Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_01	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When selecting ballot files to load, the user should be able to navigate through their file system. That is, they should be able to move through directory structures. This test checks that, when selecting files to load (not a directory), the user can navigate and sees only directories and files with the *.csv extension and can only select said files.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “BallotFileChooser.java”. This test makes use of the “ballot_file_chooser_test_dir” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The “ballot_file_chooser_test_dir” directory exists and contains the following: “dummy.txt”, “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, “test_5.csv”, and the subdirectories “test_sub_dir” and “test_empty_sub_dir”. “test_sub_dir” should contain “dummy.txt”, “test_7.csv”, “test_8.csv”, “test_9.csv”, and “test_10.csv”. “test_empty_sub_dir” should contain “dummy.txt”. No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Launch the BallotFileChooser in its file choosing mode. This can be done in parallel with another unit test for the BallotFileChooser.				Run this test alongside another test where selecting multiple files is tested.
2	Navigate to the “ballot_file_chooser_test_dir”.		The GUI should allow the user to get to the directory.	Was able to get to the directory through the GUI.	
3	Check that the GUI shows the subdirectories as well as files.		The GUI should show the subdirectories “test_sub_dir” and “test_empty_sub_dir” as well as “dummy.txt”, “test_1.csv”,	The GUI showed all of the correct subdirectories and files.	

			“test_2.csv”, “test_3.csv”, “test_4.csv”, and “test_5.csv”.		

Postconditions for Test: N/A

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_02	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When selecting ballot files to load, the user should be able to navigate through their file system. That is, they should be able to move through directory structures. This test checks that, when selecting a directory to load (not files), the user can navigate and see only directories and can only select said directories.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “BallotFileChooser.java”. This test makes use of the “ballot_file_chooser_test_dir” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The “ballot_file_chooser_test_dir” directory exists and contains the following: “dummy.txt”, “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, “test_5.csv”, and the subdirectories “test_sub_dir” and “test_empty_sub_dir”. “test_sub_dir” should contain “dummy.txt”, “test_7.csv”, “test_8.csv”, “test_9.csv”, and “test_10.csv”. “test_empty_sub_dir” should contain “dummy.txt”. No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Launch the BallotFileChooser in its directory choosing mode. This can be done in parallel with another unit test for the BallotFileChooser.				Run this test alongside another test where selecting a directory is tested.
2	Navigate to the “ballot_file_chooser_test_dir”.		The GUI should allow the user to get to the directory.	Was able to get to the directory through the GUI.	
3	Check that the GUI shows the subdirectories, but not any files.		The GUI should show the subdirectories “test_sub_dir” and “test_empty_sub_dir”, but no files.	The GUI showed all of the correct subdirectories and no files.	

Postconditions for Test: N/A

Straightforward Voting System - Testing Document — Team #2

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_03	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When using the file chooser, the user has the option to abort the selection process. This should be recognized by BallotFileChooser. This test ensures that 'null' - the value signifying such a process - is returned when selecting files (not a directory).	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "BallotFileChooser.java". The test method is 'test_choose_ballot_files_abort()'. This test makes use of the "ballot_file_chooser_test_dir" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The "ballot_file_chooser_test_dir" directory exists and contains the following: "dummy.txt", "test_1.csv", "test_2.csv", "test_3.csv", "test_4.csv", "test_5.csv", and the subdirectories "test_sub_dir" and "test_empty_sub_dir". "test_sub_dir" should contain "dummy.txt", "test_7.csv", "test_8.csv", "test_9.csv", and "test_10.csv". "test_empty_sub_dir" should contain "dummy.txt". No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the 'test_choose_ballot_files_abort()' unit test. This is a partially automated test.				Running all of the unit tests may not run the test cases in order. Just follow the prompts given to you in any case.
2	Abort the selection process by closing the file chooser or by pressing the 'cancel' button.				This step must be done manually.
3	Check that 'null' was returned.		The 'null' value should be returned.	The 'null' value was returned.	

Postconditions for Test: The ‘null’ value was returned by BallotFileChooser.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_04	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When using the file chooser, the user has the option to abort the selection process. This should be recognized by BallotFileChooser. This test ensures that 'null' - the value signifying such a process - is returned when selecting a directory (not files).	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "BallotFileChooser.java". The test method is 'test_choose_directory_abort()'. This test makes use of the "ballot_file_chooser_test_dir" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The "ballot_file_chooser_test_dir" directory exists and contains the following: "dummy.txt", "test_1.csv", "test_2.csv", "test_3.csv", "test_4.csv", "test_5.csv", and the subdirectories "test_sub_dir" and "test_empty_sub_dir". "test_sub_dir" should contain "dummy.txt", "test_7.csv", "test_8.csv", "test_9.csv", and "test_10.csv". "test_empty_sub_dir" should contain "dummy.txt". No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the 'test_choose_directory_abort()' unit test. This is a partially automated test.				Running all of the unit tests may not run the test cases in order. Just follow the prompts given to you in any case.
2	Abort the selection process by closing the file chooser or by pressing the 'cancel' button.				This step must be done manually.
3	Check that 'null' was returned.		The 'null' value should be returned.	The 'null' value was returned.	

Postconditions for Test: N/A

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_05	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When the user selects a directory, it is possible that there are no files with the correct extensions in that directory. In this case, BallotFileChooser should return an empty array (as opposed to 'null', which should only be returned when the selection process is aborted). This test case covers this scenario.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "BallotFileChooser.java". This test makes use of the "ballot_file_chooser_test_dir" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The "ballot_file_chooser_test_dir" directory exists and contains the following: "dummy.txt", "test_1.csv", "test_2.csv", "test_3.csv", "test_4.csv", "test_5.csv", and the subdirectories "test_sub_dir" and "test_empty_sub_dir". "test_sub_dir" should contain "dummy.txt", "test_7.csv", "test_8.csv", "test_9.csv", and "test_10.csv". "test_empty_sub_dir" should contain "dummy.txt". No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the 'test_select_empty_dir()' unit test. This is a partially automated test.				Running all of the unit tests may not run the test cases in order. Just follow the prompts given to you in any case.
2	Navigate to the "ballot_file_chooser_test_dir" directory.				This step must be done manually.
3	Select the "test_empty_sub_dir" directory.	This involves the "test_empty_sub_dir" directory and a "dummy.txt" which is contained within.			This step must be done manually.

4	Check that an empty array is returned.		An empty array should be returned.	An empty array was returned.	

Postconditions for Test: An empty array was returned.

Straightforward Voting System - Testing Document — Team #2

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_06	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When the user selects a directory, all of the files with valid extensions should be returned. However, no files within subdirectories should be returned, even if they are valid ballot files. This test checks that all of the correct files are returned from a directory and nothing else.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “BallotFileChooser.java”. The test method is ‘test_select_dir()’. This test makes use of the “ballot_file_chooser_test_dir” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The “ballot_file_chooser_test_dir” directory exists and contains the following: “dummy.txt”, “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, “test_5.csv”, and the subdirectories “test_sub_dir” and “test_empty_sub_dir”. “test_sub_dir” should contain “dummy.txt”, “test_7.csv”, “test_8.csv”, “test_9.csv”, and “test_10.csv”. “test_empty_sub_dir” should contain “dummy.txt”. No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the ‘test_select_dir()’ unit test. This is a partially automated test.				Running all of the unit tests may not run the test cases in order. Just follow the prompts given to you in any case.
2	Navigate to the parent directory of the “ballot_file_chooser_test_dir” directory.				This step must be done manually.
3	Select the “ballot_file_chooser_test_dir” directory.	This involves the “ballot_file_chooser_test_dir” directory and all of its contents (see the preconditions).			This step must be done manually.

4	Check that an array of 5 files is returned and that this array contains “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, and “test_5.csv”.	This involves “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, and “test_5.csv”	An array with the 5 expected files is returned.	An array with the 5 expected files was returned.	

Postconditions for Test: 5 files were returned: “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, and “test_5.csv”.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_07	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When the user is selecting files, they should be able to select a single file. This test examines that scenario.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “BallotFileChooser.java”. The test method is ‘test_select_single_file()’. This test makes use of the “ballot_file_chooser_test_dir” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The “ballot_file_chooser_test_dir” directory exists and contains the following: “dummy.txt”, “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, “test_5.csv”, and the subdirectories “test_sub_dir” and “test_empty_sub_dir”. “test_sub_dir” should contain “dummy.txt”, “test_7.csv”, “test_8.csv”, “test_9.csv”, and “test_10.csv”. “test_empty_sub_dir” should contain “dummy.txt”. No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the ‘test_select_single_file()’ unit test. This is a partially automated test.				Running all of the unit tests may not run the test cases in order. Just follow the prompts given to you in any case.
2	Navigate to the “ballot_file_chooser_test_dir” directory.				This step must be done manually.
3	Select the "test_1.csv" file.	This involves the "test_1.csv" file.			This step must be done manually.
4	Check that an array with 1 element, "test_1.csv", is returned.		An array with the one expected file is returned.	The array was returned with the one expected	

				file.	

Postconditions for Test: An array with 1 element, "test_1.csv", is returned.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_08	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When the user is selecting files, they should be able to select multiple, arbitrary files. This test examines that scenario.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “BallotFileChooser.java”. The test method is ‘test_select_some_files()’. This test makes use of the “ballot_file_chooser_test_dir” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The “ballot_file_chooser_test_dir” directory exists and contains the following: “dummy.txt”, “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, “test_5.csv”, and the subdirectories “test_sub_dir” and “test_empty_sub_dir”. “test_sub_dir” should contain “dummy.txt”, “test_7.csv”, “test_8.csv”, “test_9.csv”, and “test_10.csv”. “test_empty_sub_dir” should contain “dummy.txt”. No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the ‘test_select_some_files()’ unit test. This is a partially automated test.				Running all of the unit tests may not run the test cases in order. Just follow the prompts given to you in any case.
2	Navigate to the “ballot_file_chooser_test_dir” directory.				This step must be done manually.
3	Select the "test_1.csv", “test_3.csv”, and “test_4.csv” files.	This involves the "test_1.csv", “test_3.csv”, and “test_4.csv” files.			This step must be done manually.

4	Check that an array with 3 elements, "test_1.csv", “test_3.csv”, and “test_4.csv”, is returned.		An array with the three expected files is returned.	The array was returned with the three expected files.	

Postconditions for Test: An array with 3 elements, "test_1.csv", “test_3.csv”, and “test_4.csv”, is returned.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 5/4/2020
Test Case ID#: ballot_file_chooser_test_09	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When the user is selecting files, they should be able to every single file in their directory. This test examines that scenario.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “BallotFileChooser.java”. The test method is ‘test_select_all_files()’. This test makes use of the “ballot_file_chooser_test_dir” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): No	
Results (pass or fail): Pass	

Preconditions for Test: The “ballot_file_chooser_test_dir” directory exists and contains the following: “dummy.txt”, “test_1.csv”, “test_2.csv”, “test_3.csv”, “test_4.csv”, “test_5.csv”, and the subdirectories “test_sub_dir” and “test_empty_sub_dir”. “test_sub_dir” should contain “dummy.txt”, “test_7.csv”, “test_8.csv”, “test_9.csv”, and “test_10.csv”. “test_empty_sub_dir” should contain “dummy.txt”. No other files or directories should exist except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Run the ‘test_select_all_files()’ unit test. This is a partially automated test.				Running all of the unit tests may not run the test cases in order. Just follow the prompts given to you in any case.
2	Navigate to the “ballot_file_chooser_test_dir” directory.				This step must be done manually.
3	Select the "test_1.csv", “test_2.csv”, “test_3.csv”, “test_4.csv”, and “test_5.csv” files. This should be all of the files in the directory.	This involves the "test_1.csv", “test_2.csv”, “test_3.csv”, “test_4.csv”, and			This step must be done manually.

		“test_5.csv” files.			
4	Check that an array with 5 elements, "test_1.csv", “test_2”, “test_3.csv”, “test_4”, and “test_5.csv”, is returned.		An array with the five expected files is returned.	The array was returned with the five expected files.	

Postconditions for Test: An array with 5 elements, "test_1.csv", “test_2”, “test_3.csv”, “test_4”, and “test_5.csv”, is returned.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: ballot_test_01	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When a ballot is created, its ID should be the same as the ID provided to it when it was made. This test creates ballots with random IDs and calls get_id() to see that it returns the same ID.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in “Ballot.java”. The class with this test, “test_get_id()”, is in “BallotTest.java”
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Generate a random number.				
2	Create a ballot with ‘null’ for rankings and the random number from step 1 for the ID.				
3	Call ‘get_id()’.		The number returned is the same as the number from step 1.	The number returned is the same as the number from step 1.	
4	Repeat steps 1-3 9 more times.				

Postconditions for Test: The ID returned was the same as the ID given for construction for each ballot.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: ballot_test_02	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When a ballot is created, its current rank attribute should initially be 0. When 'current_rank()' is called immediately after this, 0 should be returned.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in "Ballot.java". The class with this test, "test_starting_rank()", is in "BallotTest.java"
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a ballot with 'null' for rankings and 101 for its ID.				
2	Call 'current_rank()'.		0 should be returned.	0 was returned.	

Postconditions for Test: N/A

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: ballot_test_03	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: When 'increment_rank()' is called, it should increase the current rank attribute for a ballot and return that value. It should only increment by one at a time, and it should return -1 if the rank is incremented more times than there are rankings.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in "Ballot.java". The class with this test, "test_ending_rank()", is in "BallotTest.java"
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a ballot with an array for 5 candidates (an array of 5 nulls) for its rankings and 101 for its ID.				
2	Increment the rank 4 times and check that each time, an incremented value is returned (1, 2, 3, and finally 4).		1, 2, 3, and finally 4 are returned, respectively.	1, 2, 3, and 4 were returned.	
3	Increment the rank one more time. -1 should be returned since we have an array for 5 rankings.		-1 is returned.	-1 was returned.	
4	Increment the rank two more times. -1 should be returned each time.		-1 is returned each time.	-1 was returned each time.	

Postconditions for Test: Incrementing the rank will return -1.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: ballot_test_04	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: Incrementing a ballot more times than there are rankings should never advance a ballots current rank attribute to a number greater than the number of rankings it has. Even when increment rank is called multiple times afterwards, 'current_rank()' should still return the greatest index for the rankings, but never more than that.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in "Ballot.java". The class with this test, "test_ending_rank()", is in "BallotTest.java"
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a ballot with an array for 5 candidates (an array of 5 nulls) for its rankings and 101 for its ID.				
2	Increment the rank 4 times.				
3	Call 'current_rank()' and check that the value returned is 4.		The value returned should be 4.	The value returned was 4.	
4	Increment the rank one more time.				
5	Call 'current_rank()' and check that the value returned is 4.		The value returned should still be 4.	The value returned was 4.	
6	Increment the rank three more times.				
7	Call 'current_rank()' and check that the value returned is 4.		The value returned should still be 4.	The value returned was 4.	

Postconditions for Test: Incrementing the rank and then checking the current rank for the ballot will be maxed out at 4.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: ballot_test_05	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: It is important that 'current_rank()' and 'increment_current_rank()' work in tandem. Getting the current rank should not increment it and incrementing the current rank should only increment the rank one at a time without ever exceeding the range of rankings. This test checks that these two methods work together and behave as expected.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in "Ballot.java". The class with this test, "test_rank_advancement()", is in "BallotTest.java"
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a ballot with an array for 5 candidates (an array of 5 nulls) for its rankings and 101 for its ID.				
2	Call 'current_rank()' three times. Check that it returns 0 each time (the rank should not advance).		The value returned should be 0 each time.	The value returned was 0 each time.	
3	Call 'increment_rank()' once.				
4	Call 'current_rank()' once. Check that it returns 1.		The value returned should be 1.	The value returned was 1.	
5	Call 'current_rank()' three times. Check that it returns 1 each time (the rank should not advance).		The value returned should be 1 each time.	The value returned was 1 each time.	
6	Call 'increment_rank()' three times.				

7	Call ‘current_rank()’ once. Check that it returns 4.		The value returned should be 4.	The value returned was 4.	

Postconditions for Test: The current rank of the ballot is 4.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: ballot_test_06	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: This test checks to see that calling 'reset_current_rank()' sets the current rank of a ballot to zero, no matter what the current state of the rank is.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in "Ballot.java". The class with this test, "test_reset_current_rank()", is in "BallotTest.java"
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create a ballot with an array for 5 candidates (an array of 5 nulls) for its rankings and 101 for its ID.				
2	Call 'reset_current_rank()'.				
3	Call 'current_rank()' once. Check that it returns 0.		The value returned is 0.	The value returned was 0.	
4	Call 'increment_rank()' 4 times.				
5	Call 'reset_current_rank()'.				
6	Call 'current_rank()' once. Check that it returns 0.		The value returned is 0.	The value returned was 0.	
7	Call 'increment_rank()' 6 times.				
8	Call 'reset_current_rank()'.				

9	Call ‘current_rank()’ once. Check that it returns 0.		The value returned is 0.	The value returned was 0.	

Postconditions for Test: The current rank of the ballot is 0.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: ballot_test_07	**Name(s) of Testers:** Nicholas Lovdahl
Test Description: Ballots need to be able to provide a reference to the candidate currently chosen	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in “Ballot.java”. The class with this test, “test_current_choice()”, is in “BallotTest.java”. This test also makes use of the constructor for the Candidate class in “Candidate.java”.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The constructor for the Candidate class works and does not produce errors.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create an array of 5 distinct candidates. The candidates can be named with letters A-E and IDs 0-4.				
2	Create a ballot with the array of candidates for rankings and 101 for the ID.				
3	Call ‘current_choice()’ twice. Check that the candidate returned is the same (not just a copy) as the first candidate created.		The candidate returned should be the first candidate each time.	The first candidate was returned each time.	
4	Call ‘increment_rank()’ once.				
5	Call ‘current_choice()’ once. Check that the candidate returned is the same as the second candidate.		The candidate returned should be the second candidate.	The second candidate was returned.	

6	Call ‘increment_rank()’ once.				
7	Call ‘current_choice()’ once. Check that the candidate returned is the same as the third candidate.		The candidate returned should be the third candidate.	The third candidate was returned.	
8	Call ‘increment_rank()’ once.				
9	Call ‘current_choice()’ once. Check that the candidate returned is the same as the fourth candidate.		The candidate returned should be the fourth candidate.	The fourth candidate was returned.	
10	Call ‘increment_rank()’ once.				
11	Call ‘current_choice()’ twice. Check that the candidate returned is the same as the fifth candidate created.		The candidate returned should be the fifth candidate each time.	The fifth candidate was returned each time.	

Postconditions for Test: The candidate returned by calling ‘current_choice()’ will be the fifth candidate.

Straightforward Voting System - Testing Document Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Candidate_Test_01	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the name of the candidates are sorted in the candidate list after the application reads the input file. It specifically compares the given string with the corresponding name of each candidate.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "CandidateTest.java" with the method test_get_name() calling method getname() form Candidate.java. This test generated data with predefined names (i.e. A,B,C,D,E) and ballot id's (i.e. 0,1,2,3,4).
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Candidates from random inputs with names A,B,C,D and E.				Candidates should be populated.
3	Count invalidated ballots from input file.		A	A	
4	Count invalidated ballots from input file.		B	C	
5	Count invalidated ballots from input file.		C	C	
6	Count invalidated ballots from input file.		D	D	
7	Count invalidated ballots from input file.		E	E	

Postconditions for Test: The candidate's name returned was the same as the candidate's name given for each candidate as the input value.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Candidate_Test_02	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the number of ballots (generated with randomly generated ID) for each candidate received after performing a series of operations over a specific set of iterations. It specifically verfies get_num_ballots() method if it correctly provides the expected number of ballots after adding, removing and resetting the number of ballots.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “CandidateTest.java” with the method test_get_num_ballots() by calling method get_num_ballots() form Candidate.java. This test generated data with predefined names as ‘’null” and randomly generated ballot id’s.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Candidates file with given names and input names and id’s..				Ballots should be populated.
4	Add 0 ballots and call get_num_ballots()		0	0	
5	Add 10 ballots and call get_num_ballots()		10	10	
6	Add 10 ballots and then remove 5 ballos. Then call call get_num_ballots()		5	5	
7	Reset ballots and call get_num_ballots()		0	0	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Candidate_Test_03	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the number of ballots (generated with randomly generated ID) for each candidate received after adding ballots for each candidate. It specifically verfies add_ballots() method if it correctly provides the expected number of ballots after adding a specific number of ballots.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “CandidateTest.java” with the method test_add_ballots() by calling method add_ballots() form Candidate.java. This test generated data with predefined names (i.e. A,B,C,D) and ballot id’s (i.e. 0,1,2,3).
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Candidates from random id’s with names A,B,C,D.				Ballots should be populated.
2	Add 10 ballots in Candidate A.		10	10	
4	Add 15 ballots in Candidate B.		15	15	
5	Add 20 ballots in Candidate C.		20	20	
6	Add 25 ballots in Candidate D.		25	25	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Candidate_Test_04	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the number of ballots (generated with randomly generated ID) for each candidate received after adding ballots for each candidate. It specifically verfies pull_ballots() method if it correctly provides the expected number of ballots after adding a specific number of ballots.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “CandidateTest.java” with the method test_pull_ballots() by calling method pull_ballots() form Candidate.java. This test generated data with predefined names (i.e. A,B) and ballot id’s (i.e. 0,1).
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Candidates from random id’s with names A,B				Ballots should be populated.
2	Call pull_ballots() ballots from candidate A with 0 ballots		0	0	
3	Add 10 ballots in candidate A and call pull_ballot() 10 times.		0	0	
4	Concurrently add and remove ballots (calling pull_ballots() 10 times).		0	0	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Candidate_Test_05	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the number of ballots (generated with randomly generated ID) for each candidate remaining after resetting the number of ballots to 0. It specifically verfies reset_ballots() method if it correctly provides the expected number of ballots (i.e. 0) after resetting all the ballots to 0.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “CandidateTest.java” with the method test_reset_ballots() by calling method reset_ballots() form Candidate.java. This test generated data with predefined names (i.e. A,B,C,D) and ballot id’s (i.e. 0,1,2,3).
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Candidates from random id’s with names A,B,C,D.				Ballots should be populated.
3	Add 10 ballots to Candidate A and call reset_ballots()		0	0	
4	Add 15 ballots to Candidate B and call reset_ballots()		0	0	
5	Add 20 ballots to Candidate C and call reset_ballots()		0	0	
6	Add 25 ballots to Candidate D and call reset_ballots()		0	0	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Candidate_Test_06	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the specific ballot (generated with randomly generated ID) after adding a given number of ballots for a candidate . It specifically verfies get_ballots(int id) method if it correctly provides the expected ballot for a given ballot_id.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “CandidateTest.java” with the method test_get_ballots(int i) by calling method get_ballots(int i) from Candidate.java. This test generated data with predefined names as “null” and ballot id’s (i.e. 0,1,2 etc.).
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Candidates from with name A and id 0				Ballots should be populated.
2	Initialize ballot with random name and id based on iteration				
3	Add ballot initialized from Step 2 to Candidate A				
4	Call get_ballot(int i) for given iteration i on candidate A				
5	Compare the given iteration value with fetched ballot’s id		True	True	
6	Perform Step 2-6 10 times.				

Postconditions for Test: The ID returned was the same as the ID given for construction for each ballot.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Candidate_Test_07	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the difference in the number of ballots between two candidates (e.g. A and B) after adding a given number of ballots for a candidate. It specifically verfies compareTo() method if it correctly provides the expected difference of ballot counts among two candidates with the given input.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "CandidateTest.java" with the method test_compareTo() by calling method compareTo()from Candidate.java. This test generated data with predefined ballot names as "null" and randomly generated ballot id's.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Candidates A and B				Ballots should be populated.
2	Add 10 Ballots to candidate A iteratively				
3	Add 15 Ballots to candidate B iteratively				
4	Call compareTo() method for difference between Candidate B and Candidate A		5	5	
5	Call compareTo() method for difference between Candidate A and Candidate B		-5	-5	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020

Straightforward Voting System - Testing Document **Team #2**

Test Case ID#: Candidate_Test_08	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks if unique id's can be added to candidate objects along with names and other attributes. The test specifically verfies get_id() method if it correctly provides the expected ballot extracted from the given candidate with the given input.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "CandidateTest.java" with the method test_get_id() by calling method get_id() from Candidate.java. This test generated data with predefined ballot names as "null" and randomly generated ballot id's.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: .N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize given number of candidates list and a random number generator				Ballots should be populated.
2	Call a random number generator for fetching a unique value.				
3	Initialize candidate with name as "null" and already generated random number from Step 2 as candidate id.				
4	Compare the current candidate id with randomly generated id in Step 2.		True	True	
5	Repeat Step 2-4 10 times.				

Postconditions for Test: The ID returned was the same as the ID given for construction for each ballot.

Straightforward Voting System - Testing Document Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Candidate_Test_09	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks if two candidates with the same name can be distinguished on the basis of their unique id's. The test specifically verfies get_id() method if it correctly provides the expected ballot extracted from the given candidate along with the same name and with the given input.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "CandidateTest.java" with the method test_get_id_same() by calling method get_id() from Candidate.java. This test generated data with a predefined candidate name as "A" with ballot id 0..
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: N/A

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize candidate with name A with unique id 0.				Ballots should be populated.
2	Initialize another candidate with name A with unique id 1.				
3	Compare if ID's of two candidates and check if they are not equal.		False	False	

Postconditions for Test: The ID returned was not the same as the ID given for construction for each ballot.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: Election_controller_01	**Name(s) of Testers:** Arun Sharma
Test Description: This test is responsible for getting ballots distributed among the candidates and updating the correct number of additions inside a specific candidate. This test specifically checks the add_ballots() method by adding single ballots iteratively inside the candidates and checks the ballot size for each candidate with the given input.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in “ElectionController.java”. The class with this test, “distributeBallot()”, is in “ElectionControllerTest.java”. This test makes use of the testing directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv”, “testinvalid10.csv” etc. The test case uses “test.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize ballots and candidates.	test.csv			Ballots and Candidates should be populated.
2	Add 1st ballot to a dummy candidate and compare with Integer 1 (AssertEqual)	test.csv	Candidate Size = 1	Candidate Size = 1	
3	Add 2nd t ballot to a dummy candidate and compare with Integer 2 (AssertEqual)	test.csv	Candidate Size = 2	Candidate Size = 2	
4	Add 3rd ballot to a dummy candidate and compare with Integer 3 (AssertEqual)	test.csv	Candidate Size = 3	Candidate Size = 3	
5	Add 4th ballot to a dummy candidate and compare with Integer 4 (AssertEqual)	test.csv	Candidate Size = 4	Candidate Size = 4	

Postconditions for Test: The Arraylist will return the same number of ballots assigned to the candidates as the expected in the output.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: Election_controller_02	**Name(s) of Testers:** Arun Sharma
Test Description: This test is responsible for getting ballots redistributed among the candidates and updating the correct number of ballots additions and deletion inside a specific candidate. This test specifically checks add_ballots() and pull_ballots() methods by either just adding single ballots iteratively or adding and removing inside given candidates.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in “ElectionController.java”. The class with this test, “redistributeBallot()”, is in “ElectionControllerTest.java”. This test makes use of the testing directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv”, “testinvalid10.csv” etc. The test case uses “test.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize ballots and candidates.	test.csv			Ballots and Candidates should be populated.
2	Add 1st ballot to a candidate	test.csv	1	1	
3	Add 2nd ballot and remove from the candidate	test.csv	1	1	
4	Add 3rd ballot to the candidate	test.csv	2	2	
5	Add 4th ballot and remove it from the candidate	test.csv	2	2	

Postconditions for Test: The Arraylist will return the same number of ballots assigned to the candidates as the expected in the output.

Straightforward Voting System - Testing Document Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: Election_controller_03	**Name(s) of Testers:** Arun Sharma
Test Description: This test is responsible for recording every operation when ballots are distributed or redistributed among candidates. The test should output the correct number of messages recorded.This test specifically checks customAuditFileMsg() and append recorded operation in the arraylist iteratively which is sent for writing them to disk.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in “ElectionController.java”. The class with this test, “customAuditFileMsg()”, is in “ElectionControllerTest.java”. This test makes use of the testing directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv”, “testinvalid10.csv” etc. The test case uses “test.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize ballots and candidates.	test.csv			Ballots and Candidates should be populated.
2	Add 1st message and check the message size	test.csv	1	1	
3	Add 2nd message and check the message size	test.csv	2	2	
4	Add 3rd message and check the message size	test.csv	3	3	
5	Add 4th message and check the message size	test.csv	4	4	

Postconditions for Test: The Arraylist will return the same set of length as the expected output.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: Ballot_Factory_01	**Name(s) of Testers:** Arun Sharma
Test Description: This test is responsible for initializing the ballots and storing the total number of ballots inside the arraylist. The array list should have the correct number of entries, correct ID and other attributes of the given ballots. This test specifically uses methods from ballots and candidate classes such as get_id(), current_choice()	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in “BallotFactory.java”. The class with this test, “initBallots()”, is in “BallotFactoryTest.java”. This test makes use of the testing directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv”, “testinvalid10.csv” etc. The test case uses “test.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize ballots and candidates and populate the arraylist with total number of ballots from the input file.	test.csv			Ballots and Candidates should be populated.
2	Compare the total number of ballots with input value	test.csv	30 (true)	30 (true)	
3	Compare IDs of the 1st ballots with given input value.	test.csv	0 (true)	0 (true)	
4	Compare names 1st current choice candidate with the given input value.	test.csv	A (true)	A (true)	

Postconditions for Test: The arraylist will return the same set of input as expected providing correctness of the method.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Ballot_Factory_02	**Name(s) of Testers:** Arun Sharma
Test Description: The test case checks the shuffling of the ballots in order to have fair elections. This operation must be done before running any election for avoiding any bias in the results. The test case specifically checks if the shuffling is performed by comparing the ballot id of the first ballot without any shuffle with the first ballot when the shuffling is turned off. The main intuition is both the ballots residing in first place must not have the same ballot id which proves that the shuffling operation is performed successfully. The test case uses the get_id() method for extracting id from the ArrayList of ballots generated from the input file.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "BallotFactoryTest.java" with the method shuffleBallots() calling Initcandidate() and InitBallot() methods from BallotFactory.java. This test makes use of the "testing" directory and its contents. The exact contents to be used are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv" etc. The test case uses "test.csv". No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Turn shuffle off.	test.csv			
2	Initialize Ballots and Candidates from given input csv file.				Ballots and Candidates should be populated.
3	Turn shuffle on.				
4	Initialize Ballots and Candidates from given input csv file.				
5	Compare the first ballot id of the initialized candidate list while turning shuffling on and off respectively.	test.csv	False	False	

Postconditions for Test: The ID returned was not the same as the ID given for each ballot. As a result the boolean value return does not match with the given boolean value as the input.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 4/2/2020
Test Case ID#: Ballot_Factory_03	**Name(s) of Testers:** Arun Sharma
Test Description: This test is responsible for initializing the candidates and verifying the contents stored inside each candidate. The contents used for the testing are names and ids for each candidate where each test case must assert true while comparing them to the given input .This test specifically uses methods from candidate classe such as get_name() and get_id() for extracting names and their unique ids respectively.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is in “ElectionController.java”. The class with this test, “initCandidates()”, is in “BallotFactoryTest.java”. This test makes use of the “testing” directory and its contents. The exact contents to be used are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv”, “testinvalid10.csv” etc. The test case uses “test.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize ballots and candidates.	test.csv			Ballots and Candidates should be populated.
2	Check the total candidate length	test.csv	6 (true)	6 (true)	
3	Compare the 1st candidate name with the given input	test.csv	A (true)	A (true)	
4	Compare the 1st candidate ID with the given input	test.csv	0 (true)	0 (true)	
5	Compare the 2nd candidate name with the given input	test.csv	B (true)	B (true)	
6	Compare the 2nd candidate ID with the given input.	test.csv	1 (true)	1 (true)	
7	Compare the 3rd candidate name with the given input.	test.csv	C (true)	C (true)	

8	Compare the 3rd candidate ID with the given input.	test.csv	2 (true)	2 (true)	
9	Compare the 4th candidate name with the given input.	test.csv	D (true)	D (true)	
10	Compare the 4th candidate ID with the given input.	test.csv	3 (true)	3 (true)	
11	Compare the 5th candidate name with the given input.	test.csv	E (true)	E (true)	
12	Compare the 5th candidate ID with the given input.	test.csv	4 (true)	4 (true)	

Postconditions for Test: The arraylist will return the same set of input as expected.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Invalidate_Ballot_Test_01	**Name(s) of Testers:** Arun Sharma
Test Description: This test checks the contents within the data structure used for storing total ballots form the given input file by randomly verifying their respective id with given sets of id's.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "InvalidationTest.java" with the method invalidate_random_ballots () and calling STVDroop Controller.java. This test makes use of the "testing" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv" etc. The test case uses testvalid.csv as the input file containing random invalidated ballots. No other files or directories should be used except for those described here.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Ballots from the input csv file.				Ballots should be populated in the data structure.
2	Extract 1st, 7th, 11th, 15th and 23rd ballot.				
3	Call invalidate_ballots() from STVDroopController.java				
4	Extract ballots from invalidate_ballots() and check their ballot ids are equal to the given ballots id extracted in Step 2	testinvalid.csv	1, 7, 11, 15, 23	1, 7, 11, 15, 23	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Invalidate_Ballot_Test_02	**Name(s) of Testers:** Arun Sharma
Test Description: In order to check the completeness, this test checks the overall contents for invalidated ballots stored within a user defined data structure containing total number ballots form the given input file. This test checks invalidated_ballots() methods on different test files containing 5, 10, 15, 20 invalidated files.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "InvalidationTest.java" with the method invalidate_ballots() and calling STVDroop Controller.java. This test makes use of the "testing" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv" etc. The test case uses "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv", "testinvalid15.csv", "testinvalid20.csv". No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Ballots from the input csv file.				Ballots should be populated.
2	Call invalidated_ballots () method				
3	Count invalidated ballots from input file.	test.csv	0	0	
4	Count invalidated ballots from input file.	testinvalid5.csv	5	5	
5	Count invalidated ballots from input file.	testinvalid10.csv	10	10	
6	Count invalidated ballots from input file.	testinvalid15.csv	15	15	
7	Count invalidated ballots from input file.	testinvalid20.csv	20	20	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Invalidate_Ballot_Test_03	**Name(s) of Testers:** Arun Sharma
Test Description: In order to check the completeness, this test checks the overall contents for invalidated and validated ballots stored within a user defined data structure containing total number ballots form the given input file. This test checks invalidated_ballots() methods on different test files containing 5, 10, 15, 20 invalidated files and 25, 20, 15, 10 validated ballots respectively. The main idea is to check if adding both validated and invalidated ballots must be equal to the total number of ballots.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "InvalidationTest.java" with the method invalidate_validate_total() and calling STVDroop Controller.java. This test makes use of the "testing" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv" etc. The test case uses "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv", "testinvalid15.csv", "testinvalid20.csv". No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Ballots from the input csv files.				Ballots should be populated.
2	Call invalidated_ballots ()				
3	Count total validate and invalidate ballots	test.csv	30	30	
4	Count total validate and invalidate ballots	testinvalid5.csv	30	30	
5	Count total validate and invalidate ballots	testinvalid10.csv	30	30	
6	Count total validate and invalidate ballots	testinvalid15.csv	30	30	
7	Count total validate and invalidate ballots	testinvalid20.csv	30	30	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Invalidate_Ballot_Test_04	**Name(s) of Testers:** Arun Sharma
Test Description: This test checks the contents of the invalidation report created after executing invalidation_ballots() method. The test specifically verifies the contents in the invalidation_ballots.txt generated after running STVDroop elections using invalidate_report() method by checking if it correctly provides the expected output string with respect to the given input string.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "InvalidationTest.java" and method invalidated_report_contents() and calling STVDroop Controller.java. This test uses the "testing" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv" etc. The test case uses "testinvalid.csv". No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize ballots and run STVDroop Election.	testinvalid.csv			Ballots should be populated.
2	Read invalidated_ballots.txt				
3	Compare 1st ballot with given input string.	invalidated_ballots.txt	1.B,D	1.B,D	
4	Compare 2nd ballot with given input string.	invalidated_ballots.txt	2.D,B	2.D,B	
5	Compare 3rd ballot with given input string.	invalidated_ballots.txt	3.A,B	3.A,B	
6	Compare 4th ballot with given input string.	invalidated_ballots.txt	4.A,D	4.A,D	
7	Compare 5th ballot with given input string.	invalidated_ballots.txt	5.E,B	5.E,B	

Postconditions for Test: The output string returned was the same as the input string given for each candidate.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Invalidate_Ballot_Test_05	**Name(s) of Testers:** Arun Sharma
Test Description: This test checks if the invalidation report is created after running the STVDroop election. The test specifically checks if the invalidation_report() method is creating an invalidation_ballots.txt file in the disk successfully. The main idea of the test is to run STVDroop election and read the invalidation_ballots.txt file by checking if the file object exists on the disk.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "InvalidationTest.java" with the method invalidated_report_create() calling invalidated_ballots () method form STVDroop Controller.java. This test makes use of the "testing" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv" etc. The test case uses "test,csv. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize Ballots from the input file	test.csv			Ballots should be populated.
2	Run STV election				
3	Check if the invalidation report is created	invalidation _ballots.txt	True	True	

Postconditions for Test: The boolean output for checking created files is equal to given boolean input value.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit / System	**Test Date:** 04/25/2020
Test Case ID#: Pluraity_Test_01	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the scalability of the application when performing plurality election for 100000 ballots for 10 Candidates and focuses on correctness of the algorithm. It specifically checks the output produced after running PlurarityController class and for each candidate, it checks the number of ballots with the given input value. The main intuition behind is if we add all the ballots from every candidate, it should be equal to total number of ballots from the given input file.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "PluralityControllerTest.java" with the method testmultipleballots() calling PluralityController.java, BallotFactory.java and ElectionController.java. This test makes use of the "testing" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv", "testinvalid10.csv" etc. The test case uses "testpluralitystress.csv". No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize 10 Candidates from the input csv file containing 100000 ballots	testplurality stress.csv			Ballots and Candidates should be populated.
2	Run PluralityController via ElectionController				
3	Compare the count of the 1st candidate with given input value using get_num_ballots().	testplurality stress.csv	9897	9897	
4	Compare the count of the 2nd candidate with given input value using get_num_ballots().	testplurality stress.csv	9901	9901	
5	Compare the count of the 3rd candidate with given input value using get_num_ballots().	testplurality stress.csv	9931	9931	
6	Compare the count of the 4th candidate with the given input value using get_num_ballots().	testplurality stress.csv	9958	9958	

7	Compare the count of 5th candidate with given input value using get_num_ballots().	testplurality stress.csv	9964	9964	
8	Compare the count of the 6th candidate with given input value using get_num_ballots().	testplurality stress.csv	9981	9981	
9	Compare the count of the 7th candidate with given input value using get_num_ballots().	testplurality stress.csv	10013	10013	
10	Compare the count of the 8th candidate with given input value using get_num_ballots().	testplurality stress.csv	10044	10044	
11	Compare the count of the 9th candidate with given input value using get_num_ballots().	testplurality stress.csv	10072	10072	
12	Compare the count of the 10th candidate with given input value using get_num_ballots().	testplurality stress.csv	10239	10239	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Pluraity_Test_02	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the correctness of the algorithm by feeding a single candidate to the Plurality Election and checks if the algorithm produces any errors. The test specifically checks the output of Plurality Election of a single candidate by comparing the number of ballots using get_num_ballots() with the given input value.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “PluralityControllerTest.java” with the method testoneCandidate() calling PluralityController.java, BallotFactory.java and ElectionController.java. This test makes use of the “testing” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv” etc. The test case uses “testpluralityonecandidate.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize candidates from the input csv file.				Ballots and Candidates should be populated.
2	Run PluralityController via ElectionController				
3	Compare the number of ballot of the candidate with given input value	testpluralityonecandidate.csv	12	12	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Pluraity_Test_03	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the correctness of the algorithm for 10 candidates and runs Plurality Election with given number of seats. The test specifically checks the output of Plurality Election of a single candidate by comparing the number of ballots using get_num_ballots() with the given input value.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "PluralityControllerTest.java" with the method testtenCandidate() calling PluralityController.java, BallotFactory.java and ElectionController.java. This test makes use of the "testing" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv" etc. The test case uses "testplurality10.csv". No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize ballots and candidates from the input file.				Ballots and Candidates should be populated.
2	Run PluralityController via ElectionController				
3	Compare the count of the 1st candidate with given input value using get_num_ballots().	testplurality10.csv	1	1	
4	Compare the count of the 2nd candidate with given input value using get_num_ballots().	testplurality10.csv	1	1	
5	Compare the count of the 3rd candidate with given input value using get_num_ballots().	testplurality10.csv	1	1	
6	Compare the count of the 4th candidate with given input value using get_num_ballots().	testplurality10.csv	2	2	

7	Compare the count of the 5th candidate with given input value using get_num_ballots().	testplurality10.csv	3	3	
8	Compare the count of the 6th candidate with given input value using get_num_ballots().	testplurality10.csv	3	3	
9	Compare the count of the 7th candidate with given input value using get_num_ballots().	testplurality10.csv	4	4	
10	Compare the count of the 8th candidate with given input value using get_num_ballots().	testplurality10.csv	4	4	
11	Compare the count of the 9th candidate with given input value using get_num_ballots().	testplurality10.csv	5	5	
12	Compare the count of the 10th candidate with given input value using get_num_ballots().	testplurality10.csv	6	6	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Pluraity_Test_04	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the correctness of the algorithm for the candidates receiving unranked ballots. In this case, the plurality algorithm chooses random sets of winners and losers and displays candidates' names along with the given number of ballots (i.e. should be 0 in this case). Hence, the test specifically checks the output of Plurality Election for each candidate by comparing the number of ballots using get_num_ballots() with the given input value.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is "PluralityControllerTest.java" with the method testnorankCandidate() calling PluralityController.java, BallotFactory.java and ElectionController.java. This test makes use of the "testing" directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The "testing" directory exists and contains the test files such as "testinvalid.csv", "testinvalid5.csv" etc. The test case uses "ttestpluralitynorank.csv". No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize 5 candidates from the input file containing ballots with no rank.				Ballots and Candidates should be populated.
2	Run PluralityController via ElectionController				
3	Compare the count of the 1st candidate with given input value using get_num_ballots().	testpluralityno rank.csv	0	0	
4	Compare the count of the 2nd candidate with given input value using get_num_ballots().	testpluralityno rank.csv	0	0	
5	Compare the count of the 3rd candidate with given input value using get_num_ballots().	testpluralityno rank.csv	0	0	
6	Compare the count of the 4th candidate with given input value using get_num_ballots().	testpluralityno rank.csv	0	0	

7	Compare the count of the 5th candidate with given input value using get_num_ballots().	testpluralityno rank.csv	0	0	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document **Team #2**

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Pluraity_Test_05	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the correctness of the algorithm for the candidates having a two way tie i.e. if two candidates have the same number of votes.. In this case, the plurality algorithm chooses random sets of winners or losers and displays candidates' names along with the given number of ballots. Hence, the test specifically checks the output of Plurality Election for each candidate by comparing the number of ballots using get_num_ballots() with the given input value.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “PluralityControllerTest.java” with the method testtwowaytie() calling PluralityController.java, BallotFactory.java and ElectionController.java. This test makes use of the “testing” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv” etc. The test case uses “testpluralitytwoway.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize and ballots candidates from the input file.				Ballots and Candidates should be populated.
2	Run PluralityController via ElectionController				
3	Compare the count of the 1st candidate with given input value using get_num_ballots().	testpluralitytwoway.csv	1	1	
4	Compare the count of the 2nd candidate with given input value using get_num_ballots().	testpluralitytwoway.csv	3	3	
5	Compare the count of the 3rd candidate with given input value using get_num_ballots().	testpluralitytwoway.csv	6	6	
6	Compare the count of the 4th candidate with given input	testplurality	10	10	

	value using get_num_ballots().	twoway.csv			
7	Compare the count of the 5th candidate with given input value using get_num_ballots().	testplurality twoway.csv	10	10	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document

Team #2

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Pluraity_Test_06	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the correctness of the algorithm for the candidates having a four way tie i.e. if four candidates have the same number of votes.. In this case, the plurality algorithm chooses random sets of winners or losers and displays candidates' names along with the given number of ballots. Hence, the test specifically checks the output of Plurality Election for each candidate by comparing the number of ballots using get_num_ballots() with the given input value.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “PluralityControllerTest.java” with the method testfourwaytie() by calling PluralityController.java, BallotFactory.java and ElectionController.java. This test makes use of the “testing” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv” etc. The test case uses “testpluralityfourway.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize candidates and ballots from the given input csv file.				Ballots and Candidates should be populated.
2	Run PluralityController via ElectionController				
3	Compare the count of the 1st candidate with given input value using get_num_ballots().	testpluralityfourway.csv	2	2	
4	Compare the count of the 2nd candidate with given input value using get_num_ballots().	testpluralityfourway.csv	7	7	
5	Compare the count of the 3rd candidate with	testpluralityf	7	7	

	given input value using get_num_ballots().	ourway.csv			
6	Compare the count of the 4th candidate with given input value using get_num_ballots().	testpluralityfourway.csv	7	7	
7	Compare the count of the 5th candidate with given input value using get_num_ballots().	testpluralityfourway.csv	7	7	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.

Straightforward Voting System - Testing Document

Team #2

‘

Test Stage (Unit or System): Unit	**Test Date:** 04/25/2020
Test Case ID#: Pluraity_Test_07	**Name(s) of Testers:** Arun Sharma
Test Description: The test checks the correctness of the algorithm for the candidates having no ballots in their possession. In this case, the plurality algorithm chooses random sets of winners or losers and displays candidates' names along with the given number of ballots (i.e. will be 0 in this case). Hence, the test specifically checks the output of Plurality Election for each candidate by comparing the number of ballots using get_num_ballots() with the given input value.	**Indicate where you are storing the tests (what file) and the name of the method being used:** The class being tested is “PluralityControllerTest.java” with the method testnoballot() by calling PluralityController.java, BallotFactory.java and ElectionController.java. This test makes use of the “testing” directory and its contents. The exact contents that should be in this directory are listed in the preconditions.
Automated (yes or no): Yes	
Results (pass or fail): Pass	

Preconditions for Test: The “testing” directory exists and contains the test files such as “testinvalid.csv”, “testinvalid5.csv” etc. The test case uses “testpluralitynoballot.csv”. No other file should be used.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Initialize 5 candidates from the input file containing 0 ballots				Ballots and Candidates should be populated.
2	Run PluralityController via ElectionController				
3	Compare the count of the 1st candidate with given input value using get_num_ballots().	testpluralitynoballot.csv	0	0	
4	Compare the count of the 2nd candidate with given input value using get_num_ballots().	testpluralitynoballot.csv	0	0	

5	Compare the count of the 3rd candidate with given input value using get_num_ballots().	testpluralitynoballot.csv	0	0	
6	Compare the count of the 4th candidate with given input value using get_num_ballots().	testpluralitynoballot.csv	0	0	
7	Compare the count of the 5th candidate with given input value using get_num_ballots().	testpluralitynoballot.csv	0	0	

Postconditions for Test: The count returned was the same as the count given for each candidate as the input value.