Project 1: Voting System   Team 10

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_1   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getCandidates() function to see if
it returns the correct array of Candidates

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

Automated: yes _X_   no __   STVBallotUT.java getCandidatesTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setCandidates is used to set the value of the candidate array to c	STVBallot.setCandidates(c)	The array candidates is now set to the value of the array c	The array candidates is equal to the array c	c and other variables are initialized at the top of the class
2	Check to see the first index of the candidates array	assertEquals(STVBallot.getCandidates()[0], candidate1);	True	True	
3	Check to see the second index of the candidates array	assertEquals(STVBallot.getCandidates()[1], candidate2);	True	True	

Post condition(s) for Test:

The candidates array is set the value of array c

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_2   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the setCandidates() function to see if
it sets the array of Candidates correctly

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

Automated: yes _X_   no __   STVBallotUT.java setCandidatesTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setCandidates is used to set the value of the candidate array to c	STVBallot.setCandidates(c)	The array candidates is now set to the value of the array c	The array candidates is equal to the array c	c and other variables are initialized at the top of the class

2	Check to see if the candidates array stores c	assertEquals(STVBallot.getCandidates(), c);	True	True	

Post condition(s) for Test:

The candidates array is set the value of array c

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_3   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getBallotID() function to see if it returns the correct ballot ID

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

Automated: yes _X_   no __   STVBallotUT.java getBallotIDTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The location of the input ballot file has been found and is being parsed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The ID of the ballot is being set to 10	b.setBallotID(10);	The ID of the ballot b is now 10	The ID of ballot b is 10	b and other variables are initialized at the top of the class
2	Check to see if the ballot b's ID is 10	assertEquals(b.getBallotID(), 10);	True	True	

Post condition(s) for Test:

The ID of the ballot is now set to 10

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_4   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the setBallotID() function to see if it correctly sets the Ballots ID number

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

Automated: yes _X_   no __   STVBallotUT.java setBallotIDTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The location of the input ballot file has been found and is being parsed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The ID of the ballot is being set to 2	b.setBallotID(2);	The ID of the ballot b is now 2	The ID of ballot b is 2	b and other variables are initialized at the top of the class
2	Check to see if the ballot b's ID is 2	assertEquals(b.	True	True	

		getBallotID(), 2);			
3	The ID of the ballot is being set to 200	b.setBallotID(200);	The ID of the ballot b is now 200	The ID of ballot b is 200	
4	Check to see if the ballot b's ID is 200	assertEquals(b.getBallotID(), 200);	True	True	

Post condition(s) for Test:

The ID of the ballot is now set to 200

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_5   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the toString() function to see if it prints the ballot correctly

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

Automated: yes _X_   no __   STVBallotUT.java toString()

Results: Pass _X_   Fail ___

Preconditions for Test:

The location of the input ballot file has been found and is being parsed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The value of the ballot is checked to see if it equals the other side	assertEquals("[0, 1]", b.toString());	True	True	b and other variables are initialized at the top of the class

Post condition(s) for Test:

The contents of the ballot's array have been printed. Nothing else has been altered.

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_6   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getChoice() function to see if it
returns the correct candidate

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

Automated: yes _X_   no __   STVBallotUT.java getChoiceTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed and the candidate array set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setCandidates is used to set the	STVBallot.setCandidates(c)	The array candidates is now	The array candidates is equal to the array c	c and other variables are

	value of the candidate array to c		set to the value of the array c		initialized at the top of the class
2	Check if getChoice returns candidate 2 for ballot b	assertEquals(b. getChoice(), candidate2);	True	True	b and other variables are initialized at the top of the class

Post condition(s) for Test:

The candidates array has been set to c

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_7   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getChoice() function to see if it returns the correct candidate

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

Automated: yes _X_   no __   STVBallotUT.java getChoiceTest2()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed and the candidate array set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize votes array	int[] myVotes2 = {1, 2};	A new array is successfully created and initialized	The array candidates is equal to the array c	
2	Create a new ballot with votes array	STVBallot b2 = new STVBallot(myVotes2);	A new ballot is successfully created	Ballot b2 has been created with myVotes2 passed in	
3	setCandidates is used to set the value of the	STVBallot.setCandidates(c)	The array candidates is now set to the value of the array c	The array candidates is equal to the array c	c and other variables are initialized at the top of the class

	candidate array to c				
4	Check if getChoice returns candidate 1 for ballot b2	assertEquals(b2.getChoice(), candidate1);	True	True	

Post condition(s) for Test:

A ballot object has been created and the candidates array has been set to c.

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_8   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getChoice() function to see if it returns the correct candidate

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

Automated: yes _X_   no __

STVBallotUT.java getChoiceTest3()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed and the candidate array set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize votes array	int[] myVotes3 = {1, 2};	A new array is successfully created and initialized	Array has been created	
2	Create a new ballot with votes array	STVBallot b3 = new STVBallot(myVotes3);	A new ballot is successfully created	Ballot b3 has been created with myVotes3 passed in	
3	Increment currNum	b3.incrementCurrNum();	currNum is now 2 for Ballot b3	currNum of b3 is 2	
4	setCandidates is used to set the value of the candidate array to c	STVBallot.setCandidates(c)	The array candidates is now set to the value of the array c	The array candidates is equal to the array c	c and other variables are initialized at the top of the class

			True	True	
5	Check if getChoice returns candidate 2 for ballot b3	assertEquals(b 2.getChoice(), candidate2);			

Post condition(s) for Test:

A ballot object has been created and it's currNum has been increased by one. The candidates array also has been set to c.

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: STVBallot_9   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the incrementCurrNum() function to
see if it correctly increments currNum

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

 STVBallotUT.java
incrementCurrNumTest()

Automated: yes _X_   no __

Results: Pass _X_   Fail ___

Preconditions for Test:

The location of the input ballot file has been found and is being parsed; the array of candidates has been set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Increment currNum	b.incrementCurrNum();	currNum is now 2 for Ballot b	currNum of b is 2	b and other variables are initialized at the top of the class
2	setCandidates is used to set the value of the candidate array to c	STVBallot.setCandidates(c)	The array candidates is now set to the value of the array c	The array candidates is equal to the array c	c and other variables are initialized at the top of the class
3	Check if getChoice returns candidate 1 for ballot b	assertEquals(b.getChoice(), candidate1);	True	True	

Post condition(s) for Test:

The ballot's currNum has been increased by one and the value of the candidates array has been set to c

Test Stage: Unit _X_   System __   **Test Date:** 4/2/20

Test Case ID#: STVBallot_10   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getCurrNum() function to see if it
returns the correct currNum

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

Automated: yes _X_   no __   STVBallotUT.java getCurrNumTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The location of the input ballot file has been found and is being parsed; the array of candidates has been set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to see if the ballot's currNum is 1, which it should be	assertEquals(b. getCurrNum(), 1	True	True	b and other variables are initialized at the top of the class

Post condition(s) for Test:

Nothing has been changed

Test Stage: Unit _X_   System __   **Test Date:** 4/2/20

Test Case ID#: STVBallot_11   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the equals() function to see if it correctly compares two STV Ballots (Ballots are compared by their votes)

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

Automated: yes _X_   no __   STVBallotUT.java equalsTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to see if Ballot b is equal to itself	assertEquals(b, b);	True	True	b and other variables are initialized at the top of the class
2	Creates a new votes array	int[] myVotes2 = {1, 2};	A new array is created	A new array is created	
3	Creates a new STV Ballot object with a different votes array than b	STVBallot b2 = new STVBallot(myVotes2);	A new STV Ballot object is created	A new STV Ballot object is created	

4	Check to see if Ballot b is equal to ballot 2	assertNotEquals(b, b2);	True	True	

Post condition(s) for Test:

A new ballot object has been created

Test Stage: Unit _X_   System __   **Test Date:** 4/2/20

Test Case ID#: STVBallot_12   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getVotesSize() function to see if it returns the correct length of the votes array

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

Automated: yes _X_   no __   STVBallotUT.java getVotesSizeTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The location of the input ballot file has been found and is being parsed; the array of candidates has been set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the function returns 2, which is the size of the votes array	assertEquals(b. getVoteSize(), 2);	True	True	b and other variables are initialized at the top of the class

Post condition(s) for Test:

Nothing has been changed

Test Stage: Unit _X_   System __   **Test Date:** 4/2/20

Test Case ID#: STVBallot_13   **Name(s) of Testers:** Jessica Moore

Test Description:

Tests the getVotes() function to see if it
returns the correct votes array

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

Automated: yes _X_   no __   STVBallotUT.java getVotesTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The location of the input ballot file has been found and is being parsed; the array of
candidates has been set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Checks to see if the function returns {2,1}, which is the votes array	assertTrue(Arrays.equals(new int[]{2, 1}, b.getVotes()));	True	True	b and other variables are initialized at the top of the class

Post condition(s) for Test:

Nothing has been changed

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: PluralityBallot_1   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getCandidates() function to see if
it returns the correct array of Candidates

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

Automated: yes _X_   no __   PluralityBallotUT.java getCandidatesTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	setCandidates is used to set the value of the candidate array to c	PluralityBallot. setCandidates(c)	The array candidates is now set to the value of the array c	The array candidates is equal to the array c	c and other variables are initialized at the top of the class
2	Check to see the first index of the candidates array	assertEquals(Pl uralityBallot.get Candidates()[0], candidate1);	True	True	
3	Check to see the second index of the candidates array	assertEquals(Pl uralityBallot.get Candidates()[1], candidate2);	True	True	

Post condition(s) for Test:

The candidates array is set the value of array c

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: PluralityBallot_2   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the setCandidates() function to see if
it sets the array of Candidates correctly

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

Automated: yes _X_   no __   PluralityBallotUT.java setCandidatesTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed

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

1	setCandidates is used to set the value of the candidate array to c	PluralityBallot. setCandidates(c)	The array candidates is now set to the value of the array c	The array candidates is equal to the array c	c and other variables are initialized at the top of the class
2	Check to see if the candidates array stores c	assertEquals(Pl uralityBallot.get Candidates(), c);	True	True	

Post condition(s) for Test:

The candidates array is set the value of array c

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: PluralityBallot_3   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the toString() function to see if it prints the ballot correctly

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

Automated: yes _X_   no __   PluralityBallotUT.java toString()

Results: Pass _X_   Fail ___

Preconditions for Test:

The location of the input ballot file has been found and is being parsed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	The value of the ballot is checked to see if it equals the other side	assertEquals("[0, 1]", b.toString());	True	True	b and other variables are initialized at the top of the class

Post condition(s) for Test:

The contents of the ballot's array have been printed. No other values have been altered.

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: PluralityBallot_4   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getChoice() function to see if it
returns the correct candidate

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

Automated: yes _X_   no __   PluralityBallotUT.java getChoiceTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed and the candidate array set

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

			The array	The array candidates is	c and other
1	setCandidates is used to set the value of the candidate array to c	PluralityBallot. setCandidates(c)	The array candidates is now set to the value of the array c	equal to the array c	variables are initialized at the top of the class
2	Check if getChoice returns candidate 2 for ballot b	assertEquals(b. getChoice(), candidate2);	True	True	b and other variables are initialized at the top of the class

Post condition(s) for Test:

The candidates array is set the value of array c

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: PluralityBallot_5   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getChoice() function to see if it returns the correct candidate

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

Automated: yes _X_   no __   PluralityBallotUT.java getChoiceTest2()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed and the candidate array set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize votes array	int[] myVotes2 = {1, 0};	A new array is successfully created and initialized	Array myVotes2 has been created	
2	Create a new ballot with votes array	PluralityBallot b2 = new PluralityBallot(myVotes2);	A new ballot is successfully created	Ballot b2 is created with myVotes2 passed in	
3	setCandidates is used to set the value of the	PluralityBallot. setCandidates(c)	The array candidates is now	The array candidates is equal to the array c	c and other variables are

			set to the value of the array c		initialized at the top of the class
	candidate array to c				
4	Check if getChoice returns candidate 1 for ballot b2	assertEquals(b2.getChoice(), candidate1);	True	True	

Post condition(s) for Test:

A new ballot has been created and the candidates array is set the value of array c.

Test Stage: Unit _X_   System __   **Test Date:** 3/30/20

Test Case ID#: PluralityBallot_6   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the getChoice() function to see if it returns the correct candidate

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

Automated: yes _X_   no __

PluralityBallotUT.java getChoiceTest3()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed and the candidate array set

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize votes array	int[] myVotes3 = {0, 1};	A new array is successfully created and initialized	Array myVotes3 has been created	
2	Create a new ballot with votes array	PluralityBallot b3 = new PluralityBallot(myVotes3);	A new ballot is successfully created	Ballot b3 has been created with myVotes3 passed in	
3	setCandidates is used to set the value of the candidate array to c	PluralityBallot. setCandidates(c)	The array candidates is now set to the value of the array c	The array candidates is equal to the array c	c and other variables are initialized at the top of the class

			True	True	
4	Check if getChoice returns candidate 2 for ballot b3	assertEquals(b 2.getChoice(), candidate2);			

Post condition(s) for Test:

A new ballot has been created and the candidates array is set the value of array c.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/20

Test Case ID#: PluralityBallot_7   **Name(s) of Testers:** Zac Tressel

Test Description:

Tests the equals() function to see if it correctly compares two Plurality Ballots (Ballots are compared by their votes)

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

Automated: yes _X_   no __   PluralityBallotUT.java equalsTest()

Results: Pass _X_   Fail ___

Preconditions for Test:

The ballots must have been processed

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to see if Ballot b is equal to itself	assertEquals(b, b);	True	True	b and other variables are initialized at the top of the class
2	Creates a new votes array	int[] myVotes2 = {1, 2};	A new array is created	A new array is created	
3	Creates a new Plurality Ballot object with a different votes array than b	PluralityBallot b2 = new PluralityBallot(myVotes2);	A new plurality Ballot object is created	A new plurality Ballot object is created	
4	Check to see if Ballot b is equal to ballot 2	assertNotEqual s(b, b2);	True	True	

Post condition(s) for Test:

A new ballot object has been created

Test Stage: Unit _X_   System __

Test Date: 4/2/2020

Test Case ID#: Report_1

Name(s) of Testers: Jessica Moore

Test Description:

Tests the createLog() function to see if it
records the results of a Plurality Election.

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

Automated: yes__X_   no __

ReportUT.java testPluralityCreateLog()

Results: Pass __X___   Fail_____

Preconditions for Test:

A PluralityElection object has been constructed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the election.	pElection.runElection();	An election is run.	pElection runs successfully.	
2	Create a Report object and run createLog() to create a log file.	reportPTest = new Report(pElection); reportPTest.createLog();	An audit file is created.	Election-Audit-File is produced.	Sometimes, randomly a NullPointerException is thrown at line 76 of createLog()
3	Create and fill a String array that contains the expected log output.	ArrayList<String> expectedLines = new ArrayList<String>();	A String array is created and filled.	Array expectedLines is created.	
4	Open log file and read through each line.	next = br.readLine()	Each line of the audit file is processed.	BufferedReader br opens Election-Audit-File and reads each line.	
5	Check if each line matches its corresponding expected String.	while ((next = br.readLine()) != null && iter.hasNext()) { assertEquals(iter.next(), next); }	True	True	When exception is thrown, result is not reached

Post condition(s) for Test:

A Plurality Election is run and an audit file is created.

Test Stage: Unit _X_ System __   **Test Date:** 4/2/2020

Test Case ID#: Report_2   **Name(s) of Testers:** Jessica Moore

Test Description:

Tests the createLog() function to see if it
records the results of an STV Election.

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

Automated: yes_X__ no __   ReportUT.java testSTVCreateLog()

Results: Pass __X___ Fail_____

Preconditions for Test:

A STVElection object has been constructed.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the election.	sElection.runElection();	An election is run.	sElection runs successfully.	

			An audit file is created.	Election-Audit-File is produced.	
2	Create a Report object and run createLog() to create a log file.	reportSTest = new Report(pElection); reportSTest.createLog();	An audit file is created.	Election-Audit-File is produced.	
3	Create and fill a String array that contains the expected log output.	ArrayList<String> expectedLines = new ArrayList<String>();	A String array is created and filled.	Array expectedLines is created.	
4	Open log file and read through each line.	next = br.readLine()	Each line of the audit file is processed.	BufferedReader br opens Election-Audit-File and reads each line.	
5	Check if each line matches its corresponding expected String.	while ((next = br.readLine()) != null && iter.hasNext()) { assertEquals(iter.next(), next); }	True	True	

Post condition(s) for Test:

An STV Election is run and an audit file is created.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_1   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the displayStats() function to see if it
correctly displays the required information
about the election.

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

Automated: yes__X_   no __   PluralityElectionUT,java testDisplayStats()

Results: Pass _____   Fail____X____

Preconditions for Test:

A PluralityElection object has been created and the runElection() function has been run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a printstream to capture any output to the terminal.	System.setOut(capture);	A printstream object captures all output to screen.	capture is set to capture output.	
2	Call displayStats() function to get actual output of the election.	pElection.displayStats();	displayStats() will produce output about the election.	capture captures the output resulting from displayStats().	

3	Create a string containing the expected output of displayStats().	String expectedString = "..."	A string is created.	expectedString contains the expected output.	
4	Check if the expected string matches the output of displayStats()	assertEquals(expectedString, res);	True	True	

Post condition(s) for Test:

The statistics of the election have been captured in a string.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_2   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the getNumSeats() function to see if it correctly returns the number of seats.

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

Automated: yes__X_   no __   PluralityElectionUT,java testGetNumSeats()

Results: Pass __X___   Fail_____

Preconditions for Test:

A PluralityElection object has been created and the runElection() function has been run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if getNumSeats() returns the expected number of seats.	assertEquals(2, pElection.getNumSeats());	True	True	

Post condition(s) for Test:

The election must have been processed.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_3   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the getNumCandidates() function to see if it correctly returns the number of seats.

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

Automated: yes__X_   no __   PluralityElectionUT,java testGetNumCandidates()

Results: Pass __X___   Fail_____

Preconditions for Test:

A PluralityElection object has been created and the runElection() function has been run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if getNumCandidates() returns the expected number of candidates.	assertEquals(4, pElection.getNumCandidates());	True	True	

Post condition(s) for Test:

The election must have been processed.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_4   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the getBallots() function to see if it
correctly returns the number of seats.

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

Automated: yes__X_   no __   PluralityElectionUT,java testGetBallots()

Results: Pass __X___   Fail_____

Preconditions for Test:

A PluralityElection object has been created and the runElection() function has been run.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create and fill an ArrayList of ballot objects as they are expected to appear.	ArrayList<Ballot> expectedVotes = new ArrayList<Ballot>();	An ArrayList is created and filled.	expectedVotes contains the expected ballots.	

		for (int i = 0; i < expectedVotes.size(); i++) { assertEquals(expectedVotes.get(i), pElection.getBallots().get(i)); }	True	True	
2	Check if each expected ballot object is equal to the ballot object that is given by getBallots().				

Post condition(s) for Test:

The election must have been processed.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_5   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the generateBallots() function to see if it correctly returns the number of seats.

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

Automated: yes__X_   no __   PluralityElectionUT,java testGenerateBallots()

Results: Pass __X___   Fail_____

Preconditions for Test:

Votes arrays for the expected Candidates of samplePluralityBallot.csv are created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create and fill an ArrayList of ballot objects as they are expected to appear.	ArrayList<Ballot> expectedVotes = new ArrayList<Ballot>();	An ArrayList is created and filled.	expectedVotes contains the expected ballots.	
2	Create a PluralityElection object with no seats and generate ballots.	pElectionNoSeat.generateBallots();	Ballots for the election are generated.	pElectionNoSeat has votes ArrayList filled.	
3	Check if generated ballots match expected ballots.	assertEquals(expectedVotes, pElectionNoSeat.getBallots());	True	True	

Post condition(s) for Test:

The no seats election has been created and processed.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_6   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the runElection() function line 86-93 to ensure the nonElected array is proper initialized and filled

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

Automated: yes__X_   no __

PluralityElectionUT,java testFillInitialNonElectedArray()

Results: Pass __X___   Fail_____

Preconditions for Test:

A PluralityElection object has been created and the runElection() function has been run. An expectedInital HashMap has been created and filled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object with no seats and run the election	pElectionNoSeat.runElection();	The election is run.	pElection is created and runElection() completes.	
2	Check if the expected initial HashMap matches the actual inital HashMap.	assertEquals(expectedInitial, pElection.getNonElected());	True	True	

Post condition(s) for Test:

The no seats election has been created and processed.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_7   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the runElection() function at the iterator loop in lines 101-111 to ensure the most popular candidate is properly identified and added to elected.

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

Automated: yes__X_   no __   PluralityElectionUT,java testGetElectedOneCandidate()

Results: Pass __X___   Fail_____

Preconditions for Test:

Votes arrays for the expected Candidates of samplePluralityBallot.csv are created.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a PluralityElection object with one seat and run the election	pElectionOneSeat.runElection();	The election is run.	pElectionOneSeat is created and runElection() completes.	
1	Create and fill an ArrayList of ballot objects as they are expected to appear.	ArrayList<Ballot> expectedWinner = new ArrayList<Ballot>();	An ArrayList is created and filled.	expectedWinner contains the expected ballots.	
2	Check if the expected losers ArrayList matches the actual losers ArrayList.	assertEquals(expectedWinner, pElection.getElected());	True	True	

Post condition(s) for Test:

The one seat election has been created and processed.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_8   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the getElected() function and test the
runElection() while loop at lines 95-113 to
ensure that multiple popular candidates can
be correctly identified and added to elected.

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

Automated: yes__X_   no __   PluralityElectionUT,java testGetElectedMultipleCandidates()

Results: Pass __X___   Fail_____

Preconditions for Test:

A PluralityElection object has been created and the runElection() function has been run. An expectedWinners
HashMap has been created and filled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if getElected() returns the expected HashMap of Candidate and Ballot objects.	assertEquals(expectedWinners, pElection.getElected());	True	True	

Post condition(s) for Test:

The election must have been processed.

Test Stage: Unit _X_   System __   **Test Date:** 4/2/2020

Test Case ID#: PluralityElection_9   **Name(s) of Testers:** Jessica Moore

Test Description:

Test the getNonElected() function to see if it
correctly returns the number of seats, and
test the runElection() lien 112 to ensure the
expected non-winners are left in nonelected

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

Automated: yes__X_   no __   PluralityElectionUT,java testGetNonElected()

Results: Pass __X___   Fail_____

Preconditions for Test:

A PluralityElection object has been created and the runElection() function has been run. An
expectedLosers ArrayList has been created and filled.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if getNonElected() returns the expected ArrayList of Candidate objects.	assertEquals(expectedLosers, pElection.getNonElected());	True	True	

Post condition(s) for Test:

The election must have been processed.

Test Stage: Unit _X_   System __   **Test Date: 4/1/2020**

Test Case ID#: Candidate_1   **Name(s) of Testers: Ian Luck**

Test Description: Testing getVotes()

Function and checking to see if it returns the correct value.

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

CandidateUT.java   getVotesTest()

Automated: yes_x__   no __

Results: Pass __x___   Fail_____

Preconditions for Test:

A candidate is properly set up.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check to see if the number of votes returned is 0.	assertEquals(candidate.getVotes(), 0)	True	True	Candidate object is initialized at the top of the class.
2	Add to the number of votes using addBallot()	candidate.addBallot()			

			True	True	
3	Check to see if the number of votes now returned is 1.	assertEquals(candidate.getVotes(), 1)			
4					

Post condition(s) for Test:

The correct number of votes has been returned by getVotes().

Test Stage: Unit _X_   System __   Test Date: 4/1/2020

Test Case ID#: Candidate_2   Name(s) of Testers: Ian Luck

Test Description: Testing the function getName() from Candidate.java to ensure that it returns a valid string representing the name.

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

Automated: yes_x_   no __

CandidateUT.java   getNameTest()

Results: Pass __x___   Fail_____

Preconditions for Test:

A candidate has a string value entered as a name and is not null or empty.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if the name of the candidate is returned properly	assertEquals(candidate.getName(), "Ian")	True	True	
2					
3					
4					

Post condition(s) for Test:

Correct string for the name is returned by getName().

Test Stage: Unit _X_   System __   **Test Date: 4/1/2020**

Test Case ID#: Candidate_3   **Name(s) of Testers: Ian Luck**

Test Description: Testing the function addBallot() from Candidate.java to ensure that it properly adds 1 to the vote count.

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

Automated: yes_x_   no __   **CandidateUT.java   addBallotTest()**

Results: Pass __x___   Fail_____

Preconditions for Test:

A candidate has been initialized and is not null.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check if the number of votes of the candidate is returned properly	assertEquals(candidate.getVotes(), 0)	True	True	
2	Use add a vote onto the numVotes.	candidate.addBallot()			
3	Use add a vote onto the numVotes.	candidate.addBallot()			
4	Check again to see if the correct number of votes has been added to the candidates numVotes.	assertEquals(candidate.getVotes(), 2)	True	True	

Post condition(s) for Test:

Correct string for the name is returned by getName().

Test Stage: Unit _X_   System __   **Test Date: 4/1/2020**

Test Case ID#: BallotGenerator_1   **Name(s) of Testers: Ian Luck**

**Test Description: This first test ensures
that the BallotGenerator function
createBallots() correctly generates
ballots when run by a plurality
ballotGenerator object.**

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

**BallotGeneratorUT.java
createPluralityBallotsTest()**

Automated: yes_x__   no __

Results: Pass __x___   Fail_____

Preconditions for Test:

A ballotGenerator has been initialized and has been given a correct file path and election type.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set newBallots ArrayList of Ballots equal to the ArrayList returned by the createBallots() function.	newBallots = Pluralitybg.createBallots(candidates);			The candidate HashMap and other variables used in the test were all set up in the setUp() method before this test ran.
2	Check that the first ballot in the arrayList is the same as the test candidates index showing that the ballots did not change in order after being created.	assertEquals(newBallots.get(0).getChoice().getIndex(), candidate2.getIndex());	True	True	

			True	True	
3	Check that the second ballot in the arrayList is the same as the test candidates index showing that the ballots did not change in order after being created.	*assertEquals(n ewBallots*.get(1).getChoice().g etIndex(), *candidate2*.getI ndex());	True	True	
4	Check that the third ballot in the arrayList is the same as the test candidates index showing that the ballots did not change in order after being created.	*assertEquals(n ewBallots*.get(2).getChoice().g etIndex(), *candidate2*.getI ndex());	True	True	

Post condition(s) for Test:

Correct Indexes for each ballot's candidates are returned ensuring createBallots() function for plurality ballotGenerator works.

Test Stage: Unit _X_ System __

Test Date: 4/20/2020

Test Case ID#: ballotGenerator_2

Name(s) of Testers: Zac Tressel

Test Description:This second test ensures that the BallotGenerator function createBallots() correctly generates ballots when run by a STV ballotGenerator object.

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

ballotGeneratorUT.java createSTVBallotsTest()

Automated: yes_x__ no __

Results: Pass __x___ Fail_____

Preconditions for Test:

A ballotGenerator object must be properly created to call the createBallots function with the election type being STV and a working file location.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a ballot generator object that doesn't shuffle	BallotGenerator bg1 = new BallotGenerator ("testing/sampl eSTVBallot.csv ", false, "STV");	A ballot generator object has successfully been created	A ballot generator object has successfully been created	An STV sample file is used for consistency but it doesn't really matter
2	Create another ballot generator object identical to the first	BallotGenerator bg2 = new BallotGenerator ("testing/sampl eSTVBallot.csv ", false, "STV");	A ballot generator object has successfully been created	A ballot generator object has successfully been created	
3	Test to make sure the two ballot generators that don't shuffle return the same array of votes	assertEquals(b g1.createBallot s(candidates), bg2.createBallo ts(candidates));	True	True	
4	Create a ballot generator object that does shuffle and uses the same file as step 1	BallotGenerator bg3 = new BallotGenerator ("testing/sampl eSTVBallot.csv ", true, "STV");	A ballot generator object has successfully been created	A ballot generator object has successfully been created	

			True	True	
5	Test to make sure that the ballot generator that does shuffle returns a different array of votes than the ballot generator that doesn't shuffle	assertNotEquals(bg3.createBallots(candidates), bg2.createBallots(candidates));			
6	Create a ballot generator object that does shuffle and uses the same file as step 1	BallotGenerator bg4 = new BallotGenerator("testing/sampleSTVBallot.csv", true, "STV");	A ballot generator object has successfully been created	A ballot generator object has successfully been created	
7	Test to make sure the two ballot generators that shuffle don't return the same array of votes	assertNotEquals(bg3.createBallots(candidates), bg4.createBallots(candidates));	True	True	

Post condition(s) for Test:

The ballot generator correctly generates ballots and shuffles the array of ballots in a non-deterministic way, when desired.

Test Stage: Unit _X_   System __   **Test Date: 4/2/2020**

Test Case ID#: ballotGenerator_3   **Name(s) of Testers: Ian Luck**

Test Description: This test wants to ensure that the shuffle() function included in createBallots() properly shuffles the ballots.

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

ballotGeneratorUT.java shuffleTest()

Automated: yes_x__   no __

Results: Pass __x___   Fail_____

Preconditions for Test:

A STV ballotGenerator object must be correctly initialized and must have the shuffle boolean set to true in order for the test to run and must have a proper file location of the .csv file.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Call the STV ballotGenerator function shuffle which will modify the given ballot ArrayList.	*STVbg*.shuffle(*testBallots*, *testBallots*.size());			Variables used in the test were all set up in the setUp() method before this test ran. B1 and b3 are STV ballots also initialized inthe setup that match the ballots in the newBallots arrayList for testing purposes.
2	Check that the newly shuffled arraylist newBallots first entry is not the	*assertNotEquals*(*testBallots*.get(0), *b1*);	True	True	If the test does fail, run again because the shuffle functionality of

	same as the original because then it shows that the ballots have been shuffled.				createBallots() has been proven to sometimes place the shuffled ballot back in its original position if the number of ballots in the list is relatively small.
3	Check that the newly shuffled arraylist newBallots third entry is not the same as the original because then it shows that the ballots have been shuffled.	*assertNotEquals(testBallots*.get (2), *b3*);	True	True	If the test does fail, run again because the shuffle functionality of createBallots() has been proven to sometimes place the shuffled ballot back in its original position if the number of ballots in the list is relatively small.
4					

Post condition(s) for Test:

The arraylist of ballots initially given as the parameter to the shuffle function is properly shuffled and the STV ballotGenerator object's original ballots arraylist is now shuffled.

Project Name: Project 1: Voting System   **Team#** 10

   Test Stage: Unit _X_   System __   **Test Date:** 4/2

   Test Case ID#: STVElection_1   **Name(s) of Testers:** Isabel Tomb

Test Description:
Test the STVElection constructor method to see if it properly
sets the numSeats and shuffled variables.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test:
STVElection object has been created using the STVElection constructor method that was passed the following variables: numSeats = 10,
ballot file location = "sampleSTVElection.csv", shuffle = true.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the actual numSeats variable is equal to the expected numSeats value	assertEquals(stvelection.getNumSeats(), 10);	True	True	
2	Check that the actual isShuffled variable matches the expected isShuffled value	assertEquals(stvelection.isShuffled(), true);	True	True	

Post condition(s) for Test:

A new STVElection object exists with the variables numSeats = 10, ballot file location = "sampleSTVElection.csv", shuffle = true

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_2   **Name(s) of Testers:** Isabel Tomb

Test Description:
Test to see that the getLosers() method correctly returns the
ArrayList of losers associated with the STVElection object

                                                                **Indicate where are you storing the tests (what file) and the name
                                                                of the method/functions being used.**
Automated: yes_X_no ___   STVElection.java getLosers()

Results: Pass __X__   Fail_____

Preconditions for Test:
STVElection object has been created using the STVElection constructor method that was passed the following variables: numSeats = 10,
ballot file location = "sampleSTVElection.csv", shuffle = true.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new empty ArrayList of Candidate objects	ArrayList<Candidate> losersTest = new ArrayList<Candidate>();	A new ArrayList of Candidate objects is created	New empty ArrayList of candidate objects, named losersTest, is created	
2	Check that the losers ArrayList associated with the STVElection object is equal to the empty ArrayList, losersTest	assertEquals(stvelection.getLosers(), losersTest);	True	True	

Post condition(s) for Test:
The losers ArrayList associated with the STVElection object is returned successfully.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_3   **Name(s) of Testers:** Isabel Tomb

Test Description:
 Tests that the isShuffled() method successfully returns the
boolean value shuffle, associated with a STVElection object,
indicating if the ballots should be shuffled or not.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test:
 STVElection object has been created using the STVElection constructor method that was passed the following variables: numSeats = 10,
 ballot file location = "sampleSTVElection.csv", shuffle = true.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the boolean value shuffle associated with the STVElection object, stvelection, is true	assertTrue(stvelection.isShuffled());	True	True	

 Post condition(s) for Test:
The boolean value, shuffle, associated with the STVElection object, stvelection, is returned.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_4   **Name(s) of Testers:** Isabel Tomb

Test Description:
Test that the getNumSeats() method successfully returns the
numSeats value associated with the STVElection object.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test:
STVElection object has been created using the STVElection constructor method that was passed the following variables: numSeats = 10,
ballot file location = "sampleSTVElection.csv", shuffle = true.

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

	Check that the numSeats value returned by getNumSeats() is equal to the the value set by the constructor	assertEquals(stvelection.getNumSeats(), 10);	True	True	
1					

Post condition(s) for Test:
The value, numSeats, associated with the STVElection object, stvelection, is returned.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_5   **Name(s) of Testers:** Isabel Tomb

Test Description:
 Test that the getElected method successfully returns the
 STVElection HashMap containing the elected candidates.

                                                                       Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Automated: yes_X_no ___   STVElection.java getElectedTest()

 Results: Pass __X__   Fail_____

Preconditions for Test:

STVElection object has been created using the STVElection constructor method that was passed the following variables: numSeats = 10, ballot file location = "sampleSTVElection.csv", shuffle = true.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create an new empty HashMap called electedTest	HashMap<Candidate, ArrayList<Ballot>> electedTest = new HashMap<Candidate, ArrayList<Ballot>>();	A new empty HashMap with the key being a Candidate object and the value being an ArrayList of Ballot objects, is created	An empty HashMap called electedTest, with key type Candidate and value type ArrayList of Ballots is created	
2	Check that the elected hashmap returned by getElected is equal to the electedTest HashMap created	assertEquals(stvelection. getElected(), electedTest);	True	True	

Post condition(s) for Test:

The elected HashMap variable associated with the STVElection object, stvelection, is returned.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_6   **Name(s) of Testers:** Isabel Tomb

Test Description:
Test that the getNonElected method successfully returns the
STVElection HashMap containing the nonElected candidates.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test:
STVElection object has been created using the STVElection constructor method that was passed the following variables: numSeats = 10,
ballot file location = "sampleSTVElection.csv", shuffle = true.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create an new empty HashMap called nonElectedTest	HashMap<Candidate, ArrayList<Ballot>> nonElectedTest = new HashMap<Candidate, ArrayList<Ballot>>();	A new empty HashMap with the key being a Candidate object and the value being an ArrayList of Ballot objects, is created	An empty HashMap called nonElectedTest, with key type Candidate and value type ArrayList of Ballots is created	

			True	True	
2	Check that the nonElected hashmap returned by getNonElected is equal to the nonElectedTest HashMap created	assertEquals(stvelection.getNonElected(), nonElectedTest);			

Post condition(s) for Test:
The nonElected HashMap variable associated with the STVElection object, stvelection, is returned.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_7   **Name(s) of Testers:** Isabel Tomb

Test Description:
Tests that the getDroop() function returns the boolean value,
droop, associated with an STVElection object.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test:

STVElection object has been created using the STVElection constructor method that was passed the following variables: numSeats = 10, ballot file location = "sampleSTVElection.csv", shuffle = true.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Check that the value returned by getBallots() is equal to the expected, initialized value.	assertEquals(stvelection.getDroop(), 0);	True	True	

Post condition(s) for Test:
The droop integer value associated with an STVElection object is returned.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_8

Name(s) of Testers: Isabel Tomb

Test Description:
Tests that the getBallots() method returns the ArrayList
of Ballot objects associated with a STVElection object.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test:
STVElection object has been created using the STVElection constructor method that was passed the following variables: numSeats = 10,
ballot file location = "sampleSTVElection.csv", shuffle = true.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new empty ArrayList of Ballot objects	ArrayList<Ballot> votesTest = new ArrayList<Ballot>();	New empty ArrayList of Ballot objects created	Empty ArrayList of Ballot objects called votesTest created	
2	Check that the ArrayList of ballots returned by the generateBallots()	assertEquals(stvelection.getBallots(), votesTest);	True	True	

	function is equal to the empty				

Post condition(s) for Test:
The ArrayList of ballots associated with the STVElection object, stvelection is returned.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_9   **Name(s) of Testers:** Isabel Tomb

Test Description:
Tests that the generateBallots() function successfully
creates a ballotGenerator object and calls the createBallot()
function on it to generate an ArrayList of Ballots.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new STVElection object stvelection2	STVElection stvelection2 = new STVElection(2, "testing/sampleSTVBallot2.csv", false);	A new STVElection is created	STVElection object created with the inputs: numSeats = 2, ballot file location = "testing/sampleSTVBallot2.csv", shuffle = false	
2	Create a new ArrayList of Ballot objects	ArrayList<Ballot> votesTest = new ArrayList<Ballot>();	A new empty arraylist of ballot objects is created	ArrayList of Ballot objects created and named votesTest	
3	7 STVBallot objects created with integer arrays	STVBallot testballot1 = new STVBallot(v1); STVBallot testballot2 = new STVBallot(v2); STVBallot testballot3 = new STVBallot(v3); STVBallot testballot4 = new STVBallot(v4); STVBallot testballot5 = new STVBallot(v5); STVBallot testballot6 = new STVBallot(v6); STVBallot testballot7 = new STVBallot(v7);	7 different ballot objects, with the same integer arrays as the ballots objects in the test file, are created.	testballot1-testballot7 ballot objects created representing the ballot objects in "testing/sampleSTVBallot2.csv"	

			7 test ballots added to the ArrayList votesTest	7 test ballots added to the ArrayList votesTest	
4	7 test ballots are added to the votesTest ArrayList created in step 2	votesTest.add(testballot1); votesTest.add(testballot2); votesTest.add(testballot3); votesTest.add(testballot4); votesTest.add(testballot5); votesTest.add(testballot6); votesTest.add(testballot7);			
5	Check that the votesTest array is equal to the ArrayList returned by the generateBallots() method and set equal to the STVElection votes variable	assertEquals(votesTest, stvelection2.getBallots());	True	True	

Post condition(s) for Test:
The votes variable in the STVElection stvelection2 is set to be the ArrayList of ballots returned by the generateBallots() method.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_10   **Name(s) of Testers:** Isabel Tomb

Test Description:
 Tests that when the STVElection method, runElection() is run
with ballots that cause it to only need one pass for all seats to be
filled, it returns the correct results.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new STVElection object stvelection3	STVElection stvelection2 = new STVElection(2, "testing/sampleSTVBallot3.csv", false);	A new STVElection is created	STVElection object created with the inputs: numSeats = 2, ballot file location = "testing/sampleSTVBallot3.csv", shuffle = false	
2	Create 3 new Candidate objects	Candidate testCand0 = new Candidate(0, "A");	Three new candidate objects created with	Three candidate objects created that are equivalent to the candidates in sampleSTVBallot3.csv	

		Candidate testCand1 = new Candidate(1, "B"); Candidate testCand2 = new Candidate(2, "C");			
3	ArrayList expectedLosers created and test candidate added to it	ArrayList<Candidate> expectedLosers = new ArrayList<Candidate>(); expectedLosers.add(test Cand2);	New expectedLosers ArrayList of candidate objects created and losing candidate added	The expected loser, testCand2 is added to the new expectedLosers ArrayList	
4	ArrayList expectedWinners created and test candidate added to it	ArrayList<Candidate> expectedWinners = new ArrayList<Candidate>(); expectedWinners.add(te stCand2);	New expectedWinners ArrayList of candidate objects created and winning candidates added	The expected winner, testCand0, and 1 are added to the new expectedWinners ArrayList	
5	Run election	stvelection3.RunElection ()	RunElection method called on the stvelection3 election object	RunElection method called on the stvelection3 election object	
6	Check that the droop value, elected candidates and losing candidate values are as expected	assertEquals(stvelection 3.getDroop(), 3); assertEquals(expectedW inners, testWinners); assertEquals(expectedL	True	True	

		osers, stvelection3.getLosers());			

Post condition(s) for Test:
STVElection has been run and droop,elected and losers values have all been set.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_11   **Name(s) of Testers:** Isabel Tomb

Test Description:
 Tests that when the STVElection method, runElection() is run
with ballots that cause it to reach the second loop in the method
which simulates the second pass, it returns the correct result.

                                                                **Indicate where are you storing the tests (what file) and the name
                                                                of the method/functions being used.**
Automated: yes_X_no ___   STVElection.java runElectionSecondPassTest()

 Results: Pass __X__   Fail_____

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a new STVElection object stvelection3	STVElection stvelection2 = new STVElection(2, "testing/sampleSTVBallot2.csv", false);	A new STVElection is created	STVElection object created with the inputs: numSeats = 2, ballot file location = "testing/sampleSTVBallot2.csv", shuffle = false	
2	Create 3 new Candidate objects	Candidate testCand0 = new Candidate(0, "A"); Candidate testCand1 = new Candidate(1, "B"); Candidate testCand2 = new Candidate(2, "C");	Three new candidate objects created with	Three candidate objects created that are equivalent to the candidates in sampleSTVBallot2.csv	
3	ArrayList expectedLosers created and test candidate added to it	ArrayList<Candidate> expectedLosers = new ArrayList<Candidate>(); expectedLosers.add(testCand2);	New expectedLosers ArrayList of candidate objects created and losing candidate added	The expected loser, testCand2 is added to the new expectedLosers ArrayList	

4	ArrayList expectedWinners created and test candidate added to it	ArrayList<Candidate> expectedWinners = new ArrayList<Candidate>(); expectedWinners.add(testCand2);	New expectedWinners ArrayList of candidate objects created and winning candidates added	The expected winner, testCand0, and 1 are added to the new expectedWinners ArrayList	
5	Run election	stvelection3.RunElection()	RunElection method called on the stvelection3 election object	RunElection method called on the stvelection3 election object	
6	Check that the droop value, elected candidates and losing candidate values are as expected	assertEquals(stvelection3.getDroop(), 3); assertEquals(expectedWinners, testWinners); assertEquals(expectedLosers, stvelection3.getLosers());	True	True	

Post condition(s) for Test:
STVElection has been run and droop,elected and losers values have all been set.

Project Name: Project 1: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_12   **Name(s) of Testers:** Isabel Tomb

Test Description:
 Tests that when the STVElection method, runElection() is run
with ballots that cause it to reach the second loop in the method
which simulates the second pass, it returns the correct result.

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

Automated: yes_X_no ___

Results: Pass __X__   Fail_____

Preconditions for Test:

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

1	Create a new STVElection object stvelection3	STVElection stvelection2 = new STVElection(3, "testing/sampleSTVBallot4.csv", false);	A new STVElection is created	STVElection object created with the inputs: numSeats = 3, ballot file location = "testing/sampleSTVBallot4.csv", shuffle = false	
2	Create 4 new Candidate objects	Candidate testCand0 = new Candidate(0, "A"); Candidate testCand1 = new Candidate(1, "B"); Candidate testCand2 = new Candidate(2, "C"); Candidate testCand3 = new Candidate(3, "D");	Four new candidate objects created with	Four candidate objects created that are equivalent to the candidates in sampleSTVBallot4.csv	
3	ArrayList expectedLosers created and test candidate added to it	ArrayList<Candidate> expectedLosers = new ArrayList<Candidate>(); expectedLosers.add(testCand3);	New expectedLosers ArrayList of candidate objects created and losing candidate added	The expected loser, testCand3 is added to the new expectedLosers ArrayList	
4	ArrayList expectedWinners created and test candidate added to it	ArrayList<Candidate> expectedWinners = new	New expectedWinners ArrayList of candidate objects created and winning candidates added	The expected winner, testCand0, 1 and 2 are added to the new expectedWinners ArrayList	

		ArrayList<Candidate >(); expectedWinners.ad d(testCand0); expectedWinners.ad d(testCand1); expectedWinners.ad d(testCand2);			
5	Run election	stvelection3.RunElec tion()	RunElection method called on the stvelection3 election object	RunElection method called on the stvelection3 election object	
6	Check that the droop value, elected candidates and losing candidate values are as expected	assertEquals(stvelect ion3.getDroop(), 3); assertEquals(expect edWinners, testWinners); assertEquals(expect edLosers, stvelection3.getLoser s());	True	True	

Post condition(s) for Test:
STVElection has been run and droop,elected and losers values have all been set.

Project Name: Project 1: Voting System   **Team# 10**

Test Stage: Unit _X_   System __   **Test Date:** 4/2

Test Case ID#: STVElection_13   **Name(s) of Testers:** Isabel Tomb

Test Description:

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

Automated: yes_X_no ___   STVElection.java

Results: Pass __X__   Fail_____

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	New election object stvelection3 is created and run and displayStats() is called on it	STVElection stvelection3 = new STVElection(2, "testing/sampleSTVBallot2.csv", false);	Election object is created, runElection is called on it then displayStats is called on it	Election object is created, runElection is called on it then displayStats is called on it	

		stvelection3.runElection(); stvelection3.displayStats();			
1	Create a printstream to capture any output to the terminal.	System.setOut(ps);	A printstream object captures all output to screen.	ps is set to capture output.	
2	Call displayStats() function on the STVElection object to get actual output of the election.	stvelection3.displayStats();	displayStats() will produce output about the election.	capture captures the output resulting from displayStats().	
3	Create a string containing the expected output of displayStats().	String expectedDisp = "..."	A new string object is created.	expectedDisp contains the expected output.	
4	Check if the expected string and the output of displayStats() are equivalent	assertEquals(expectedDisp, res);	True	True	

Post condition(s) for Test:

Stats are displayed for the stvelection3 STVElection object after the election has been run.

Test Stage: Unit _X_   System __   Test Date: 4/2/2020

Test Case ID#: ballotGenerator_4

Name(s) of Testers: Ian Luck

Test Description: This test wants to ensure that the createBallots() function will also be able to handle when it is given a non-empty hashmap and is called by an Plurality Ballot Generator.

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

ballotGeneratorUT.java
createBallotsTest2()

Automated: yes_x__   no __

Results: Pass __x___   Fail_____

Preconditions for Test:

A Plurality ballotGenerator object must be correctly initialized and must have the shuffle boolean set to false in order for the test to run and must have a proper file location of the .csv file.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	In these small steps the test is set up by creating an example plurality ballot arraylist and adding a ballot to it to make it non-empty.	ArrayList<Ballot> t2 = new ArrayList<Ballot>(); int[] votes = {0,1,0}; PluralityBallot p = new PluralityBallot(votes); t2.add(p);			Variables used in the test were all set up in the setUp() method before this test ran.
2	Sets the first array to the value with the empty candidate hashmap given to createBallots().	testBallots = Pluralitybg.createBallots(candidates);			
3	Adds a candidate and respective Ballot ArrayList to the Hashmap.	candidates.put(candidate2, t2);			

4	Calls createBallots() again this time with a non-empty HashMap.	*newBallots = Pluralitybg*.crea teBallots(*candid ates*);			
5	Checks to see that the first value is the same ensuring that the HashMap being empty or not does not affect generating correct ballots.	*assertEquals*(*n ewBallots*.get(0), *testBallots*.get(0));	True	True	

Post condition(s) for Test:

The createBallots() function successfully creates a new ArrayList of Ballots despite being fed a non-empty HashMap.

Test Stage: Unit _X_   System __   **Test Date: 4/2/2020**

Test Case ID#: ballotGenerator_5   **Name(s) of Testers: Ian Luck**

Test Description: This test wants to ensure that the createBallots() function will also be able to handle when it is given a non-empty hashmap and is called by an STV Ballot Generator.

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

ballotGeneratorUT.java
createBallotsTest3()

Automated: yes_x__   no __

Results: Pass __x___   Fail_____

Preconditions for Test:

 A STV ballotGenerator object must be correctly initialized and must have the shuffle boolean set to true in order for the test to run and must have a proper file location of the .csv file.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	In these small steps the test is set up by creating an example STV ballot arraylist and adding a ballot to it to make it non-empty.	ArrayList<Ballot> t3 = new ArrayList<Ballot>(); STVBallot s = new STVBallot(*v1*); t3.add(s);			Variables used in the test were all set up in the setUp() method before this test ran.
2	Sets the first array to the value with the empty candidate hashmap given to createBallots().	*testBallots* = *STVbg*.createBallots(*candidates*);			
3	Adds a candidate and respective Ballot ArrayList to the Hashmap.	*candidates*.put(*candidate1*, t3);			
4	Calls createBallots() again this time	*newBallots* = *STVbg*.createB			

	with a non-empty HashMap.	allots(*candidates*);			
5	Checks to see that the first value is not the same ensuring that the HashMap being empty or not does not affect generating correct ballots that have been shuffled successfully.	*assertNotEquals(newBallots*.get(0), *testBallots*.get(0));	True	True	

Post condition(s) for Test:

The createBallots() function successfully creates a new ArrayList of Ballots despite being fed a non-empty HashMap.

Test Stage: Unit ___   System _X_   **Test Date: 4/2/2020**

Test Case ID#: PluralitySystem_1   **Name(s) of Testers: Isabel Tomb**

Test Description: This test wants to ensure
that the system as a whole will run plurality
elections and give the expected output

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

PluralitySystemTest.java,
testPluralityNoTieWinners(),
testPluralityNoTieLosers(),
testPluralityTieWinners(),
testPluralityTieLosers()

Automated: yes_x__   no __

Results: Pass __x___   Fail_____

Preconditions for Test:

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set up expected winner hashmap with equivalent ballots to the ballot file that was passed in for testing	3 candidates for the first election, 4 candidates for the second election	Create new hashmap of candidate and arraylist ballot objects containing winners	Created new hashmap of candidate and arraylist ballot objects containing winners and their ballots	
2	Set up expected losers hashmap equivalent to the expected result of the ballot file	One loser in the first election, two losers for the second	New hashmap created of type candidate key with arraylist ballot value	Created new hashmap of candidate and arraylist ballot objects containing the losers and their ballots	
3	Run the election on both election objects		Election is run	Election is run	
4	Compare the expected		True	True	This sometimes fails because of

				a known null pointer exception that is documented in the bug list

Post condition(s) for Test:

Two elections testing the situation where there is no tie and the situation where there is a tie are run.

Project Name: Project 2: Voting System   **Team#** 10

Test Stage: Unit _X_   System __   **Test Date:** 4/18

Test Case ID#: BallotGenerator_7   **Name(s) of Testers:** Isabel Tomb

Test Description:
Test the invalidator method in the BallotGenerator class to ensure
It successfully removes any ballots where more than half the
candidates are not ranked.

                                                                        Indicate where are you storing the tests (what file) and the name of the method/functions being used.
Automated: yes_X_no ___   BallotGeneratorUT.java InvalidatorTest()

Results: Pass __X__   Fail_____

Preconditions for Test:
N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create six integer arrays representing the votes arrays in a ballot	int[] votes1 = new int[] {1,0,2,0,3,4}; int[] votes2 = new int[] {0,0,2,0,1,0}; int[] votes3 = new int[] {1,0,0,0,2,0}; int[] votes4 = new int[] {1,2,3,4,5,0}; int[] votes5 = new int[] {1,2,0,0,0,0}; int[] votes6 = new int[] {2,3,1,4,0,0};	Six integer arrays created	Six integer arrays created	

	Create six STVballot objects and set the six integer arrays to be the votes array attribute associated with them	testBallots.add(testBallot1);
		testBallots.add(testBallot2);
		testBallots.add(testBallot3);
		testBallots.add(testBallot4);
		testBallots.add(testBallot5);
		testBallots.add(testBallot6);
			Six ballot objects created and votes arrays assigned to them	Six ballot objects created and votes arrays assigned to them	

	Expected arraylist of valid ballots created	ArrayList<STVBallot> expectedValid = new ArrayList<STVBallot>(); expectedValid.add(testBallot1); expectedValid.add(testBallot4); expectedValid.add(testBallot6);	The valid ballot objects are added to the array representing the expected outcome	Ballots successfully added to arraylist of expected ballots	
	Call invalidator method on arraylist of all test ballots	invalidator(testBallots)	Invalidator method run	Invalidator method successfully run	
	Compare actual result to expected result	assertEquals(expectedValid, testBallots);	True	True	

Post condition(s) for Test:

The invalidator method has been run on an ArrayList of ballots and successfully removes the ballots where more than half of the candidates are not ranked.

Test Stage: Unit _X_   System __   **Test Date:** 4/21/20

Test Case ID#: FileChooser_1   **Name(s) of Testers:** Jess Moore

Test Description:

Tests the functionality of the file selection
GUI with one file chosen.

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

Automated: yes __   no _X_   N/A

Results: Pass _X_   Fail ___

Preconditions for Test:

None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run main file to start the election.	None	Program begins	Program begins successfully	
2	Enter number of candidates on command line.	2	File selection GUI appears	File selection appears on screen	
3	Choose one ballot file and choose "Open"	Choose testing/sampleSTVballot.csv	Selected file path is displayed correctly.	Full file path is displayed for testing/sampleSTVballot.csv	

4	Enter election type and begin election.	S (for STV election)	Election runs and correctly displays results	Election runs successfully and correct results are displayed.	

Post condition(s) for Test:

Process finished with exit code 0

Test Stage: Unit _X_   System __   **Test Date:** 4/21/20

Test Case ID#: FileChooser_2   **Name(s) of Testers:** Jess Moore

Test Description:

Tests the functionality of the file selection GUI with multiple files chosen.

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

Automated: yes __   no _X_   N/A

Results: Pass _X_   Fail ___

Preconditions for Test:

None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run main file to start the election.	None	Program begins	Program begins successfully	
2	Enter number of candidates on command line.	2	File selection GUI appears	File selection appears on screen	
3	Choose multiple ballot files and choose "Open"	Choose testing/sampleSTVballot2.csv, testing/sampleSTVballot3.csv, testing/sampleSTVballot4.csv, & testing/sampleSTVballot5.csv	Selected file path is displayed correctly.	Full file path is displayed for testing/sampleSTVballot.csv	
4	Enter election type and begin election.	S (for STV election)	Election runs and correctly displays results	Election runs successfully and correct results are displayed.	

Post condition(s) for Test:

Process finished with exit code 0

Test Stage: Unit _X_   System __   **Test Date:** 4/21/20

Test Case ID#: FileChooser_1   **Name(s) of Testers:** Jess Moore

Test Description:

Tests the functionality of the file selection
GUI with one file chosen.

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

Automated: yes __   no _X_   N/A

Results: Pass _X_   Fail ___

Preconditions for Test:

None

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run main file to start the election.	None	Program begins	Program begins successfully	

			File selection GUI appears	File selection appears on screen	
2	Enter number of candidates on command line.	2			
3	Choose one ballot file and choose "Open"	Choose testing/sampleS TVballot.csv	Selected file path is displayed correctly.	Full file path is displayed for testing/sampleSTVballo t.csv	
4	Enter election type and begin election.	S (for STV election)	Election runs and correctly displays results	Election runs successfully and correct results are displayed.	

Post condition(s) for Test:

Process finished with exit code 0

Test Stage: Unit _X_   System __   **Test Date:** 4/21/20

Test Case ID#: InvalidBallotReport_1   **Name(s) of Testers:** Ian Luck

Test Description:

Testing the functionality of the invalidBallotReport() function in the BallotGenerator class.

Automated: yes _x_   no __

Results: Pass _X_   Fail ___

Preconditions for Test:

 An instance of invalidator must be correctly created and must call this function feeding it the array list of invalid ballots.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set Up arraylist and expected string values in the list.	ArrayList<String> expectedLines = new ArrayList<String>(); expectedLines.add("Number of Invalidated Ballots: 3"); expectedLines.add("Percentage of Ballots that were Invalidated: 50%"); expectedLines.add("Ratio of Valid Ballots to Invalidated Ballots: 1:1");			

		expectedLines.add("List of Invalidated Ballots:");
		expectedLines.add("[0, 0, 2, 0, 1, 0, 3]");
		expectedLines.add("[1, 0, 0, 0, 2, 0, 0]");
		expectedLines.add("[1, 2, 0, 0, 0, 0, 3]");			

2	Run through created invalid ballot file and set up loop to check data.	BufferedReader br = new BufferedReader(new FileReader("Invalid-Ballots-Report")); String next; Iterator<String> iter = expectedLines.iterator(); while ((next = br.readLine()) != null && iter.hasNext()){			
3	Check each line of file to ensure similar values as expected.	*assertEquals*(iter.next(), next);	True	True	This assertEquals is part of a loop so will run and check each line.

Post condition(s) for Test:

Test passes and invalid report file is properly generated and in correct location.

Test Stage: Unit _X_   System __   **Test Date:** 4/21/20

Test Case ID#: InvalidBallotReport_2   **Name(s) of Testers:** Ian Luck

Test Description:

Testing the functionality of the
invalidBallotReport() function in the
BallotGenerator class.

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

**BallotGeneratorUT.Java
invalidBallotsReportTest2()**

Automated: yes _x_   no __

Results: Pass _X_   Fail ___

Preconditions for Test:

An instance of invalidator must be correctly created and must call this function feeding it the
array list of invalid ballots.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Set Up arraylist and expected string values in the list.	ArrayList<String> expectedLines = new ArrayList<String>(); expectedLines.add("Number of Invalidated Ballots: 0"); expectedLines.add("Percentage of Ballots that were Invalidated: 0%"); expectedLines.add("Ratio of Valid Ballots to Invalidated Ballots: 1:1"); expectedLines.add("No invalidated ballots to display.");			
2	Run through created invalid ballot file and set up loop to check data.	BufferedReader br = new BufferedReader(new FileReader("Invalid-Ballots-Report")); String next; Iterator<String> iter = expectedLines.iterator(); while ((next = br.readLine()) != null && iter.hasNext()){			
3	Check each line of file to ensure similar values as expected.	*assertEquals*(iter.next(), next);	True	True	This assertEquals is part of a loop

				so will run and check each line.

Post condition(s) for Test:

Test passes and invalid report file is properly generated and in correct location.