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 S	Stage: Unit _X_	System		Test Date	:		
				11/8/23			
	Case ID#: 1 Description:			Name(s)	of Testers: Steven Petze	old	
Testin class	Testing the getter method for the name attribute of the Candidate class						
Class					where are you storing t he method/functions b	the tests (what file) and the eing used.	
Autor	nated: yes X no			Project1/s	rc/CandidateTests.java		
		ail					
Preco	Preconditions for Test: A candidate has been created with a valid candidate name and party name.						
Step	Test Step	Test	Expected		Actual		
# 1	Description	Data	Result		Result	Notes	

	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 b/hamcrest-core-1.3.jar:. RunAllUnitTestCases	String candidateName = "Rosen";	getName() should return "Rosen" and the test should return 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	Candidate	String partyName = "Independent";			
3		,			
4					

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

Project Name: Project1 - VoteEasy	Team# 8
Test Stage: Unit _X_ System	Test Date:
	11/8/23
Test Case ID#: 2 Test Description:	Name(s) of Testers: Steven Petzold
Testing the getter method for the party attribute of the Candidate class	
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	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		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
	.//lib/junit-4.13.2.jar:.//li	String candidateName = "Rosen"; String partyName = "Independent";			
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	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:.		should update the number of		Once you execute the junit testing commands defined in the test step description, all unit tests defined in the CandidateTests class are run
	RunAllUnitTestCases.java java -cp //lib/junit-4.13.2.jar:.//li	String candidateName = "Kleinberg";			
1 2	RunAllUnitTestCases	String partyName = "Party A";			
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 Test Description:	Name(s) of Testers: Steven Petzold
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.
Automated: yes X no	Project1/src/CandidateTests.java
Results: Pass X Fail	
Preconditions for Test: A candidate has been created with a valid	d candidate name and party name.

Step	Test Step	Test	Expected	Actual	
#	Description	Data	Result	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:.//li	String candidateName = "Kleinberg"; String partyName =	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	Pass	Once you execute the junit testing commands defined in the test step description, all unit tests defined in the CandidateTests class are run

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

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 Test Description:	Name(s) of Testers: Steven Petzold
Testing the resetRedistributedVotes method for the Candidate class. Also checks if the getRedistributedVotes returns the correct value.	t
varae.	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 vali	d candidate name and party name.

Step	Test Step	Test	Expected	Actual	
#	Description	Data	Result	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:.//li b/hamcrest-core-1.3.jar:. RunAllUnitTestCases Candidate		resetRedistributedVotes() method should reset the value of the redistributed votes to 0. The getRedistributedVotes() should verify this value.	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	Candidate	Taity A,			
4					

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	
	•		_	
Precondi	tions for	r Test:	A candidate h	as been created with a valid candidate name and party name.

Step	Test Step	Test	Expected	Actual	
#	Description	Data	1 -	Result	Notes
1	1				
	Open a terminal in the "src" directory and execute the following two commands:		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
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java				lun
2	java -cp .//lib/junit-4.13.2.jar:.//li b/hamcrest-core-1.3.jar:. RunAllUnitTestCases Candidate	String candidateName = "Kleinberg"; String partyName = "Party A";			
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	see sich annac		returns raise.		
4					

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 Test Description:	Name(s) of Testers: Steven Petzold
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.
Automated: yes X no	Project1/src/PartyTests.java
Results: Pass X Fail	
Preconditions for Test: A Party has been created with a valid par	ty name.

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:		getPartyName() should return "PartyTest" and the test should return 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	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java java -cp	String partyName = "PartyTest";			

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

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 Test Description:	Name(s) of Testers: Steven Petzold
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.
Automated: yes X no	Project1/src/PartyTests.java
Test Stage: Unit _X_ System Test Date: 11/8/23 Test Case ID#: 8 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	
	ty name. Candidates have been created with valid candidate names

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

1					
2	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java java -cp .//lib/junit-4.13.2.jar:.//li b/hamcrest-core-1.3.jar:. RunAllUnitTestCases	Party testParty = new Party(partyName); Candidate testCandidate = new Candidate("TestCandidate", partyName); Candidate testCandidate2 = 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	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					

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 Test Description:	Name(s) of Testers: Steven Petzold
Testing the incrementPartyVote() method and getTotalPartyVotes() methods for the party class	
The state of the s	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	

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

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:		The test for getTotalPartyVotes() returns true when the value returned is 2. This is because incrementPartyVote() is called twice.	Pass	Once you execute the junit testing commands defined in the test step description, all unit tests defined in the PartyTests class are run
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java				
	java -cp .//lib/junit-4.13.2.jar:.//li b/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party	String partyName = "PartyTest";			
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

	T	•	. •
LOCT	DOCC	rini	tion.
1031	Desc	TIV	uvn.
		1	

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
1					
	Open a terminal in the "src" directory and execute the following two commands:		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.	Pass	Once you execute the junit testing commands defined in the test step description, all unit tests defined in the PartyTests class are run
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java				
	java -cp .//lib/junit-4.13.2.jar:.//li b/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party	String partyName = "PartyTest";			
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.

ect Name: Project1 - VoteEasy

Test Stage:	Unit	_X_	System	Test Date:
-------------	------	-----	--------	------------

11/8/23

Test Case ID#: 11

Test Description:

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

Name(s) of Testers: Steven Petzold

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

Team# 8

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
1					
	Open a terminal in the "src" directory and execute the following two commands:		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	Pass	Once you execute the junit testing commands defined in the test step description, all unit tests defined in the PartyTests class are run
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java		number of seats allocated for the party object		
	java -cp .//lib/junit-4.13.2.jar:.//li b/hamcrest-core-1.3.jar:. RunAllUnitTestCases Party	String partyName = "PartyTest";			
3	_ · ·J				
4					

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 Test Description:	Name(s) of Testers: Steven Petzold
Testing the setInitialPartyVotes() method and getInitialPartyVotes() methods for the party class	
getimetan arty votes() 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 v	valid party name.

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:		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	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	javac -cp .//lib/junit-4.13.2.jar:.	String partyName = "PartyTest";	party object		

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

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 Test Description:	Name(s) of Testers: Carlos Chasi-Mejia

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

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
	Open a terminal in the "src" directory and execute the following two commands:	./testing/test_file_parser_I R_file.csv	getFileHeader() should return the string "IR".	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
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java				
1	java -cp .//lib/junit-4.13.2.jar:.//lib/hamcrest-core-1 .3.jar:. RunAllUnitTestCases FileParser				
	Open a terminal in the "src" directory and execute the following two commands:	./testing/test_file_parser_O PL_file.csv	_	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
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java				
2	java -cp .//lib/junit-4.13.2.jar:.//lib/hamcrest-core-1 .3.jar:. RunAllUnitTestCases FileParser				

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

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.

Name(s) of Testers: Carlos Chasi-Mejia

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	_ <u> </u>	Actual	
#	Description	Data	Result	Result	Notes
	Open a terminal in the "src" directory and execute the following two commands: javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java	./testing/test_file_parser_IR _file.csv	should return theF int 4.	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
1	java -cp .//lib/junit-4.13.2.jar:.//lib/hamcrest-cor e-1.3.jar:. RunAllUnitTestCases FileParser				

			· ·	getNumberOfCandidates()	Once you execute the junit
	and execute the following two	OPL_file.csv	should return the int 6.	returns the int 6.	testing commands defined in
	commands:			Pass	the test step description, all unit tests defined in the
				1 455	FileParserTests class are run
	javac -cp .//lib/junit-4.13.2.jar:.				1 1101 01001 10010 01000 010 1011
	RunAllUnitTestCases.java				
	java -cp				
	///lib/junit-4.13.2.jar:.//lib/hamcrest-cor				
	e-1.3.jar:. RunAllUnitTestCases				
2	FileParser				

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 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.	Name(s) of Testers: Carlos Chasi-Mejia	
	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 cas contains the actual names and party of each candidate in the elect	· · · · · · · · · · · · · · · · · · ·	

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

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
	Open a terminal in the "src" directory and execute the following two commands: javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java	./testing/test_file_parser_IR _file.csv	getNumberOfBallots() should return 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
1	java -cp .//lib/junit-4.13.2.jar:.//lib/hamcrest-cor e-1.3.jar:. RunAllUnitTestCases FileParser				
	Open a terminal in the "src" directory and execute the following two commands:	./testing/test_file_parser_ OPL_file.csv	getNumberOfBallots() should return 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
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java				i nor arson rests etass are run
	java -cp .//lib/junit-4.13.2.jar:.//lib/hamcrest-cor e-1.3.jar:. RunAllUnitTestCases FileParser				

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: Project	t1 - VoteEasy
-----------------------	---------------

Team#8

Test Stage:	Unit	X	System	
- 000 ~ 000 5 00			~,, > • • • • • • • • • • • • • • • • • •	

11/8/23

Test Date:

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	Test	Expected	Actual	Notes
#	Description	Data	Result	Result	
	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-cor e-1.3.jar:. RunAllUnitTestCases FileParser	./testing/test_file_parser_IR _file.csv	getNumberOfSeats() should return the int 0.	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

			getNumberOfSeats() should	~	Once you execute the junit
	and execute the following two	OPL_file.csv	return the int 3.		testing commands defined in the test step description, all
	commands:				unit tests defined in the
	iavaa on / /lih/iunit 4 13 2 iam				FileParserTests class are run
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java				
	java -cp				
	.//lib/junit-4.13.2.jar:.//lib/hamcrest-cor				
	e-1.3.jar:. RunAllUnitTestCases				
2	FileParser				

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 fil is parsed and stored correctly into an array list so that we can properly retrieve the BallotList attribute of the FileParser class.	e
	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 should be on line 6 if line 1 is "OPL"; or line 5 if line 1 is "IR". The test file associated with this test case should be on line 6 if line 1 is "OPL"; or line 5 if line 1 is "IR".	

Actual

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

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
	-				
	Open a terminal in the "src" directory and execute the		All created	All created	Once you execute the junit
	following two commands:		ballots should	ballots have	testing commands defined in
			have candidates	candidates	the test step description, all
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java		ordered	ordered	unit tests defined in the
	javae ep """no/jame "ro.z.jar. Rum memeresteases.java		correctly.	correctly.	IRVotingTests class are run.
	java -cp .//lib/junit-4.13.2.jar:.//lib/hamcrest-core-1.3.jar:.			Pass	
	RunAllUnitTestCases IRVoting	./testing/IR_test_ballot_dist			
1		ribution_order.txt			

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

Name(s) of Testers: Jashwin Acharya

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

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

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

Name(s) of Testers: Jashwin Acharya

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.

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

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

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 ca the "testing" directory for ease of use. We do not take any user inp	se (IR_test_only_one_candidate_majority.csv) is already present in out for this unit test.

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

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 Name(s) of Testers: Jashwin Acharya

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.

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

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

		· ·	•	. 1	. •	1 1	
tΛ	ancura	fairness	111	tha	11A	hrankar	
w	CHSUIC	Talliloss	111	uic	LIC	DICARCI.	

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	<u>Fail</u>	

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

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

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
	_				On an array array to the invit
	Open a terminal in the "src" directory and execute the following two commands:		The winning candidate should be "Rosen" after	candidate chosen is "Rosen" after	Once you execute the junit testing commands defined in the test step description, all
	javac -cp .//lib/junit-4.13.2.jar:. RunAllUnitTestCases.java		redistribution is complete and the number of votes	complete and the number of votes	unit tests defined in the IRVotingTests class are run.
1	java -cp .//lib/junit-4.13.2.jar:.//lib/hamcrest-core-1.3.jar: . RunAllUnitTestCases IRVoting	/testing/IR_test_majority_ winner_after_one_round_r edistribution.csv	Rosen received should be 3.		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 Name(s) of Testers: Jashwin Acharya

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.

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

Post	conditio	n(s) f	or 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 Name(s) of Testers: Jashwin Acharya

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.

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	Test	Expected	Actual	Notes
#	Description	Data	Result	Result	

	Open a terminal in the "src" directory and		The winning	The winning	Once you execute the junit
	execute the following two commands:		candidate should be	candidate chosen is	testing commands defined in
			"Chou" after	"Chou" after	the test step description, all
	javac -cp .//lib/junit-4.13.2.jar:.		redistribution is	redistribution is	unit tests defined in the
	RunAllUnitTestCases.java		complete and the	complete and the	IRVotingTests class are run.
	Train in a nic rest cuses. Juvu		number of votes	number of votes	
	java -cp		Chou received	Chou received is 6.	No audit file is generated for
	.//lib/junit-4.13.2.jar:.//lib/hamcrest-core-1.3.jar:	/testing/IR test two leadi	should be 6.		this test.
		ng candidate loss after r		Pass	
1	l e	edistribution.csv			

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 Name(s) of Testers: Jashwin Acharya

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.

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

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

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
	Copy			IR_test_first_round_majo	
	IR_test_first_round_majority_winner.csv			rity_winner.csv is present in the "src" directory.	
1	from the "/testing" to "/src" directory manually using copy-paste.		directory	in the sic directory.	
	Open a terminal in the "src" directory and		Welcome messages are	VoteEasy welcome	
	execute the following two commands:			messages are displayed on	
			1 1	screen and user is prompted	
	javac VoteEasy.java			to enter the name of the	
				CSV file.	
2	java VoteEasy	./src/IR_test_first_round_majority_			
		winner.csv	TC.1	m	X.7
					You can run
			,		this test
	I Igar antary the name			calculations are initiated. If	
	User enters the name				to test what
	"IR_test_first_round_majority_winner.csv"		performed. If the incorrect		happens when
	when prompted for the file name by the			prompted to enter the name	
	system and presses ENTER on their	./src/IR_test_first_round_majority_	the file is missing from the	, ,	file name is
3	keyboard.	winner.csv	"src" directory, the user is	name is entered.	entered or the

			prompted indefinitely until		file is missing.
			they enter the right name.		me is missing.
			Once the right file name	The winner details along	
			is entered, the user	with the number / % of	
			should see the winner	votes each candidate	
			details displayed on the	received is displayed on the	
				screen with appropriate	
	Tigan aan gaa tha winnan dataila dianlawad		number / % of votes	formatting.	
4	User can see the winner details displayed on the screen.	",, -	each candidate received.		
	on the screen.	winner.csv		The audit file named	
			The user should see an		
			audit file with the name	"audit_file.txt" is created in the src directory	
_	Navigate to the /src directory and check		addit_inc.txt created	the sic directory	
5	for the "audit_file.txt" file		in the src directory.		
			The voting protocol	The voting protocol name	
			ilatile off the first fille	mentioned on the first line	
	Open the audit file and ensure the Voting		ishidulu de mistant	of the file is "Instant Runoff	
6	Protocol name is the first line of the file.		Runoff (IR)"	(IR)"	
			The user should see the	Once the audit file is	
				opened, the first round of	
			nound of voinig	ballot calculation	
			calculation where a	information is shown in a	
			majority is achieved for	tabular format, below	
			Degan and should go	which you can see the final	
			4 6 1 14 64	voting results also in a	
			1 1	tabular format, followed by	
				Rosen's party affiliation details and the number of	
	Observe lines 3-9 of the audit file for first		which Rosen is declared		
7			as the winner.	votes they received.	
- ' -	round of voting calculation information.			On manual inspection, it	
				Was confirmed that Rosen	
			received reduit votes	did receive 4 votes and	
			wille Kielloeig silouid	Kleinherg had 0 votes while	
	Ensure that the vote calculations have		receive 0 votes and	Chou and Royce were tied	
	been done correctly at lines 3-9 of the		receive 0 votes and Chou and Royce are tied	at 1 vote each.	
8	audit file.		at I vote each.		
			The last line of the audit		
			THE BROWN ROBER	file does show Rosen as the	
			as the willier with 0	winner with 6 votes to their	
	Observe the last line of the audit file to		votes to their name	name.	
9				D.c	
	verify the winner information.			Pass	

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

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 one of two candidates 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 (IR_test_two_candidate_tie.csv) should be copied from the "testing" directory to the "src" directory.

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

			CSV file.	
	java VoteEasy		CS V IIIC.	
	,			
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.	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	the incorrect file name is entered, then the user is prompted to enter the name	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.	should see the winner details displayed on the screen along with the number / % of votes	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	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 results after the first round of ballot calculations where Rosen and Kleinberg are tied.		
8	Ensure that the vote calculations have been done correctly at lines 3-9 of the audit file.	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.	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 t winner.	on Line 16.	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 shoul have either Rosen or Kleinberg's name as declared winner with votes.	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.

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 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. 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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
	•				
			IR_test_two_leading_can	IR_test_two_leading_can	
	Сору			didate_loss_after_redistri	
	IR_test_two_leading_candidate_loss_after_			bution.csv is present in the	
1	redistribution.csv from the "/testing" to		present in the "src"	"src" directory.	
1	"/src" directory manually using copy-paste.		directory		
	Open a terminal in the "src" directory and		Welcome messages are	VoteEasy welcome	
	execute the following two commands:			messages are displayed on	
				screen and user is prompted to enter the name of the	
	javac VoteEasy.java		the name of the CSV me.	CSV file.	
	iana VotaFasa	/ /ID / / / I I' I' I' I		CBV IIIC.	
2	java VoteEasy	/src/IR_test_two_leading_candidat			
		e loss after redistribution.csv	If the correct file name is	The correct file name is	You can run
				entered and ballot	this test
				calculations are initiated. If	
					to test what
				entered, then the user is	happens when
	User enters the name			prompted to enter the name	
	"IR_test_two_leading_candidate_loss_after_r		the file is missing from the	indefinitely until the right	file name is
	edistribution.csv" when prompted for the file			name is entered.	entered or the
1 2	name by the system and presses ENTER on	$./src/IR_test_two_leading_candidat$			file is missing.
3	their keyboard.	e loss after redistribution.csv	they enter the right name.		
			Once the right file name	The winner details along	
			is critered, the aser	with the number / % of	
			should see the willier	votes each candidate received is displayed on the	
			uctails displayed on the	screen with appropriate	
			Screen along with the	formatting.	
1 ,	User can see the winner details displayed	./src/IR_test_two_leading_candidat	number / % of votes		
4	on the screen.	e loss after redistribution.csv	each candidate received.		
			The user should see an	The audit file named	
			audit file with the name	"audit_file.txt" is created in	
_ ا	Navigate to the /src directory and check		addit_inc.txt created	the "src" directory	
5	for the "audit_file.txt" file		in the "src" directory.		
1 .	Open the audit file and ensure the Voting		The voting protocol	The voting protocol name	
6	Protocol name is the first line of the file.		name on the first line	mentioned on the first line	

			d
		should be "Instant of the file is "Instant Runoff	Ī
		Runoff (IR)" (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.	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	

Team# 08

Test Stage: Unit System X Test Date: 11/10/2023

Test Case ID#: 33 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 redistribution is performed multiple times and two candidates end up in a tie-breaker with only one of them winning the election.

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
			. – – – – –	IR_test_candidate_tie_aft	
	Copy			er_multiple_rounds_of_re	
	IR_test_candidate_tie_after_multiple_roun		edistribution.csv should	distribution.csv is present	
	ds_of_redistribution.csv from the "/testing"		be present in the "src"	in the "src" directory.	
1	to "/src" directory manually using copy-paste.		directory	-	
	Open a terminal in the "src" directory and		Welcome messages are	VoteEasy welcome	
	execute the following two commands:		displayed on screen and	messages are displayed on	
		./src/IR_test_candidate_tie_after_m	user is prompted to enter	screen and user is prompted	
	javac VoteEasy.java	ultiple_rounds_of_redistribution.cs	the name of the CSV file.	to enter the name of the	
2	,	V		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_m ultiple_rounds_of_redistribution.cs	performed. If the incorrect file name is entered or if the file is missing from the "src" directory, the user is	prompted to enter the name	to test what happens when
4	User can see the winner details displayed on the screen.	_/src/IR_test_candidate_tie_after_m ultiple_rounds_of_redistribution.cs	Once the right file name is entered, the user should see the winner details displayed on the	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.		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.	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.	"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 (50-50 split).	
11	Observe tied candidate names at lines 31 and 32 of the audit file.	Lines 31 and 32 mention Lines 31 and 32 do mention explicitly that Rosen and 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.

OPLVoting Tests:

Unit Tests:

Project Name: Project1 - VoteEasy Team# 08

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

Test Case ID#: 34 Name(s) of Testers: Vincent Hoang

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.

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

:. RunAllUnitTestCases OPLVoting		

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

Test Stage: Unit X System Test Date: 11/10/2023

Test Case ID#: 35 Name(s) of Testers: Vincent Hoang

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.

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

Team# 08

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

RunAl	llUnitTestCases.java	votes, and the	and the winning	OPLVotingTests class are run.
		winning candidate	candidate is "Smith"	
java -c	ep	should be "Smith"	receiving "9" votes.	No audit file is generated for
.//lib/j	junit-4.13.2.jar:.//lib/hamcrest-core-1.3.jar	receiving "9" votes.	Pass	this test.
:. Run	AllUnitTestCases OPLVoting			

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.

Team# 0		
est Date: 11/10/2023		
ame(s) of Testers: Vincent Hoang		
andicate where are you storing the tests (what file) and the ame 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 estAllVotesToOneParty()".		
a h		

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	Test	Expected	Actual	Notes
#	Description	Data	Result	Result	

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

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		
single candidate, with no ties occurring in this situation.	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			
11001100 1 1100			

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

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 **Test Date: 11/10/2023** System Test Case ID#: 38 Name(s) of Testers: Vincent Hoang **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. 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 Fail **Results: Pass**

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

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

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()".

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

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 Name(s) of Testers: Vincent Hoang

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.

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_re mainder ties.csv	each have a 45-55% chance of being	are each chosen as the tie party that	Once you execute the junit testing commands defined in the test step description, all unit tests defined in the

RunAllUnitTestCases.java	that	receives a sea	eat 45-55% of the	OPLVotingTests class are run.
	remain	ing seat from time	ne after simulating	
java -cp	a tie	e breaker. the	e tie breaker 1000 🏻	No audit file is generated for
.//lib/junit-4.13.2.jar:.//lib/hamcrest-core-1.3.jar			times. t	his test.
:. RunAllUnitTestCases OPLVoting			Pass	

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 Name(s) of Testers: Vincent Hoang

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.

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

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 Name(s) of Testers: Vincent Hoang

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.

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

۲4 م م ۲ ۲		r: ar		~(\)	"
"test]	LWO.	near	arue	S()	١.

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
	Open a terminal in the "src" directory and		The party D and R	The party D and R	Once you execute the junit
	execute the following two commands:				testing commands defined in
			chance of being	1 2	the test step description, all
	javac -cp .//lib/junit-4.13.2.jar:.		chosen as the party		unit tests defined in the
	RunAllUnitTestCases.java		winner from a tie		OPLVotingTests class are run.
			breaker.	after simulating the	
	java -cp			tie breaker 1000	No audit file is generated for
	.//lib/junit-4.13.2.jar:.//lib/hamcrest-core-1.3.jar	./testing/OPL test two ti		times.	this test.
1	"	ed parties.csv		Pass	

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 Name(s) of Testers: Vincent Hoang

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

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

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 - VoteEas	Project Name	: Project1 -	VoteEasy
----------------------------------	--------------	--------------	-----------------

Team# 08

	System X	Unit	Test Stage:
--	----------	------	--------------------

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

Test Date: 11/11/2023

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	K_ 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.		of_votes.csv should be	OPL_test_large_number_ of_votes.csv is present in the "src" directory.	
	Open a terminal in the "src" directory and execute the following two commands: javac VoteEasy.java java VoteEasy		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.	

1				I	
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.		message is shown and seat allocations are performed. If the incorrect file name is entered or if the file is missing from the "src"	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.
	User can see the winner details displayed	./src/OPL_test_large_number_of_	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	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	on the screen. Navigate to the /src directory and check for the "audit file.txt" file	votes.csv	each candidate received. 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.		initial results for each party before the first round of seat allocations in a tabular format where the parties' 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 all	
		are listed.	are listed.
			otes, R On manual inspection, it
		should have 5 vote	
	Ensure that the seat allocations have been	I should have 9 vo	tes. D, votes, R has 5 votes, and I
	done correctly at lines 5-8 of the audit	R, and I should all	have 0 has 9 votes. D, R, and I have 0 seats allocated.
8	file.	seats allocated.	nave o seats anocated.
		The user should se	
		results after the fire	
		round of seat alloc	lin 1. i - 1 41
		in a tabular format	where in which you can see the
		the parties' remain	
		votes and the num	ber of they have allocated are
	Observe lines 12-15 of the audit file for	seats they have all	ocated listed.
9	the first round of seat allocations.	are listed.	
		D should have 1	On manual inspection, it
		remaining vote, R	should was confirmed that D has 1
		have 0 remaining v	votes. remaining vote, R has 0
		and I should have	remaining votes, and I has 4
	Ensure that the seat allocations have been	remaining votes. D	P. R. remaining votes. D, R, and
	done correctly at lines 12-15 of the audit	and I should all ha	
10	file.	seat allocated.	
		The user should se	e the Looking below the previous
		results after the sec	cond round, the second round of
		round of seat alloc	ations seat allocations information
		in a tabular format	where is shown in a tabular
		the parties' remain	format, in which you can
		votes and the number	
	Observe lines 19-22 of the audit file for	seats they have all	seats they have allocated
11	the second round of seat allocations.	are listed.	are listed.
	and become round of bout uncountries.	D should have 1	On manual inspection, it
		remaining vote R	should was confirmed that D has 1
		have 0 remaining v	
		and I should have	remaining votes, and I has 4
		remaining votes T	and R remaining votes. D and R have 1 seat allocated and I
	Ensure that the seat allocations have been	should have 1 seat	
	done correctly at lines 19-22 of the audit	allocated and I sho	inave / seats attocated - 1
12	file.	have 2 seats alloca	
		Line 31 should dis	
		the I party as the p	
		winner with 9 vote	
		2 seats allocated.	
12	Ensure that line 31 displays the correct	2 seas anotated.	
13	party winner.		

14	Ensure that line 36 displays the correct candidate winner.	Smith as	s the candidate from I party with	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.	match w displaye after the protocol	what was ed onto the screen	Lines 26-36 matches with what was displayed onto the screen after the line "Voting protocol chosen is Open Party List (OPL)" Pass	

ı	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	Name(s) of Testers: Vincent Hoang
	two candidates 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	
- 1	Preconditions for Test: The test file associated with this test case "testing" directory to the "src" directory.	e (OPL_test_two_tied_candidates.csv) should be copied from the

Test Expected Actual Result Result Result	llt Notes
Copy OPL_test_two_tied_candidates.csv from the "/testing" to "/src" directory manually using copy-paste. Copy OPL_test_two_tied_candidates.csv is present in the "src" directory.	
Copy OPL_test_two_tied_candidates.csv from the "/testing" to "/src" directory manually using copy-paste. Copy OPL_test_two_tied_candidates.csv is present in the "src" directory.	
from the "/testing" to "/src" directory manually using copy-paste. from the "/testing" to "/src" directory manually using copy-paste. present in the "src" directory. directory	
1 manually using copy-paste. directory	sent in the
mandany danig copy paste.	
I han a tarminal in the "are" directory and I IWALAMA IWALAMA MACAMA ARE INCIDENCE WALLAM	
l l	
execute the following two commands: displayed on screen and user is prompted to enter screen and user is prompted to ente	
javac VoteEasy.java the name of the CSV file. to enter the name CSV file. CSV file.	
java VoteEasy ./src/OPL_test_two_tied_candidate	
2 s.csv	
If the correct file name is The correct file name is	I
entered, then no error entered and ballo	
message is shown and calculations are ballot calculations are the incorrect file	initiated. If multiple times to test what
performed. If the incorrect lentered, then the	
	er the name the incorrect
User enters the name the file is missing from the indefinitely until	
"OPL_test_two_tied_candidates.csv" when "src" directory, the user is name is entered.	
prompted for the file name by the system and ./src/OPL_test_two_tied_candidate prompted indefinitely until	file is missing.
presses ENTER on their keyboard. s.csv they enter the right name.	
Once the right file name The party and ca	andidate
is entered, the user winner details displayed see the party and the screen, inclu	
Should see the party that	
calididate willief details,	
Including the winning received and the	
party's name, number of candidate's nam	
votes and seats received associated party,	
and the winning number of votes candidate's name, its received, along the second secon	
candidate's name, its received, along vassociated party, and the number of votes	
number of votes they of Votes to % of	
received, displayed on the parties, and i	
the screen along with the votes and % of v	votes each
l l l l l l l l l l l l l l l l l l l	700 1C
% of Votes to % of Seats displayed on the	e screen
User can see the winner details displayed _/src/OPL test two tied candidate of the parties, and with appropriate	e formatting.
4 on the screen. Size of Each Section Si	

		C 4 1 1:14	
		of votes each candidate	
		received.	
		The user should see an The audit file named	
		audit file with the name "audit_file.txt" is created in	
	Navigate to the /src directory and check	"audit_file.txt" created the "src" directory	
5	for the "audit file.txt" file	in the "src" directory.	
		The voting protocol The voting protocol name	
		name on the first line mentioned on the first line	
	Open the audit file and ensure the Voting	should be "Open Party of the file is "Open Party	
6	Protocol name is the first line of the file.	List (OPL)" List (OPL)"	
	TOVO CO TAMENO AS VICE THIS THIS OF VICE THE	The user should see the Once the audit file is	
		initial results for each opened, the initial results	
		party before the first before the first round of	
		round of seat allocations information	
		round of seat anocations is shown in a tabular	
		in a tabular format format, in which you can	
		where the parties' see the parties' number of	
		number of votes and the votes and the number of	
_	Observe lines 5-8 of the audit file before	number of seats they seats they have allocated	
7	the first round of seat allocations.	have allocated are listed.	
		D should have 10 votes, On manual inspection, it	
		and R and I should have was confirmed that D has	
	Ensure that the seat allocations have been	0 votes. D, R, and I 10 votes, and R and I have	
	done correctly at lines 5-8 of the audit	should all have 0 seats 0 votes. D, R, and I have 0	
8	file.	allocated. seats allocated.	
		The user should see the The first round of seat	
		results after the first allocations information is	
		round of seat allocations shown in a tabular format,	
		in a tabular format in which you can see the	
		where the parties' parties' remaining votes	
		remaining votes and the land the number of seats	
	Observe lines 12-15 of the audit file for	uncy have anocated are	
9	the first round of seat allocations.	number of seats they listed. have allocated are listed.	
<u> </u>	and first round of scat anocations.	D should have 0 D has 0 remaining votes	
		remaining votes and 4 and 4 seats allocated. R and I have 0 remaining votes	
		seats affocated. K and 1 have a femalining votes	
	Ensure that the seat allocations have been	should have 0 remaining and seats allocated.	
10	done correctly at lines 12-15 of the audit	votes and seats	
10	file.	allocated.	a: 1
		Lines 19 and 20 should Lines 19 and 20 of the audit	
		display "Pike" and file list the names "Pike"	of a tie breaker
	Observe Lines 19-22 that display the	"Foster" and Line 22 and "Foster" as the	is random,
	names of the tied candidates as well as	should have one of the currently tied candidates,	"Foster" can
11	the winner of the tie.	two names listed as the	also be

		winner. the winner of the tie breaker on Line 22.	displayed on Line 22 instead of "Pike." Eitherways, the test should succeed.
	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.
1 4 4	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	

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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
1	Copy OPL_test_two_tied_parties.csv from the "/testing" to "/src" directory manually using copy-paste.		es.csv should be present in	OPL_test_two_tied_partie s.csv is present in the "src" directory.	
	Open a terminal in the "src" directory and execute the following two commands: javac VoteEasy.java		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.	
2	java VoteEasy	./src/OPL_test_two_tied_parties.cs			
			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	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
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.		prompted indefinitely until they enter the right name.		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.cs	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.	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		audit file with the name	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.		should be "Open Party	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.		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.		
	Ensure that the seat allocations have been done correctly at lines 5-8 of the audit file.		votes, and I should have 0 votes. D, R, and I should all have 0 seats	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. D, R, and I should have D, R, and I have 0	
10	Ensure that the seat allocations have been done correctly at lines 12-15 of the audit file.	0 remaining votes. D remaining votes. D and R and R should have 1 seat allocated, and I should have 0 seats allocated.	
	Observe Lines 19-22 that display the	Lines 19 and 20 should display "D" and "R" and file list the names "D" and "R" as the currently tied parties, and R is displayed 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
11	names of the tied parties as well as the winner of the tie.		should succeed.
12	Ensure the party winner displayed on Line 22 and Line 31 is the same party.	1 Scat.	
	Ensure that the candidate winner is	winner from the R party with 5 votes. with 5 votes or Pike as the candidate winner from the D party with 5 votes.	The candidate winner that is determined depends on the party that has been declared as
13	displayed correctly on the last line.		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.	was displayed onto the screen after the line	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)"	

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.	
	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
Automated: yes no X	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	Test	Expected	Actual	
#	Description	Data	Result	Result	Notes
	Copy OPL_test_two_remainder_ties.csv from the "/testing" to "/src" directory manually using copy-paste.		OPL_test_two_remainde r_ties.csv should be present in the "src" directory	OPL_test_two_remainder _ties.csv is present in the "sre" directory.	
	Open a terminal in the "src" directory and execute the following two commands: javac VoteEasy.java		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.	
2	java VoteEasy	./src/OPL_test_two_remainder_ties			
	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.		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.	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	should see the party and candidate winner details, including the winning party's name, number of	winner details displayed on the screen, including 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. and the winning 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.	ŗ.
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" is created in the "src" directory. The audit file named "audit_file.txt" is created in the "src" directory	1
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.	1
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 I has 5 votes, and I have 0 votes. D, R, and I have 0 seats allocated.	2
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 have 1 seat allocated, and the I party should have 0 seats allocated. Lines 19 and 20 should Lines 19 and 20 of the audit	
	Observe Lines 19-22 that display the names of the tied parties with the same	Lines 19 and 20 should display "D" and "R" and file list the names "D" and "R" as the currently tied parties, and R is displayed 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.	result of a tie breaker is random, D can also be displayed on Line 22 instead of R. Eitherways, the test
11	most remaining seats as well as the winner of the tie.		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 have 0 remaining vote. R has 2 seats allocated and D has 1 seat allocated, or R should have 1 seat allocated, or R should have 2 seats allocated and D should have 1 seat allocated, and I seat allocated and D should have 1 seat allocated, and I should have 0 seats allocated.	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.	

		<u> </u>	_
			the test
			should
			succeed.
		D or R should be the R is the winning party with	n Again, D
		winning party with 5 5 votes and 2 seats.	could be
		votes and 2 seats	displayed too
		votes and 2 souts	if the test is
			run multiple
			times.
			Eitherways,
			the test
	Ensure that the correct party winner is		should
15	displayed on line 38.		succeed.
		The last line should be The last line is Deutsch as	The
		Deutsch as the candidate the candidate winner from	candidate
		winner from the R party with 5 votes.	winner that
		with 5 votes or Pike as	is
		the candidate winner	determined
			depends on
		from the D party with 5	the party that
		votes.	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
	Ensure that the candidate winner is		should
16	displayed correctly on the last line.		succeed.
		Lines 33 to the very last Lines 33 to the very last	
		line should match what line matches with what wa	s
		was displayed onto the displayed onto the screen	
		screen after the line "Voting	
		selecti after the fine	
	Observe lines 33 to the very last line of	voting protocor chosen [party I jet (OPI)"	
1	the audit file for the results after the seat	is Open Party List	
17	allocations have been completed.	(OPL)" Pass	