test4.txt	For loop does not give a NullPointerException	As expected	
3	Compare the returned value with the expected number of objects.	File: “CPL_test4.txt”	Iterations - 4 partyGot - 1 Republican remainder - 2	As expected	

Post condition(s) for Test: The for loop has gone through the parties in the dictionary the correct number of times and allocated 1 seat to the republican party and as a result has 2 seats in their remainder.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: whileLoopFirstAllocTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang

Test Description: Tests that the for loops in AllocateSeats work properly
Automated: yes
Results: Pass
Preconditions for Test: File is read in correctly and a vote object was created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "CPL_test4.txt" for reading.	File: "CPL_test4.txt"	File is read in	As expected	
2	For loop goes through, and checks number of ties present in the current dictionary	File: “CPL_test4.txt	For loop does not give a NullPointerException	As expected	
3	Compare the returned value with the expected number of objects.	File: “CPL_test4.txt”	onePartyTie = 4 Democrats 2 seats, Republicans 1 seat or Democrats 1 seat, Republicans 2 seats = true Green party 0 seats or 1 seat = true	As expected	

Post condition(s) for Test: The for loop has gone through the parties in the dictionary and the parties have the correct number of seats,

---

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: whileSecondAllocTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the for loops in AllocateSeats work properly
Automated: yes
Results: Pass
Preconditions for Test: File is read in correctly and a vote object was created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading.	File: "OPL_test.txt"	File is read in	As expected	
2	For loop goes through the dictionary of parties.	File: "OPL_test.txt"	For loop does not give a NullPointerException	As expected	
3	Compare the returned value with the expected number of objects.	File: "OPL_test.txt"	whileIterations = 1 forIterations = 3 Democrat Second Allocation = 1 Republican Second Allocation = 0 Independent Second Allocation = 0	As expected	

Post condition(s) for Test: The for loop has gone through the parties in the dictionary and the parties have the correct number second allocation seats.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: forOPLGiveSeatsTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the for loops in AllocateSeats work properly
Automated: yes
Results: Pass
Preconditions for Test: File is read in correctly and a vote object was created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading.	File: "OPL_test.txt"	File is read in	As expected	

2	For loop goes through the dictionary of parties.	File: "OPL_test.txt"	For loop does not give a NullPointerException	As expected	
3	Compare the returned value with the expected number of objects.	File: "OPL_test.txt"	forIteration = 3	As expected	

Post condition(s) for Test: The for loop has gone through the parties in the dictionary.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: for2OPLGiveSeatsTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the for loops in AllocateSeats iterates properly
Automated: yes
Results: Pass
Preconditions for Test: File is read in correctly and a vote object was created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading.	File: "OPL_test.txt"	File is read in	As expected	
2	For loop goes through the dictionary of parties.	File: "OPL_test.txt"	For loop does not give a NullPointerException	As expected	
3	Another for loop goes through dictionary of candidates	File: “OPL_test.txt”	For loop iterates three times for the three candidates in the party	As expected	
3	Compare the returned value with the expected number of objects.	File: "OPL_test.txt"	For Iteration = 3, for the 3 candidates in democratic	As expected	

Post condition(s) for Test: The for loop has gone through the candidates in Democratic party.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: whileOPLGiveSeatsTest
Name(s) of Testers: Michael Diep, Billy Ha, Riandy Setiadi, Vivian Tsang
Test Description: Tests that the for loops in AllocateSeats work properly
Automated: yes
Results: Pass
Preconditions for Test: File is read in correctly and a vote object was created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open "OPL_test.txt" for reading.	File: "OPL_test.txt"	File is read in	As expected	
2	Set the whileIterations to the number of the Democrat allocated seats.	File: "OPL_test.txt"	Democrats have seats allocated to them.	As expected	
3	Compare the returned value with the expected number of objects.	File: "OPL_test.txt"	whileIteration = 1	As expected	

Post condition(s) for Test: whileIteration has been set to the number of allocated seats the Democrats have.

Project Name: Voting System
Test Stage: Unit
Test Date: 3/23/24
Test Case ID#: conditionalOPLGiveSeatsTest
Name(s) of Testers: Michael Diep
Test Description: Tests that the conditions in OPLGiveSeats implements tie breaker properly
Automated: yes
Results: Not pass, Null pointer exception
Preconditions for Test: File is read in correctly and a vote object was created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new scanner to take in a file	Created Scanner object	new scanner object created	As expected	
2	Created a new OPL manager object	Created OPL manager object	new manager object created	As expected	
3	Set Voting object to the manager object that parses the scanner file	Created scanner file object, manager, and voting object	Voting parses the passed in file	As expected	
4	Verify each of the Candidates gotSeats boolean	Created Voting object	Voting Candidates getGotSeats were verified	As expected	

Post condition(s) for Test: Candidates will be awarded seats.