Testing Logs:

Names: Jashwin Acharya (achar061), Carlos Chasi-Mejia(chasi009), Vincent Hoang(hoang317), Steve Petzold(petzo017)
Team: 08

Candidate Tests:

Unit Tests:

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 1

Name(s) of Testers: Steven Petzold

Test Description:

Testing the getter method for the name attribute of the Candidate class

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

Project1/src/CandidateTests.java

Automated: yes X **no**

Results: Pass X **Fail**

Preconditions for Test: A candidate has been created with a valid candidate name and party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1					

2	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Candidate**	String candidateName = "Rosen"; String partyName = "Independent";	getName() should return "Rosen" and the test should return True.	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the CandidateTests class are run
3					
4					

Post condition(s) for Test:

The AssertEquals test passes where the .getName() method equals the original candidate name that was created.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 2

Name(s) of Testers: Steven Petzold

Test Description:

Testing the getter method for the party attribute of the Candidate class

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

Project1/src/CandidateTests.java

Automated: yes _X_ **no** ___

Results: Pass X **Fail**

Preconditions for Test: A candidate has been created with a valid candidate name and party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Candidate**	String candidateName = "Rosen"; String partyName = "Independent";	getParty() function should return the party “Independent” and the test should return True	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the CandidateTests class are run
2					
3					
4					

Post condition(s) for Test:

The AssertEquals test passes where the .getParty() method equals the original candidate name that was created.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: **Unit** _X_ **System** __

Test Date: 11/8/23

Test Case ID#: 3

Name(s) of Testers: Steven Petzold

Test Description:

Testing the NumVotesIncrement method for the Candidate class.
Also checks if the getNumVotes method returns the correct number of votes.

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

Project1/src/CandidateTests.java

Automated: yes__X_ no ___

Results: Pass ___X___ Fail _____

Preconditions for Test: A candidate has been created with a valid candidate name and party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Candidate**	String candidateName = "Kleinberg"; String partyName = "Party A";	NumVotesIncrement() method should update the number of votes by 1. Test should return true when the value of getNumVotes() is compared to value of "2" which is the number of times the function has been called within the test.	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the CandidateTests class are run
2					
3					
4					

Post condition(s) for Test:

The AssertEquals test passes where the getNumVotes() method equals 2, which is the number of times the numVotesIncrement() has been called.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 4

Name(s) of Testers: Steven Petzold

Test Description:

Testing the incrementRedistributedVotes method for the Candidate class. Also checks if the getRedistributedVotes returns the correct value.

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

Project1/src/CandidateTests.java

Automated: yes X **no**

Results: Pass ___X__ **Fail**_____

Preconditions for Test: A candidate has been created with a valid candidate name and party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../li**	String candidateName = "Kleinberg"; String partyName = "Party A";	incrementRedistributedVotes() method should update the number of redistributed votes by 1. getRedistributedVotes should obtain the value 2 as this is the number of times the increment function has been called. Test should return true	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the CandidateTests class are run
2					

	b/hamcrest-core-1.3.jar:. RunAllUnitTestCases Candidate				
3					
4					

Post condition(s) for Test:

The AssertEquals test passes where the getRedistributedVotes() method equals 2, which is the number of times the incrementRedistributedVotes() has been called.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 5

Name(s) of Testers: Steven Petzold

Test Description:

Testing the resetRedistributedVotes method for the Candidate class. Also checks if the getRedistributedVotes returns the correct value.

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

Project1/src/CandidateTests.java

Automated: yes X **no**

Results: Pass X **Fail**

Preconditions for Test: A candidate has been created with a valid candidate name and party name.

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

1					
2	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Candidate**	String candidateName = "Kleinberg"; String partyName = "Party A";	resetRedistributedVotes() method should reset the value of the redistributed votes to 0. The getRedistributedVotes() should verify this value.	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the CandidateTests class are run
3					
4					

Post condition(s) for Test:

The AssertEquals test passes where the getRedistributedVotes() method equals 0, which is the value that should be returned after the resetRedistributedVotes() is called.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: **Unit** _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 6

Name(s) of Testers: Steven Petzold

Test Description:

Testing the setElimination() method for the Candidate class. Also checks if the isEliminated() method returns the correct value.

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

Automated: **yes** __X__ **no** ___

Project1/src/CandidateTests.java

Results: Pass ___X__ Fail_____

Preconditions for Test: A candidate has been created with a valid candidate name and party name.

Step # 1	Test Step Description	Test Data	Expected Result	Actual Result	Notes
2	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Candidate**	String candidateName = "Kleinberg"; String partyName = "Party A";	setElimination() method will take the value "true" in and set the value accordingly. The assert will check if the value returned by the isEliminated() method returns true.	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the CandidateTests class are run
3	See step above		setElimination() method will take the value "false" in and set the value accordingly. The assert will check if the value returned by the isEliminated() method returns false.	Assert returned false **Pass**	See step above
4					

Post condition(s) for Test:

The AssertTrue and AssertFalse tesst pass where the setElimination() value is tested with the isEliminated() method according to the respective values set.

Party Tests:

Unit Tests:

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 7

Name(s) of Testers: Steven Petzold

Test Description:

Testing the getter method for the party name attribute of the party class

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

Project1/src/PartyTests.java

Automated: yes __X_ **no** ___

Results: Pass ___X___ **Fail** _____

Preconditions for Test: A Party has been created with a valid party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** 2 **java -cp**	String partyName = "PartyTest";	getPartyName() should return "PartyTest" and the test should return True.	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the PartyTests class are run

	./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party				
3					
4					

Post condition(s) for Test:

The AssertEquals test passes where the getPartyName() method equals the original party name that was created and assigned to the variable partyName.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 8

Name(s) of Testers: Steven Petzold

Test Description:

Testing the adder and getter method for the candidate attribute of the party class

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

Project1/src/PartyTests.java

Automated: yes X **no**

Results: Pass X **Fail**

Preconditions for Test: A Party has been created with a valid party name. Candidates have been created with valid candidate names and party names.

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

1					
2	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party**	Party testParty = new Party(partyName); Candidate testCandidate = new Candidate("TestCandidate", partyName); Candidate testCandidate2 = new Candidate("TestCandidate", partyName); Candidate testCandidate3 = new Candidate("TestCandidate", partyName);	getCandidates() should return the list of candidate objects associated with the party. We then use the get() method associated with Java lists to obtain the name of the candidate at the specified index. The Assert will check if the values of the names of the candidates are equal to the candidate names we set to each variable in the test data	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the PartyTests class are run
3					
4					

Post condition(s) for Test:

The AssertEquals test passes where the candidates that have been added to the candidate list inside the party, equals the values the getCandidate() method returns.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 9

Name(s) of Testers: Steven Petzold

Test Description:

Testing the incrementPartyVote() method and getTotalPartyVotes() methods for the party class

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

Project1/src/PartyTests.java

Automated: yes _X_ **no** ___

Results: Pass ___X__ Fail_____

Preconditions for Test: A Party has been created with a valid party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party**	String partyName = "PartyTest";	The test for getTotalPartyVotes() returns true when the value returned is 2. This is because incrementPartyVote() is called twice.	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the PartyTests class are run
2					
3					
4					

Post condition(s) for Test:

The AssertEquals test passes when the value returned by getTotalPartyVotes() equals 2 because incrementPartyVote() is called twice.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ System __

Test Date:

11/8/23

Test Case ID#: 10

Name(s) of Testers: Steven Petzold

Test Description:

Testing the setPartyVote() method and getTotalPartyVotes() methods for the party class

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

Project1/src/PartyTests.java

Automated: yes__X_ no ___

Results: Pass___X___ Fail_____

Preconditions for Test: A Party has been created with a valid party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party**	String partyName = "PartyTest";	The test for getTotalPartyVotes() returns true when the value returned is 5. This is because setPartyVote() was passed "5" as a parameter, thus setting the number of party votes to 5.	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the PartyTests class are run
2					
3					
4					

Post condition(s) for Test:

The AssertEquals test passes when the value returned by getTotalPartyVotes() equals 5 because setPartyVote() is passed the parameter "5" which sets the number of party votes to 5.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 11

Name(s) of Testers: Steven Petzold

Test Description:

Testing the incrementNumSeatsAllocated() method and getNumSeatsAllocated() methods for the party class

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

Project1/src/PartyTests.java

Automated: yes X **no**

Results: Pass X **Fail**

Preconditions for Test: A Party has been created with a valid party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party**	String partyName = "PartyTest";	The test for getNumSeatsAllocated() returns true when the value returned is 3. This is because incrementNumSeatsAllocated() is given the value '3' to set to the number of seats allocated for the party object. .	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the PartyTests class are run
2					
3					
4					

Post condition(s) for Test:

The AssertEquals test passes when the value returned by getNumSeatsAllocated() equals 3 because incrementNumSeatsAllocated is given the value”3”.

Project Name: Project1 - VoteEasy **Team#** 8

Test Stage: Unit _X_ **System** __

Test Date:

11/8/23

Test Case ID#: 12

Name(s) of Testers: Steven Petzold

Test Description:

Testing the setInitialPartyVotes() method and getInitialPartyVotes() methods for the party class

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

Project1/src/PartyTests.java

Automated: yes __X_ **no** ___

Results: Pass ___X___ **Fail** _____

Preconditions for Test: A Party has been created with a valid party name.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:.**	String partyName = "PartyTest";	The test for getInitalPartyVotes() returns true when the value returned is 10. This is because setInitialPartyVotes() is given the value ‘10’ to set to the number of initial votes allocated for the party object. .	Assert returned True **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the PartyTests class are run
2					

	RunAllUnitTestCases.java **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party**				
3					
4					

Post condition(s) for Test:

The AssertEquals test passes when the value returned by getInitialPartyVotes() equals 10 because setInitialPartyVotes() is given the value”10”.

FileParser Tests:

Unit Tests:

Project Name: Project1 - VoteEasy **Team# 8**

Test Stage: Unit _X_ System __

Test Date:

11/8/23

Test Case ID#: 13

Name(s) of Testers: Carlos Chasi-Mejia

Test Description:
This test ensures that the File Header of the election file is parsed and stored correctly so that we can properly retrieve the fileHeader attribute of the FileParser class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file “FileParserTests.java” is in the Project1/src/ directory. The unit test method is “fileHeaderTest()”.

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: The test file associated with this test case is already present in the "testing" directory and the first line contains the valid file header i.e. the voting protocols: "IR", "OPL".

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_IR_file.csv	getFileHeader() should return the string "IR".	getFileHeader() returns the string "IR". **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run
2	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_OPL_file.csv	getFileHeader() should return the string "OPL".	getFileHeader() returns the string "OPL". **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run

Post condition(s) for Test:
The string returned from getFileHeader() is a string of the only two voting protocols: "IR" and "OPL", which is later used to process the number of ballots to determine a winner.

Project Name: Project1 - VoteEasy **Team# 8**

Test Stage: Unit _X_ System __ **Test Date:**

11/8/23

Test Case ID#: 14
Name(s) of Testers: Carlos Chasi-Mejia

Test Description:
This test ensures that the number of candidates in the election file is parsed and stored correctly so that we can properly retrieve the numberOfCandidates attribute of the FileParser class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file "FileParserTests.java" is in the Project1/src/ directory. The unit test method is "numberOfCandidatesTest()".

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: The test file associated with this test case is already present in the "testing" directory and the second line contains the actual number of candidates in the election.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_IR_file.csv	getNumberOfCandidates() should return theF int 4.	getNumberOfCandidates() returns the int 4. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run

2	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_OPL_file.csv	getNumberOfCandidates() should return the int 6.	getNumberOfCandidates() returns the int 6. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run

Post condition(s) for Test:
The integer returned from getNumberOfCandidates() is a non-negative integer and accurately represents the number of candidates in the very beginning of the election.

Project Name: Project1 - VoteEasy **Team# 8**

Test Stage: Unit _X_ System __

Test Date:

11/8/23

Test Case ID#: 15

Name(s) of Testers: Carlos Chasi-Mejia

Test Description:
This test ensures that the candidate line in the election file is parsed and stored correctly so that we can properly retrieve the candidateLine attribute of the FileParser class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file “FileParserTests.java” is in the Project1/src/ directory. The unit test method is “candidateLineTest()”.

Automated: yes_X__ no ___

Results: Pass ___X__ Fail _____

Preconditions for Test: The test file associated with this test case is already present in the “testing” directory and the third line contains the actual names and party of each candidate in the election in the format: Name (Party)

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_IR_file.csv	getCandidateLine() should return the String “Rosen (D), Kleinberg (R), Chou (I), Royce (L)”	getCandidateLine() returns the String “Rosen (D), Kleinberg (R), Chou (I), Royce (L)”. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run
2	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_OPL_file.csv	getCandidateLine() should return the String "Pike (D), Foster (D), Deutsch (R), Borg (R), Jones (R), Smith (I)"	getCandidateLine() returns the String "Pike (D), Foster (D), Deutsch (R), Borg (R), Jones (R), Smith (I)". **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run

Post condition(s) for Test:
The String returned from getCandidateLine() contains the name and party for all candidates in the election.

Project Name: Project1 - VoteEasy **Team# 8**

Test Stage: Unit _X_ System __

Test Date:

11/8/23

Test Case ID#: 16

Name(s) of Testers: Carlos Chasi-Mejia

Test Description:
This test ensures that the number of ballots in the election file is parsed and stored correctly so that we can properly retrieve the numberOfBallots attribute of the FileParser class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file "FileParserTests.java" is in the Project1/src/ directory.
The unit test method is "numberOfBallotsTest()".

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: The test file associated with this test case is already present in the "testing" directory and the line which contains the actual number of ballots is on line 4 if line 1 is "IR", or line 5 if line 1 is "OPL".

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_IR_file.csv	getNumberOfBallots() should return the int 6	getNumberOfBallots() returns the int 6. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run
2	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_OPL_file.csv	getNumberOfBallots() should return the int 9.	getNumberOfBallots() returns the int 9. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run

Post condition(s) for Test:
The integer returned from getNumberOfBallots() is a non-negative integer and accurately represents the number of ballots in the very beginning of the election.

Project Name: Project1 - VoteEasy **Team# 8**

Test Stage: Unit _X_ System __

Test Date:

11/8/23

Test Case ID#: 17

Name(s) of Testers: Carlos Chasi-Mejia

Test Description:
This test ensures that the number of seats in the election file is parsed and stored correctly so that we can properly retrieve the numberOfSeats attribute of the FileParser class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file "FileParserTests.java" is in the Project1/src/ directory. The unit test method is "numberOfSeatsTest()".

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: The test file associated with this test case is already present in the "testing" directory and actual number of seats should be on line 4 if line 1 is "OPL"; otherwise there is no line that contains the number of seats.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_IR_file.csv	getNumberOfSeats() should return the int 0.	getNumberOfSeats() returns the int 0. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run

2	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_OPL_file.csv	getNumberOfSeats() should return the int 3.	getNumberOfSeats() returns the int 3. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run

Post condition(s) for Test:
The integer returned from getNumberOfSeats() is a non-negative integer and accurately represents the number of seats available in the very beginning of only the OPL election.

Project Name: Project1 - VoteEasy **Team# 8**

Test Stage: Unit _X_ System __

Test Date:

11/8/23

Test Case ID#: 18

Name(s) of Testers: Carlos Chasi-Mejia

Test Description:
This test ensures that all the ballot information in the election file is parsed and stored correctly into an array list so that we can properly retrieve the BallotList attribute of the FileParser class.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file “FileParserTests.java” is in the Project1/src/ directory. The unit test method is “BallotListTest()”.

Automated: yes_X__ no ___

Results: Pass ___X___ Fail _____

Preconditions for Test: The test file associated with this test case is already present in the “testing” directory and the first ballot should be on line 6 if line 1 is “OPL”; or line 5 if line 1 is “IR”. The ballot format is also accurate according to the file’s file header.

Step	Test Step	Test	Expected	Actual	

#	Description	Data	Result	Result	Notes
1	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_IR_file.csv	getBallotList() should return the ArrayList: [“1,3,4,2”, “1,,2,”, “1,2,3,”, “3,2,1,4”, “,,1,2”, “,,,1”]	getNumberOfSeats() returns the ArrayList: [“1,3,4,2”, “1,,2,”, “1,2,3,”, “3,2,1,4”, “,,1,2”, “,,,1”] **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run
2	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases FileParser**	./testing/test_file_parser_OPL_file.csv	getBallotList() should return the ArrayList: [“1,,,,,”, “1,,,,,”, “,1,,,,”, “,,,,1,”, “,,,,,1”, “,,,1,,”, “,,,1,,”, “1,,,,,”, “,1,,,,”]	getNumberOfSeats() returns the Array List: [“1,,,,,”, “1,,,,,”, “,1,,,,”, “,,,,1,”, “,,,,,1”, “,,,1,,”, “,,,1,,”, “1,,,,,”, “,1,,,,”] **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the FileParserTests class are run

Post condition(s) for Test:
The ArrayList returned from getBallotList() contains accurate string representations of every ballot’s information from the election.

IRVoting tests:

Unit Tests:

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/09/2023**

Test Case ID#: 19 **Name(s) of Testers:** Jashwin Acharya

Test Description: This test ensures that Ballots are parsed correctly and candidates are ranked according to a voter's preference for each ballot. For example: If we have 4 candidates: Rosen, Kleinberg, Chou and Royce and one of the ballots has the form 1,3,2,4, then we should ensure that after the ballot is processed, the ordering of candidates for this ballot is [Rosen,Royce,Kleinberg,Chou].

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testBallotDistributionOrder()".

Automated: yes_X__ no ___

Results: Pass ___X___ Fail _____

Preconditions for Test: The test file associated with this test case (IR_test_ballot_distribution_order.txt) is already present in the "testing" directory for ease of use. We do not take any user input or CSV file for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	./testing/IR_test_ballot_distribution_order.txt	All created ballots should have candidates ordered correctly.	All created ballots have candidates ordered correctly. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run.

Post condition(s) for Test:

The list of ballots is created where each element of ballots is an arraylist of candidates sorted by voter preference and it can contain nulls if there no preference has been set by a voter for a certain position.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/09/2023**

Test Case ID#: 20
Test Description: This unit test checks if the correct winner is found when a majority occurs in the first round of calculations itself.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testFirstRoundMajorityWinner()".

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: The CSV file associated with this test case (IR_test_first_round_majority_winner.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	./testing/IR_test_first_round_majority_winner.csv	The winning candidate should be "Rosen" and the number of votes Rosen received should be 4.	The winning candidate chosen is "Rosen" and the number of votes Rosen received is 4. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate is chosen by the system and the assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: **Unit _X_** **System __**

Test Date: **11/09/2023**

Test Case ID#: **21**
Test Description: In this test case, we check if the correct candidate is declared as the winner in the event that only one candidate receives all the votes resulting in a landslide victory.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testLandslideMajority()".

Automated: **yes X** **no**

Results: **Pass ___X__** **Fail_____**

Preconditions for Test: The CSV file associated with this test case (IR_test_landslide_majority.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands:	/testing/IR_test_landslide_majority.csv	The winning candidate should be "Rosen" and the	The winning candidate chosen is "Rosen" and the	Once you execute the junit testing commands defined in the test step description, **all**

	javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**		number of votes Rosen received should be 6.	number of votes Rosen received is 6. **Pass**	unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate is chosen by the system and the assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __

Test Date: 11/09/2023

Test Case ID#: 22
Test Description: In this test case, we check if the vote calculations are correct when only one candidate is participating in the election, in which case, that candidate gets all the votes.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testOnlyOneCandidateMajority()".

Automated: yes X no __

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (IR_test_only_one_candidate_majority.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	/testing/IR_test_only_one_candidate_majority.csv	The winning candidate should be “Rosen” and the number of votes Rosen received should be 6.	The winning candidate chosen is “Rosen” and the number of votes Rosen received is 6. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate is chosen by the system and the assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __

Test Date: 11/09/2023

Test Case ID#: 23

Test Description: This function tests if a two-candidate tie is resolved fairly after a first-round majority is not found. By "fairly" we mean that we will simulate the tie scenario 1000 times to ensure the randomizer is not biased towards one candidate. If the percentage of times a candidate wins a tie is between 45-55%, then we can assume that the tie is fair since the chances of it being exactly 50% every time is quite low.

Name(s) of Testers: Jashwin Acharya

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

The test file is stored in the Project1/src directory and is named “IRVotingTests.java”. The unit test method name is “testTwoCandidateTieAfterNoFirstRoundMajority()”.

Automated: yes X no __

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (IR_test_two_candidate_tie.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	/testing/IR_test_two candidate_tie.csv	Rosen and Kleinberg each have a 45-55% chance of being chosen as a tie breaker winner.	Rosen and Kleinberg each are chosen as the tie winners of the election 45-55% of the time after simulating the tie breaker 1000 times. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The tie breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/09/2023**

Test Case ID#: 24 **Name(s) of Testers:** Jashwin Acharya

Test Description: This function tests if a four candidate tie is resolved fairly after a first round majority is not found. By "fairly" we mean that we will simulate the tie scenario 1000 times to ensure the randomizer is not biased towards one candidate. Every candidate has about a 20-30% chance of winning in order

to ensure fairness in the tie breaker.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testFourCandidateTieAfterNoFirstRoundMajority()".

Automated: yes_X__ no ___

Results: Pass ___X___ Fail _____

Preconditions for Test: The CSV file associated with this test case (IR_test_four_candidate_tie.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	/testing/IR_test_four_candidate_tie.csv	Rosen, Kleinberg, Chou and Royce each have a 20-30% chance of being chosen as a tie breaker winner.	Rosen, Kleinberg, Chou and Royce each are chosen 20-30% of the time as the winner of a tie breaker which was simulated 1000 times. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The tie breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/09/2023**

Test Case ID#: 25
Test Description: In this test case, we check if the correct winner is chosen after only one round of redistribution. During the one redistribution round, candidate Kleinberg is eliminated for having 0 votes and candidate Royce is eliminated and the one ballot assigned to Royce is deleted since it doesn't contain any valid candidates to assign votes to, leading Rosen to win because of a majority.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testMajorityWinnerAfterOneRoundRedistribution()".

Automated: yes_X__ no ___

Results: Pass ___X___ Fail _____

Preconditions for Test: The CSV file associated with this test case (IR_test_majority_winner_after_one_round_redistribution.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	/testing/IR_test_majority_winner_after_one_round_redistribution.csv	The winning candidate should be "Rosen" after redistribution is complete and the number of votes Rosen received should be 3.	The winning candidate chosen is "Rosen" after redistribution is complete and the number of votes Rosen received is 3. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate is chosen by the system after vote redistribution is complete and all assert statements pass successfully.

Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/09/2023**

Test Case ID#: 26
Test Description: In this test case, we check if the correct winner is chosen after redistribution is complete. Despite Rosen leading in the beginning after the first round of ballot calculations, Kleinberg ends up winning once redistribution is complete.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testOneLeadingCandidateLossAfterRedistribution()".

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: The CSV file associated with this test case (IR_test_one_leading_candidate_loss_after_redistribution.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	/testing/IR_test_one_leading_candidate_loss_after_redistribution.csv	The winning candidate should be "Kleinberg" after redistribution is complete and the number of votes Kleinberg received should be 6.	The winning candidate chosen is "Kleinberg" after redistribution is complete and the number of votes Kleinberg received is 6. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate is chosen by the system after vote redistribution is complete and all assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: **Unit _X_** **System __** **Test Date:** **11/09/2023**

Test Case ID#: 27
Test Description: In this test case, we check if the correct winner is chosen after redistribution is complete. For this test case, Rosen and Royce are tied at the highest number of votes initially, but neither have a majority yet. After redistribution, Chou ends up winning the election.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testTwoLeadingCandidateLossAfterRedistribution()".

Automated: yes X no __

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (IR_test_two_leading_candidate_loss_after_redistribution.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

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

1	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	/testing/IR_test_two_leading_candidate_loss_after_redistribution.csv	The winning candidate should be “Chou” after redistribution is complete and the number of votes Chou received should be 6.	The winning candidate chosen is “Chou” after redistribution is complete and the number of votes Chou received is 6. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate is chosen by the system after vote redistribution is complete and all assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/09/2023**

Test Case ID#: 28
Test Description: Test for when a tie is found after one round of redistribution itself. We ensure that each candidate has a fair chance of winning a tie and thus perform the tie 1000 times. Every tied candidate has about a 45-55% chance of winning in order to ensure fairness in the tie-breaker.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named “IRVotingTests.java”. The unit test method name is “testCandidateTieAfterOneRoundOfRedistribution()”.

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (IR_test_candidate_tie_after_one_round_of_redistribution.csv) is already present in the “testing” directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	/testing/IR_test_candidate_tie_after_one_round_of_redistribution.csv	Rosen and Chou are the tied candidates and are chosen 45-55% of the time after simulating the tie-breaker 1000 times.	Rosen and Chou do win the tie-breaker 45-55% of the time each after simulating the tie-breaker 1000 times. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

All tie-breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: **Unit _X_** **System __**

Test Date: **11/09/2023**

Test Case ID#: **29**

Test Description: Test for when a tie is found after multiple rounds of redistribution. We ensure that each candidate has a fair chance of winning a tie and thus perform the tie 1000 times. Every tied candidate has about a 45-55% chance of winning in order to ensure fairness in the tie-breaker.

Name(s) of Testers: Jashwin Acharya

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

The test file is stored in the Project1/src directory and is named "IRVotingTests.java". The unit test method name is "testCandidateTieAfterMultipleRoundsOfRedistribution()".

Automated: **yes_X__** **no ___**

Results: Pass ___X__ Fail_____

Preconditions for Test: The CSV file associated with this test case (IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases IRVoting**	/testing/IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv	Rosen and Kleinberg are the tied candidates after multiple rounds of redistribution and are chosen 45-55% of the time after simulating the tie-breaker 1000 times.	Rosen and Chou do win the tie-breaker 45-55% of the time each after simulating the tie-breaker 1000 times and performing multiple rounds of redistribution. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the IRVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

All tie-breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

System Tests:

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit __ System _X_ **Test Date: 11/10/2023**

Test Case ID#: 30 **Name(s) of Testers:** Jashwin Acharya

Test Description: This test ensures that the user is able to enter the correct name of the CSV file and initiate ballot calculations using the **Instant Runoff (IR) Voting Protocol** where a candidate wins in the first round of calculations itself**.** Once the test is complete, an audit file should be generated and winner information is displayed on the screen to the user.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
This is a manual system test and no test file has been defined for this test. All the checks will be done manually by the user.

Automated: yes no X

Results: Pass X Fail

Preconditions for Test: The test file associated with this test case (IR_test_first_round_majority_winner.csv) should be copied from the "testing" directory to the "src" directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Copy **IR_test_first_round_majority_winner.csv** from the "/testing" to "/src" directory manually using copy-paste.		**IR_test_first_round_majority_winner.csv** should be present in the "src" directory	**IR_test_first_round_majority_winner.csv** is present in the "src" directory.	
2	Open a terminal in the "src" directory and execute the following two commands: **javac VoteEasy.java** **java VoteEasy**	./src/IR_test_first_round_majority_winner.csv	Welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	VoteEasy welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	
3	User enters the name "IR_test_first_round_majority_winner.csv" when prompted for the file name by the system and presses ENTER on their keyboard.	./src/IR_test_first_round_majority_winner.csv	If the correct file name is entered, then no error message is shown and ballot calculations are performed. If the incorrect file name is entered or if the file is missing from the "src" directory, the user is	The correct file name is entered and ballot calculations are initiated. If the incorrect file name is entered, then the user is prompted to enter the name indefinitely until the right name is entered.	You can run this test multiple times to test what happens when the incorrect file name is entered or the

			prompted indefinitely until they enter the right name.		file is missing.
4	User can see the winner details displayed on the screen.	./src/IR_test_first_round_majority_winner.csv	Once the right file name is entered, the user should see the winner details displayed on the screen along with the number / % of votes each candidate received.	The winner details along with the number / % of votes each candidate received is displayed on the screen with appropriate formatting.	
5	Navigate to the /src directory and check for the “audit_file.txt” file		The user should see an audit file with the name “audit_file.txt” created in the src directory.	The audit file named “audit_file.txt” is created in the src directory	
6	Open the audit file and ensure the Voting Protocol name is the first line of the file.		The voting protocol name on the first line should be “Instant Runoff (IR)”	The voting protocol name mentioned on the first line of the file is “Instant Runoff (IR)”	
7	Observe lines 3-9 of the audit file for first round of voting calculation information.		The user should see the results after the first round of voting calculation where a majority is achieved for Rosen and should see the final result of the vote calculations in a tabular format below which Rosen is declared as the winner.	Once the audit file is opened, the first round of ballot calculation information is shown in a tabular format, below which you can see the final voting results also in a tabular format, followed by Rosen’s party affiliation details and the number of votes they received.	
8	Ensure that the vote calculations have been done correctly at lines 3-9 of the audit file.		Rosen should have received 4 total votes while Kleinberg should receive 0 votes and Chou and Royce are tied at 1 vote each.	On manual inspection, it was confirmed that Rosen did receive 4 votes and Kleinberg had 0 votes while Chou and Royce were tied at 1 vote each.	
9	Observe the last line of the audit file to verify the winner information.		The last line of the audit file should show Rosen as the winner with 6 votes to their name	The last line of the audit file does show Rosen as the winner with 6 votes to their name. **Pass**	

Post condition(s) for Test:

The program should exit gracefully once winner information is displayed on the screen and the audit file is generated.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit __ System _X_ **Test Date: 11/10/2023**

Test Case ID#: 31
Test Description: This test ensures that the user is able to enter the correct name of the CSV file and initiate ballot calculations using the **Instant Runoff (IR) Voting Protocol** where one of two candidates is chosen as the winner of the tie-breaker.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
This is a manual system test and no test file has been defined for this test. All the checks will be done manually by the user.

Automated: yes no X

Results: Pass X Fail

Preconditions for Test: The test file associated with this test case (IR_test_two_candidate_tie.csv) should be copied from the "testing" directory to the "src" directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Copy **IR_test_two_candidate_tie.csv** from the "/testing" to "/src" directory manually using copy-paste.		**IR_test_two_candidate_tie.csv** should be present in the "src" directory	**IR_test_two_candidate_tie.csv** is present in the "src" directory.	
2	Open a terminal in the "src" directory and execute the following two commands: **javac VoteEasy.java**	./src/IR_test_two_candidate_tie.csv	Welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	VoteEasy welcome messages are displayed on screen and user is prompted to enter the name of the	

	java VoteEasy			CSV file.	
3	User enters the name “IR_test_two_candidate_tie.csv” when prompted for the file name by the system and presses ENTER on their keyboard.	./src/IR_test_two_candidate_tie.csv	If the correct file name is entered, then no error message is shown and ballot calculations are performed. If the incorrect file name is entered or if the file is missing from the “src” directory, the user is prompted indefinitely until they enter the right name.	The correct file name is entered and ballot calculations are initiated. If the incorrect file name is entered, then the user is prompted to enter the name indefinitely until the right name is entered.	You can run this test multiple times to test what happens when the incorrect file name is entered or the file is missing.
4	User can see the winner details displayed on the screen.	./src/IR_test_two_candidate_tie.csv	Once the right file name is entered, the user should see the winner details displayed on the screen along with the number / % of votes each candidate received.	The winner details along with the number / % of votes each candidate received is displayed on the screen with appropriate formatting.	
5	Navigate to the /src directory and check for the “audit_file.txt” file		The user should see an audit file with the name “audit_file.txt” created in the “src” directory.	The audit file named “audit_file.txt” is created in the “src” directory	
6	Open the audit file and ensure the Voting Protocol name is the first line of the file.		The voting protocol name on the first line should be “Instant Runoff (IR)”	The voting protocol name mentioned on the first line of the file is “Instant Runoff (IR)”	
7	Observe lines 3-9 of the audit file for first round of voting calculation information.		Lines 3-9 should show the results after the first round of ballot calculations where Rosen and Kleinberg are tied.	Once the audit file is opened, Lines 3-9 do show the results after the round of ballot calculations and both Rosen and Kleinberg are tied at 2 votes each.	
8	Ensure that the vote calculations have been done correctly at lines 3-9 of the audit file.		Rosen and Kleinberg should have received 2 votes each while Chou and Royce should have received 0 votes each. (Lines 18-24)	Lines 18-24 of the audit file show that Rosen and Kleinberg received 2 votes each while Chou and Royce both received 0 votes.	
9	Observe Lines 11-16 that display the names of the tied candidates as well as the winner of the tie.		Lines 13 and 14 should display “Rosen” and “Kleinberg” and Line 16 should have one of the	Lines 13 and 14 of the audit file list the names “Rosen” and “Kleimberg” as the currently tied candidates, and Rosen is displayed as	Since the result of a tie breaker is random, Kleinberg can also be

			two names listed as the winner.	the winner of the tie breaker on Line 16.	displayed on Line 16 instead of Rosen. Eitherways, the test should succeed.
10	Ensure the winner displayed on Line 16 and Line 26 is the same person.		Line 16 and 26 should have either Rosen or Kleinberg's name as the declared winner with 2 votes.	Lines 16 and 26 of the audit file show Rosen as the winner with 2 votes. **Pass**	Again, Kleinberg could be displayed too if the test is run multiple times. Eitherways, the test should succeed.

Post condition(s) for Test:

The program should exit gracefully once winner information is displayed on the screen and the audit file is generated.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit __ System _X_ **Test Date: 11/10/2023**

Test Case ID#: 32
Test Description: This test ensures that the user is able to enter the correct name of the CSV file and initiate ballot calculations using the **Instant Runoff (IR) Voting Protocol** where redistribution is performed multiple times and one candidate wins because of a majority.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
This is a manual system test and no test file has been defined for this test. All the checks will be done manually by the user.

Automated: yes___ no _X__

Results: Pass ___X___ Fail _____

Preconditions for Test: The test file associated with this test case (IR_test_two_leading_candidate_loss_after_redistribution.csv) should be copied from the "testing" directory to the "src" directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Copy **IR_test_two_leading_candidate_loss_after_redistribution.csv** from the "/testing" to "/src" directory manually using copy-paste.		**IR_test_two_leading_candidate_loss_after_redistribution.csv** should be present in the "src" directory	**IR_test_two_leading_candidate_loss_after_redistribution.csv** is present in the "src" directory.	
2	Open a terminal in the "src" directory and execute the following two commands: **javac VoteEasy.java** **java VoteEasy**	./src/IR_test_two_leading_candidate_loss_after_redistribution.csv	Welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	VoteEasy welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	
3	User enters the name "IR_test_two_leading_candidate_loss_after_redistribution.csv" when prompted for the file name by the system and presses ENTER on their keyboard.	./src/IR_test_two_leading_candidate_loss_after_redistribution.csv	If the correct file name is entered, then no error message is shown and ballot calculations are performed. If the incorrect file name is entered or if the file is missing from the "src" directory, the user is prompted indefinitely until they enter the right name.	The correct file name is entered and ballot calculations are initiated. If the incorrect file name is entered, then the user is prompted to enter the name indefinitely until the right name is entered.	You can run this test multiple times to test what happens when the incorrect file name is entered or the file is missing.
4	User can see the winner details displayed on the screen.	./src/IR_test_two_leading_candidate_loss_after_redistribution.csv	Once the right file name is entered, the user should see the winner details displayed on the screen along with the number / % of votes each candidate received.	The winner details along with the number / % of votes each candidate received is displayed on the screen with appropriate formatting.	
5	Navigate to the /src directory and check for the "audit_file.txt" file		The user should see an audit file with the name "audit_file.txt" created in the "src" directory.	The audit file named "audit_file.txt" is created in the "src" directory	
6	Open the audit file and ensure the Voting Protocol name is the first line of the file.		The voting protocol name on the first line	The voting protocol name mentioned on the first line	

			should be “Instant Runoff (IR)”	of the file is “Instant Runoff (IR)”	
7	Observe lines 3-9 of the audit file for first round of voting calculation information.		Lines 3-9 should show the results after the first round of ballot calculations where there is currently no majority and no tie.	Once the audit file is opened, Lines 3-9 do show the results after the round of ballot calculations and no majority or tie has occurred.	
8	Observe first round of redistribution results on lines 11-18 of the audit file.		Line 12 should show that Kleinberg was eliminated in the first round of redistribution and their 2 votes were transferred to Chou. Rosen and Royce’s vote counts should remain the same as before.	Line 12 indicates Kleinberg was indeed eliminated in the first redistribution round and 2 of their votes were transferred to Chou. Rosen and Royce didn’t receive any redistribution votes in this current round.	All redistributed votes can be seen in the “Number of redistributed votes” column
9	Observe second round of redistribution results on lines 20-27 of the audit file.		Line 21 shows that Rosen was eliminated and 2 of Rosen’s votes went to Chou and 1 vote went to Royce.	Line 21 does show that Rosen was eliminated and the table values at lines 26 and 27 indicate that Chou received 2 of Rosen’s votes while Royce received only 1 of Rosen’s votes.	All redistributed votes can be seen in the “Number of redistributed votes” column
10	Observe final round of voting calculation information at lines 29-35 of the audit file.		Only Chou and Royce’s vote counts should be displayed since Rosen and Kleinberg were eliminated.	Only Chou and Royce’s vote counts are displayed and Kleinberg and Rosen are marked as eliminated in the final vote count table at lines 32 and 33 of the audit file.	
11	Observe last line of the audit file for winner information.		Chou should be displayed as the winner with 6 votes to their name.	Chou is displayed as the winner with 6 votes to their name. **Pass**	

Post condition(s) for Test:

The program should exit gracefully once winner information is displayed on the screen and the audit file is generated.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit __ System _X_ **Test Date: 11/10/2023**

Test Case ID#: 33
Test Description: This test ensures that the user is able to enter the correct name of the CSV file and initiate ballot calculations using the **Instant Runoff (IR) Voting Protocol** where redistribution is performed multiple times and two candidates end up in a tie-breaker with only one of them winning the election.

Name(s) of Testers: Jashwin Acharya

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
This is a manual system test and no test file has been defined for this test. All the checks will be done manually by the user.

Automated: yes___ no _X__

Results: Pass X Fail

Preconditions for Test: The test file associated with this test case (IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv) should be copied from the "testing" directory to the "src" directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Copy **IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv** from the "/testing" to "/src" directory manually using copy-paste.		**IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv** should be present in the "src" directory	**IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv** is present in the "src" directory.	
2	Open a terminal in the "src" directory and execute the following two commands: **javac VoteEasy.java**	./src/IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv	Welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	VoteEasy welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	

	java VoteEasy				
3	User enters the name “IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv” when prompted for the file name by the system and presses ENTER on their keyboard.	./src/IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv	If the correct file name is entered, then no error message is shown and ballot calculations are performed. If the incorrect file name is entered or if the file is missing from the “src” directory, the user is prompted indefinitely until they enter the right name.	The correct file name is entered and ballot calculations are initiated. If the incorrect file name is entered, then the user is prompted to enter the name indefinitely until the right name is entered.	You can run this test multiple times to test what happens when the incorrect file name is entered or the file is missing.
4	User can see the winner details displayed on the screen.	./src/IR_test_candidate_tie_after_multiple_rounds_of_redistribution.csv	Once the right file name is entered, the user should see the winner details displayed on the screen along with the number / % of votes each candidate received.	The winner details along with the number / % of votes each candidate received is displayed on the screen with appropriate formatting.	
5	Navigate to the /src directory and check for the “audit_file.txt” file		The user should see an audit file with the name “audit_file.txt” created in the “src” directory.	The audit file named “audit_file.txt” is created in the “src” directory	
6	Open the audit file and ensure the Voting Protocol name is the first line of the file.		The voting protocol name on the first line should be “Instant Runoff (IR)”	The voting protocol name mentioned on the first line of the file is “Instant Runoff (IR)”	
7	Ensure that initial round of voting calculations completed successfully.		Lines 3-9 should show the results after the first round of ballot calculations where there is currently no majority and no tie.	Once the audit file is opened, Lines 3-9 do show the results after the round of ballot calculations and no majority or tie has occurred.	
8	Observe first round of redistribution results on lines 11-18 of the audit file.		Line 12 should show that Royce was eliminated in the first round of redistribution and their votes were transferred to Rosen. Chou and Kleinberg’s vote counts should remain the same as before.	Line 12 indicates Royce was indeed eliminated in the first redistribution round and Royce’s 1 vote was transferred to Rosen. Chou and Kleinberg didn’t receive any redistribution votes in this current round.	All redistributed votes can be seen in the “Number of redistributed votes” column
9	Observe second round of redistribution		Line 21 shows that Chou	Line 21 does show that	All

	results on lines 20-27 of the audit file.		was eliminated and Rosen and Kleinberge received 1 vote each from Chou's votes during redistribution.	Chou was eliminated and Rosen and Kleinberg received 1 vote each during redistribution.	redistributed votes can be seen in the "Number of redistributed votes" column
10	Observe in the table at lines 20-27 of the audit file that Rosen and Kleinberg are tied		Lines 24-25 of the audit file should indicate that both Rosen and Kleinberg are tied at 4 votes each.	Lines 24-25 do indicate that Rosen and Kleinberg are tied at 4 votes each (50-50 split).	
11	Observe tied candidate names at lines 31 and 32 of the audit file.		Lines 31 and 32 mention explicitly that Rosen and Kleinberg are tied.	Lines 31 and 32 do mention that candidates Rosen and Kleinberg are tied.	
12	Observe the result of the tie breaker at Line 34 of the audit file.		Either Rosen or Kleinberg should be displayed as the tie-breaker winner on line 34.	Line 34 shows that Rosen was the winner of the tie-breaker.	Since the tie-breaker is random, Kleinberg could've won too. Rosen and Kleinberg have an equal chance of winning.
10	Observe final round of voting calculation information at lines 36-42 of the audit file.		Only Rosen and Kleinberg's vote counts should be displayed since Chou and Royce were eliminated.	Only Rosen and Kleinberg's vote counts are displayed and Chou and Royce are marked as eliminated in the final vote count table at lines 39 and 40 of the audit file.	
11	Observe last line of the audit file for winner information.		Rosen or Kleinberg should be displayed as the winner with 4 votes to either of their name's.	Rosen is displayed as the winner with 4 votes to their name. **Pass**	Since this is a tie-breaker result, Kleinberg could also be displayed. Eitherways, the test is successful.

Post condition(s) for Test:

The program should exit gracefully once winner information is displayed on the screen and the audit file is generated.

OPLVoting Tests:

Unit Tests:

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date:** 11/10/2023

Test Case ID#: 34
Test Description: This test ensures that the correct winner for a party and candidate is found for a smaller set of ballots with no ties occurring in this situation.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testSmallNumberOfVotes()".

Automated: yes_X__ no ___

Results: Pass ___X__ Fail _____

Preconditions for Test: The CSV file associated with this test case (OPL_test_small_number_of_votes.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar**	./testing/OPL_test_small_number_of_votes.csv	The winning party should be "D" having "2" seats allocated and receiving "5" votes, and the winning candidate should be "Pike" receiving "3" votes.	The winning party is "D" having "2" seats allocated and receiving "5" votes, and the winning candidate is "Pike" receiving "3" votes. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

	:. RunAllUnitTestCases OPLVoting				

Post condition(s) for Test:

The correct winning candidate and party is chosen by the system and the assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/10/2023**

Test Case ID#: 35
Test Description: This test ensures that the correct winner for a party and candidate is found for a larger set of ballots with no ties occurring in this situation.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testLargeNumberOfVotes()".

Automated: yes_X__ no ___

Results: Pass ___X__ Fail_____

Preconditions for Test: The CSV file associated with this test case (OPL_test_large_number_of_votes.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:.**	./testing/OPL_test_large_number_of_votes.csv	The winning party should be "I" having "2" seats allocated and receiving "9"	The winning party is "I" having "2" seats allocated and receiving "9" votes,	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the

	RunAllUnitTestCases.java **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar :. RunAllUnitTestCases OPLVoting**		votes, and the winning candidate should be "Smith" receiving "9" votes.	and the winning candidate is "Smith" receiving "9" votes. **Pass**	OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate and party is chosen by the system and the assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy

Team# 08

Test Stage: Unit _X_ System __

Test Date: 11/10/2023

Test Case ID#: 36

Test Description: This test ensures that the correct winner for a party and candidate is found when all the votes go towards a single party, with no ties occurring in this situation.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testAllVotesToOneParty()".

Automated: yes_X__ no ___

Results: Pass ___X__ Fail _____

Preconditions for Test: The CSV file associated with this test case (OPL_test_all_votes_to_one_party.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

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

1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases OPLVoting**	./testing/OPL_test_all_votes_to_one_party.csv	The winning party should be "D" having "3" seats allocated and receiving "9" votes, and the winning candidate should be "Foster" receiving "5" votes.	The winning party is "D" having "3" seats allocated and receiving "9" votes, and the winning candidate is "Foster" receiving "5" votes. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate and party is chosen by the system and the assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __

Test Date: 11/10/2023

Test Case ID#: 37

Test Description: This test ensures that the correct winner for a party and candidate is found when all the votes go towards a single candidate, with no ties occurring in this situation.

Name(s) of Testers: Vincent Hoang

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

The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testAllVotesToOneCandidate()".

Automated: yes X no

Results: Pass ___X__ Fail_____

Preconditions for Test: The CSV file associated with this test case (OPL_test_all_votes_to_one_candidate.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases OPLVoting**	./testing/OPL_test_all_votes_to_one_candidate.csv	The winning party should be “R” having “3” seats allocated and receiving “9” votes, and the winning candidate should be “Borg” receiving “9” votes.	The winning party is “R” having “3” seats allocated and receiving “9” votes, and the winning candidate is “Borg” receiving “9” votes. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate and party is chosen by the system and the assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/10/2023**

Test Case ID#: 38
Test Description: This test ensures that the correct winner for a party and candidate is found after the first round of seat allocations, with no ties occurring in this situation.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named “OPLVotingTests.java”. The unit test method name is “testFirstRoundWinner()”.

Automated: yes X no __

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (OPL_test_first_round_winner.csv) is already present in the

"testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases OPLVoting**	./testing/OPL_test_first_round_winner.csv	The winning party should be "D" having "2" seats allocated and receiving "6" votes, and the winning candidate should be "Pike" receiving "4" votes.	The winning party is "D" having "2" seats allocated and receiving "6" votes, and the winning candidate is "Pike" receiving "4" votes. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The correct winning candidate and party is chosen by the system and the assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/10/2023**

Test Case ID#: 39 **Name(s) of Testers:** Vincent Hoang

Test Description: This function tests if a two candidate tie is resolved fairly after the seat allocations and a party winner has been determined, with no parties or remaining votes tie occurring. By "fairly" we mean that we will simulate the tie scenario 1000 times to ensure the randomizer is not biased towards one candidate. If the percentage of times a candidate wins a tie is between 45-55%, then we can assume that the tie is fair since the chances of it being exactly 50% everytime is quite low.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testTwoTiedCandidates()".

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (OPL_test_two_tied_candidates.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases OPLVoting**	./testing/OPL_test_two_tied_candidates.csv	Pike and Foster each have a 45-55% chance of being chosen as the candidate winner from a tie breaker.	Pike and Foster each are chosen as the tie candidate winners of the election 45-55% of the time after simulating the tie breaker 1000 times. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The tie breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/10/2023**

Test Case ID#: 40 **Name(s) of Testers:** Vincent Hoang

Test Description: This function tests if a four candidate tie is resolved fairly after the seat allocations and a party winner has been determined, with no parties or remaining votes tie occurring. By "fairly" we mean that we will simulate the tie scenario 1000 times to ensure the randomizer is not biased towards one candidate. Every candidate has about a 20-30% chance of winning in order to ensure fairness in the tie breaker.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testFourTiedCandidates()".

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (OPL_test_four_tied_candidates.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases OPLVoting**	./testing/OPL_test_four_tied_candidates.csv	Foster, Deutsch, Borg, and Jones each have a 20-30% chance of being chosen as the candidate winner from a tie breaker.	Foster, Deutsch, Borg, and Jones are each chosen as the tie candidate winners of the election 20-30% of the time after simulating the tie breaker 1000 times. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The tie breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/10/2023**

Test Case ID#: 41
Test Description: This function tests if two most remaining vote ties are resolved fairly after the first round of seat allocations, with no parties or candidates tie occurring. By "fairly" we mean that we will simulate the tie scenario 1000 times to ensure the randomizer is not biased towards one party to be allocated a remaining seat. If the percentage of times a party wins a tie to be allocated a remaining seat is between 45-55%, then we can assume that the tie is fair since the chances of it being exactly 50% everytime is quite low.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testTwoRemainderTies()".

Automated: yes_X__ no ___

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (OPL_test_two_remainder_ties.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:.**	./testing/OPL_test_two_remainder_ties.csv	The party D and R each have a 45-55% chance of being chosen as the party	The party D and R are each chosen as the tie party that receives a remaining	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the

	RunAllUnitTestCases.java **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar :. RunAllUnitTestCases OPLVoting**		that receives a remaining seat from a tie breaker.	seat 45-55% of the time after simulating the tie breaker 1000 times. **Pass**	OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The tie breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/10/2023**

Test Case ID#: 42
Test Description: This function tests if four most remaining vote ties are resolved fairly after the first round of seat allocations, with no parties or candidates tie occurring. By "fairly" we mean that we will simulate the tie scenario 1000 times to ensure the randomizer is not biased towards one party to be allocated a remaining seat. Every party has about a 20-30% chance of being allocated a remaining seat in order to ensure fairness in the tie breaker.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testFourRemainderTies()".

Automated: yes X no

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (OPL_test_four_remainder_ties.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the “src” directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases OPLVoting**	./testing/OPL_test_four_remainder_ties.csv	The party D, R, G, and I each have a 20-30% chance of being chosen as the party that receives a remaining seat from a tie breaker.	The party D, R, G, and I are each chosen as the tie party that receives a remaining seat 20-30% of the time after simulating the tie breaker 1000 times. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The tie breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/10/2023**

Test Case ID#: 43
Test Description: This function tests if a two party tie is resolved fairly after the seat allocations and a party winner is being determined, with no candidates or remaining votes tie occurring. By "fairly" we mean that we will simulate the tie scenario 1000 times to ensure the randomizer is not biased towards one party. If the percentage of times a party wins a tie is between 45-55%, then we can assume that the tie is fair since the chances of it being exactly 50% everytime is quite low.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named “OPLVotingTests.java”. The unit test method name is

Automated: yes_X__ no ___

"testTwoTiedParties()".

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (OPL_test_two_tied_parties.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases OPLVoting**	./testing/OPL_test_two_tied_parties.csv	The party D and R each have a 45-55% chance of being chosen as the party winner from a tie breaker.	The party D and R each are chosen as the tie party winners of the election 45-55% of the time after simulating the tie breaker 1000 times. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The tie breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit _X_ System __ **Test Date: 11/10/2023**

Test Case ID#: 44
Test Description: This function tests if a four party tie is resolved fairly after the seat allocations and a party winner is being determined, with no candidates or remaining votes tie

Name(s) of Testers: Vincent Hoang

occurring. By "fairly" we mean that we will simulate the tie scenario 1000 times to ensure the randomizer is not biased towards one party. Every party has about a 20-30% chance of winning in order to ensure fairness in the tie breaker.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
The test file is stored in the Project1/src directory and is named "OPLVotingTests.java". The unit test method name is "testFourTiedParties()".

Automated: yes_X__ no___

Results: Pass X Fail

Preconditions for Test: The CSV file associated with this test case (OPL_test_four_tied_parties.csv) is already present in the "testing" directory for ease of use. We do not take any user input for this unit test.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Open a terminal in the "src" directory and execute the following two commands: **javac -cp ./../lib/junit-4.13.2.jar:. RunAllUnitTestCases.java** **java -cp ./../lib/junit-4.13.2.jar:./../lib/hamcrest-core-1.3.jar:. RunAllUnitTestCases OPLVoting**	./testing/OPL_test_four_tied_parties.csv	The party D, R, G, and I each have a 20-30% chance of being chosen as the party winner from a tie breaker.	The party D, R, G, and I are each chosen as the tie party winners of the election 20-30% of the time after simulating the tie breaker 1000 times. **Pass**	Once you execute the junit testing commands defined in the test step description, **all** unit tests defined in the OPLVotingTests class are run. No audit file is generated for this test.

Post condition(s) for Test:

The tie breaker assert statements pass successfully. Nothing is displayed on screen and an audit file is not generated either since this is a unit test and not a system test.

System Tests:

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit __ System _X_ **Test Date: 11/11/2023**

Test Case ID#: 45
Test Description: This test ensures that the user is able to enter the correct name of the CSV file and perform seat allocations using the **Open Party List (OPL) Voting Protocol** where no ties occur in this situation. Once the test is complete, an audit file should be generated and winner information is displayed on the screen to the user.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
This is a manual system test and no test file has been defined for this test. All the checks will be done manually by the user.

Automated: yes___ no _X__

Results: Pass ___X__ Fail_____

Preconditions for Test: The test file associated with this test case (OPL_test_large_number_of_votes.csv) should be copied from the "testing" directory to the "src" directory

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Copy **OPL_test_large_number_of_votes.csv** from the "/testing" to "/src" directory manually using copy-paste.		**OPL_test_large_number_of_votes.csv** should be present in the "src" directory	**OPL_test_large_number_of_votes.csv** is present in the "src" directory.	
2	Open a terminal in the "src" directory and execute the following two commands: **javac VoteEasy.java** **java VoteEasy**	./src/OPL_test_large_number_of_votes.csv	Welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	VoteEasy welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	

3	User enters the name "OPL_test_large_number_of_votes.csv" when prompted for the file name by the system and presses ENTER on their keyboard.	./src/OPL_test_large_number_of_votes.csv	If the correct file name is entered, then no error message is shown and seat allocations are performed. If the incorrect file name is entered or if the file is missing from the "src" directory, the user is prompted indefinitely until they enter the right name.	The correct file name is entered and ballot calculations are initiated. If the incorrect file name is entered, then the user is prompted to enter the name indefinitely until the right name is entered.	You can run this test multiple times to test what happens when the incorrect file name is entered or the file is missing.
4	User can see the winner details displayed on the screen.	./src/OPL_test_large_number_of_votes.csv	Once the right file name is entered, the user should see the party and candidate winner details, including the winning party's name, number of votes and seats received and the winning candidate's name, its associated party, and the number of votes they received, displayed on the screen along with the number of votes, seats, % of Votes to % of Seats of the parties, and number of votes and % of votes each candidate received.	The party and candidate winner details displayed on the screen, including the winning party's name, number of votes and seats received and the winning candidate's name, its associated party, and the number of votes they received, along with the number of votes, seats, % of Votes to % of Seats of the parties, and number of votes and % of votes each candidate received is displayed on the screen with appropriate formatting.	
5	Navigate to the /src directory and check for the "audit_file.txt" file		The user should see an audit file with the name "audit_file.txt" created in the src directory.	The audit file named "audit_file.txt" is created in the src directory	
6	Open the audit file and ensure the Voting Protocol name is the first line of the file.		The voting protocol name on the first line should be "Open Party List (OPL)"	The voting protocol name mentioned on the first line of the file is "Open Party List (OPL)"	
7	Observe lines 5-8 of the audit file before the first round of seat allocations.		The user should see the initial results for each party before the first round of seat allocations in a tabular format where the parties' number of votes and the number of	Once the audit file is opened, the initial results before the first round of seat allocations information is shown in a tabular format, in which you can see the parties' number of votes and the number of	

			seats they have allocated are listed.	seats they have allocated are listed.	
8	Ensure that the seat allocations have been done correctly at lines 5-8 of the audit file.		D should have 6 votes, R should have 5 votes, and I should have 9 votes. D, R, and I should all have 0 seats allocated.	On manual inspection, it was confirmed that D has 6 votes, R has 5 votes, and I has 9 votes. D, R, and I have 0 seats allocated.	
9	Observe lines 12-15 of the audit file for the first round of seat allocations.		The user should see the results after the first round of seat allocations in a tabular format where the parties' remaining votes and the number of seats they have allocated are listed.	The first round of seat allocations information is shown in a tabular format, in which you can see the parties' remaining votes and the number of seats they have allocated are listed.	
10	Ensure that the seat allocations have been done correctly at lines 12-15 of the audit file.		D should have 1 remaining vote, R should have 0 remaining votes, and I should have 4 remaining votes. D, R, and I should all have 1 seat allocated.	On manual inspection, it was confirmed that D has 1 remaining vote, R has 0 remaining votes, and I has 4 remaining votes. D, R, and I also have 1 seat allocated.	
11	Observe lines 19-22 of the audit file for the second round of seat allocations.		The user should see the results after the second round of seat allocations in a tabular format where the parties' remaining votes and the number of seats they have allocated are listed.	Looking below the previous round, the second round of seat allocations information is shown in a tabular format, in which you can see the parties' remaining votes and the number of seats they have allocated are listed.	
12	Ensure that the seat allocations have been done correctly at lines 19-22 of the audit file.		D should have 1 remaining vote, R should have 0 remaining votes, and I should have 4 remaining votes. D and R should have 1 seat allocated and I should have 2 seats allocated.	On manual inspection, it was confirmed that D has 1 remaining vote, R has 0 remaining votes, and I has 4 remaining votes. D and R have 1 seat allocated and I have 2 seats allocated.	
13	Ensure that line 31 displays the correct party winner.		Line 31 should display the I party as the party winner with 9 votes and 2 seats allocated.	Line 31 displays the I party as the party winner with 9 votes and 2 seats allocated.	

14	Ensure that line 36 displays the correct candidate winner.		Line 36 should display Smith as the candidate winner from I party with 9 votes.	Line 36 displays Smith as the candidate winner from I party with 9 votes.	
15	Observe lines 26-36 of the audit file for the results after the seat allocations have been completed.		Lines 26-36 should match what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)"	Lines 26-36 matches with what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)" **Pass**	

Post condition(s) for Test:

The program should exit gracefully once winner information is displayed on the screen and the audit file is generated.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit __ System _X_ **Test Date: 11/11/2023**

Test Case ID#: 46
Test Description: This test ensures that the user is able to enter the correct name of the CSV file and initiate ballot calculations using the **Open Party List (OPL) Voting Protocol** where one of two candidates is chosen as the winner of the tie-breaker.

Name(s) of Testers: Vincent Hoang

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
This is a manual system test and no test file has been defined for this test. All the checks will be done manually by the user.

Automated: yes___ no _X__

Results: Pass ___X__ Fail_____

Preconditions for Test: The test file associated with this test case (OPL_test_two_tied_candidates.csv) should be copied from the "testing" directory to the "src" directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Copy **OPL_test_two_tied_candidates.csv** from the "/testing" to "/src" directory manually using copy-paste.		**OPL_test_two_tied_candidates.csv** should be present in the "src" directory	**OPL_test_two_tied_candidates.csv** is present in the "src" directory.	
2	Open a terminal in the "src" directory and execute the following two commands: **javac VoteEasy.java** **java VoteEasy**	./src/OPL_test_two_tied_candidates.csv	Welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	VoteEasy welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	
3	User enters the name "OPL_test_two_tied_candidates.csv" when prompted for the file name by the system and presses ENTER on their keyboard.	./src/OPL_test_two_tied_candidates.csv	If the correct file name is entered, then no error message is shown and ballot calculations are performed. If the incorrect file name is entered or if the file is missing from the "src" directory, the user is prompted indefinitely until they enter the right name.	The correct file name is entered and ballot calculations are initiated. If the incorrect file name is entered, then the user is prompted to enter the name indefinitely until the right name is entered.	You can run this test multiple times to test what happens when the incorrect file name is entered or the file is missing.
4	User can see the winner details displayed on the screen.	./src/OPL_test_two_tied_candidates.csv	Once the right file name is entered, the user should see the party and candidate winner details, including the winning party's name, number of votes and seats received and the winning candidate's name, its associated party, and the number of votes they received, displayed on the screen along with the number of votes, seats, % of Votes to % of Seats of the parties, and number of votes and %	The party and candidate winner details displayed on the screen, including the winning party's name, number of votes and seats received and the winning candidate's name, its associated party, and the number of votes they received, along with the number of votes, seats, % of Votes to % of Seats of the parties, and number of votes and % of votes each candidate received is displayed on the screen with appropriate formatting.	

			of votes each candidate received.		
5	Navigate to the /src directory and check for the “audit_file.txt” file		The user should see an audit file with the name “audit_file.txt” created in the “src” directory.	The audit file named “audit_file.txt” is created in the “src” directory	
6	Open the audit file and ensure the Voting Protocol name is the first line of the file.		The voting protocol name on the first line should be “Open Party List (OPL)”	The voting protocol name mentioned on the first line of the file is “Open Party List (OPL)”	
7	Observe lines 5-8 of the audit file before the first round of seat allocations.		The user should see the initial results for each party before the first round of seat allocations in a tabular format where the parties’ number of votes and the number of seats they have allocated are listed.	Once the audit file is opened, the initial results before the first round of seat allocations information is shown in a tabular format, in which you can see the parties’ number of votes and the number of seats they have allocated are listed.	
8	Ensure that the seat allocations have been done correctly at lines 5-8 of the audit file.		D should have 10 votes, and R and I should have 0 votes. D, R, and I should all have 0 seats allocated.	On manual inspection, it was confirmed that D has 10 votes, and R and I have 0 votes. D, R, and I have 0 seats allocated.	
9	Observe lines 12-15 of the audit file for the first round of seat allocations.		The user should see the results after the first round of seat allocations in a tabular format where the parties’ remaining votes and the number of seats they have allocated are listed.	The first round of seat allocations information is shown in a tabular format, in which you can see the parties’ remaining votes and the number of seats they have allocated are listed.	
10	Ensure that the seat allocations have been done correctly at lines 12-15 of the audit file.		D should have 0 remaining votes and 4 seats allocated. R and I should have 0 remaining votes and seats allocated.	D has 0 remaining votes and 4 seats allocated. R and I have 0 remaining votes and seats allocated.	
11	Observe Lines 19-22 that display the names of the tied candidates as well as the winner of the tie.		Lines 19 and 20 should display “Pike” and “Foster” and Line 22 should have one of the two names listed as the	Lines 19 and 20 of the audit file list the names “Pike” and “Foster” as the currently tied candidates, and “Pike” is displayed as	Since the result of a tie breaker is random, “Foster” can also be

			winner.	the winner of the tie breaker on Line 22.	displayed on Line 22 instead of "Pike." Eitherways, the test should succeed.
12	Ensure that the party winner is displayed correctly on line 31.		Line 31 should have D as the party winner with 10 votes and 4 seats.	Line 31 has D as the party winner with 10 votes and 4 seats.	
13	Ensure the candidate winner is displayed on Line 22 and Line 37 is the same person.		Line 22 and 37 should have either Pike's or Foster's name as the declared winner from the D party with 5 votes.	Lines 22 and 37 of the audit file show Pike from the D party as the winner with 5 votes.	Again, "Foster" could be displayed too if the test is run multiple times. Eitherways, the test should succeed.
14	Observe lines 26-37 of the audit file for the results after the seat allocations have been completed.		Lines 26-37 should match what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)"	Lines 26-37 matches with what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)" **Pass**	

Post condition(s) for Test:

The program should exit gracefully once winner information is displayed on the screen and the audit file is generated.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: Unit __ System _X_ **Test Date: 11/11/2023**

Test Case ID#: 47 **Name(s) of Testers:** Vincent Hoang

Test Description: This test ensures that the user is able to enter the correct name of the CSV file and initiate ballot calculations using the **Open Party List (OPL) Voting Protocol** where one of

two parties is chosen as the winner of the tie-breaker.

Indicate where are you storing the tests (what file) and the name of the method/functions being used.
This is a manual system test and no test file has been defined for this test. All the checks will be done manually by the user.

Automated: yes no X

Results: Pass X Fail

Preconditions for Test: The test file associated with this test case (OPL_test_two_tied_parties.csv) should be copied from the "testing" directory to the "src" directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Copy **OPL_test_two_tied_parties.csv** from the "/testing" to "/src" directory manually using copy-paste.		**OPL_test_two_tied_parties.csv** should be present in the "src" directory	**OPL_test_two_tied_parties.csv** is present in the "src" directory.	
2	Open a terminal in the "src" directory and execute the following two commands: **javac VoteEasy.java** **java VoteEasy**	./src/OPL_test_two_tied_parties.csv	Welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	VoteEasy welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	
3	User enters the name "OPL_test_two_tied_parties.csv" when prompted for the file name by the system and presses ENTER on their keyboard.	./src/OPL_test_two_tied_parties.csv	If the correct file name is entered, then no error message is shown and ballot calculations are performed. If the incorrect file name is entered or if the file is missing from the "src" directory, the user is prompted indefinitely until they enter the right name.	The correct file name is entered and ballot calculations are initiated. If the incorrect file name is entered, then the user is prompted to enter the name indefinitely until the right name is entered.	You can run this test multiple times to test what happens when the incorrect file name is entered or the file is missing.

4	User can see the winner details displayed on the screen.	./src/OPL_test_two_tied_parties.csv	Once the right file name is entered, the user should see the party and candidate winner details, including the winning party's name, number of votes and seats received and the winning candidate's name, its associated party, and the number of votes they received, displayed on the screen along with the number of votes, seats, % of Votes to % of Seats of the parties, and number of votes and % of votes each candidate received.	The party and candidate winner details displayed on the screen, including the winning party's name, number of votes and seats received and the winning candidate's name, its associated party, and the number of votes they received, along with the number of votes, seats, % of Votes to % of Seats of the parties, and number of votes and % of votes each candidate received is displayed on the screen with appropriate formatting.	
5	Navigate to the /src directory and check for the "audit_file.txt" file		The user should see an audit file with the name "audit_file.txt" created in the "src" directory.	The audit file named "audit_file.txt" is created in the "src" directory	
6	Open the audit file and ensure the Voting Protocol name is the first line of the file.		The voting protocol name on the first line should be "Open Party List (OPL)"	The voting protocol name mentioned on the first line of the file is "Open Party List (OPL)"	
7	Observe lines 5-8 of the audit file before the first round of seat allocations.		The user should see the initial results for each party before the first round of seat allocations in a tabular format where the parties' number of votes and the number of seats they have allocated are listed.	Once the audit file is opened, the initial results before the first round of seat allocations information is shown in a tabular format, in which you can see the parties' number of votes and the number of seats they have allocated are listed.	
8	Ensure that the seat allocations have been done correctly at lines 5-8 of the audit file.		D and R should have 5 votes, and I should have 0 votes. D, R, and I should all have 0 seats allocated.	On manual inspection, it was confirmed that D and R has 5 votes, and I have 0 votes. D, R, and I have 0 seats allocated.	

9	Observe lines 12-15 of the audit file for the first round of seat allocations.		The user should see the results after the first round of seat allocations in a tabular format where the parties' remaining votes and the number of seats they have allocated are listed.	The first round of seat allocations information is shown in a tabular format, in which you can see the parties' remaining votes and the number of seats they have allocated are listed.	
10	Ensure that the seat allocations have been done correctly at lines 12-15 of the audit file.		D, R, and I should have 0 remaining votes. D and R should have 1 seat allocated, and I should have 0 seats allocated.	D, R, and I have 0 remaining votes. D and R have 1 seat allocated, and I have 0 seats allocated.	
11	Observe Lines 19-22 that display the names of the tied parties as well as the winner of the tie.		Lines 19 and 20 should display "D" and "R" and Line 22 should have one of the two parties listed as the winner.	Lines 19 and 20 of the audit file list the names "D" and "R" as the currently tied parties, and R is displayed as the winner of the tie breaker on Line 22.	Since the result of a tie breaker is random, D can also be displayed on Line 22 instead of R. Eitherways, the test should succeed.
12	Ensure the party winner displayed on Line 22 and Line 31 is the same party.		Line 22 and 31 should either be the D or R party as the declared winner with 5 votes and 1 seat.	Lines 22 and 31 of the audit file show R as the winner with 5 votes and 1 seat.	Again, D could be displayed too if the test is run multiple times. Eitherways, the test should succeed.
13	Ensure that the candidate winner is displayed correctly on the last line.		The last line should be Deutsch as the candidate winner from the R party with 5 votes or Pike as the candidate winner from the D party with 5 votes.	The last line is Deutsch as the candidate winner from the R party with 5 votes.	The candidate winner that is determined depends on the party that has been declared as the winner

					by the tie breaker. If the tie breaker chose D, then Pike would be the candidate winner. Eitherways, the test should succeed.
14	Observe lines 26 to the very last line of the audit file for the results after the seat allocations have been completed.		Lines 26 to the very last line should match what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)"	Lines 26 to the very last line matches with what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)" **Pass**	

Post condition(s) for Test:

The program should exit gracefully once winner information is displayed on the screen and the audit file is generated.

Project Name: Project1 - VoteEasy **Team# 08**

Test Stage: **Unit __** **System _X_**

Test Date: **11/11/2023**

Test Case ID#: **48**

Test Description: This test ensures that the user is able to enter the correct name of the CSV file and initiate ballot calculations using the **Open Party List (OPL) Voting Protocol** where one of two parties is chosen to receive a remaining seat as the winner of the tie-breaker.

Name(s) of Testers: Vincent Hoang

Automated: **yes** **no X**

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

This is a manual system test and no test file has been defined for this test. All the checks will be done manually by the user.

Results: Pass X Fail

Preconditions for Test: The test file associated with this test case (OPL_test_two_remainder_ties.csv) should be copied from the "testing" directory to the "src" directory.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Copy **OPL_test_two_remainder_ties.csv** from the "/testing" to "/src" directory manually using copy-paste.		**OPL_test_two_remainder_ties.csv** should be present in the "src" directory	**OPL_test_two_remainder_ties.csv** is present in the "src" directory.	
2	Open a terminal in the "src" directory and execute the following two commands: **javac VoteEasy.java** **java VoteEasy**	./src/OPL_test_two_remainder_ties.csv	Welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	VoteEasy welcome messages are displayed on screen and user is prompted to enter the name of the CSV file.	
3	User enters the name "OPL_test_two_remainder_ties.csv" when prompted for the file name by the system and presses ENTER on their keyboard.	./src/OPL_test_two_remainder_ties.csv	If the correct file name is entered, then no error message is shown and ballot calculations are performed. If the incorrect file name is entered or if the file is missing from the "src" directory, the user is prompted indefinitely until they enter the right name.	The correct file name is entered and ballot calculations are initiated. If the incorrect file name is entered, then the user is prompted to enter the name indefinitely until the right name is entered.	You can run this test multiple times to test what happens when the incorrect file name is entered or the file is missing.
4	User can see the winner details displayed on the screen.	./src/OPL_test_two_remainder_ties.csv	Once the right file name is entered, the user should see the party and candidate winner details, including the winning party's name, number of votes and seats received	The party and candidate winner details displayed on the screen, including the winning party's name, number of votes and seats received and the winning candidate's name, its associated party, and the	

			and the winning candidate's name, its associated party, and the number of votes they received, displayed on the screen along with the number of votes, seats, % of Votes to % of Seats of the parties, and number of votes and % of votes each candidate received.	number of votes they received, along with the number of votes, seats, % of Votes to % of Seats of the parties, and number of votes and % of votes each candidate received is displayed on the screen with appropriate formatting.	
5	Navigate to the /src directory and check for the "audit_file.txt" file		The user should see an audit file with the name "audit_file.txt" created in the "src" directory.	The audit file named "audit_file.txt" is created in the "src" directory	
6	Open the audit file and ensure the Voting Protocol name is the first line of the file.		The voting protocol name on the first line should be "Open Party List (OPL)"	The voting protocol name mentioned on the first line of the file is "Open Party List (OPL)"	
7	Observe lines 5-8 of the audit file before the first round of seat allocations.		The user should see the initial results for each party before the first round of seat allocations in a tabular format where the parties' number of votes and the number of seats they have allocated are listed.	Once the audit file is opened, the initial results before the first round of seat allocations information is shown in a tabular format, in which you can see the parties' number of votes and the number of seats they have allocated are listed.	
8	Ensure that the seat allocations have been done correctly at lines 5-8 of the audit file.		D and R should have 5 votes, and I should have 0 votes. D, R, and I should all have 0 seats allocated.	On manual inspection, it was confirmed that D and R has 5 votes, and I have 0 votes. D, R, and I have 0 seats allocated.	
9	Observe lines 12-15 of the audit file for the first round of seat allocations.		The user should see the results after the first round of seat allocations in a tabular format where the parties' remaining votes and the number of seats they have allocated are listed.	The first round of seat allocations information is shown in a tabular format, in which you can see the parties' remaining votes and the number of seats they have allocated are listed.	

10	Ensure that the seat allocations have been done correctly at lines 12-15 of the audit file.		D and R should have 1 remaining vote and I should have 0 remaining vote. D and R should have 1 seat allocated, and the I party should have 0 seats allocated.	D and R have 1 remaining vote and I have 0 remaining vote. D and R have 1 seat allocated, and I have 0 seats allocated.	
11	Observe Lines 19-22 that display the names of the tied parties with the same most remaining seats as well as the winner of the tie.		Lines 19 and 20 should display “D” and “R” and Line 22 should have one of the two parties listed as the winner.	Lines 19 and 20 of the audit file list the names “D” and “R” as the currently tied parties, and R is displayed as the winner of the tie breaker on Line 22.	Since the result of a tie breaker is random, D can also be displayed on Line 22 instead of R. Eitherways, the test should succeed.
12	Observe lines 26-29 of the audit file for the second round of seat allocations.		The user should see the results after the second round of seat allocations in a tabular format where the parties’ remaining votes and the number of seats they have allocated are listed.	The second round of seat allocations information is shown in a tabular format, in which you can see the parties’ remaining votes and the number of seats they have allocated are listed.	
13	Ensure that the seat allocations have been done correctly at lines 26-29 of the audit file.		D and R should have 1 remaining vote and I should have 0 remaining vote. D should have 2 seats allocated and R should have 1 seat allocated, or R should have 2 seats allocated and D should have 1 seat allocated, and I should have 0 seats allocated.	D and R have 1 remaining vote and I have 0 remaining vote. R has 2 seats allocated and D has 1 seat allocated, and I has 0 seats allocated.	Again, D could be displayed too if the test is run multiple times. Eitherways, the test should succeed.
14	Ensure that the party that won the tie on line 22 got the remaining seats looking at lines 26-29.		D or R should have their number of seats allocated incremented by 1.	R has their number of seats allocated incremented by 1.	Again, D could be displayed too if the test is run multiple times. Eitherways,

					the test should succeed.
15	Ensure that the correct party winner is displayed on line 38.		D or R should be the winning party with 5 votes and 2 seats	R is the winning party with 5 votes and 2 seats.	Again, D could be displayed too if the test is run multiple times. Eitherways, the test should succeed.
16	Ensure that the candidate winner is displayed correctly on the last line.		The last line should be Deutsch as the candidate winner from the R party with 5 votes or Pike as the candidate winner from the D party with 5 votes.	The last line is Deutsch as the candidate winner from the R party with 5 votes.	The candidate winner that is determined depends on the party that has been declared as the winner by the tie breaker. If the tie breaker chose D, then Pike would be the candidate winner. Eitherways, the test should succeed.
17	Observe lines 33 to the very last line of the audit file for the results after the seat allocations have been completed.		Lines 33 to the very last line should match what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)"	Lines 33 to the very last line matches with what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)" **Pass**	

Post condition(s) for Test:

The program should exit gracefully once winner information is displayed on the screen and the audit file is generated.
