Project Name Project 1: Voting System
Team# 5

Test Stage: Unit __ System X

Test Case ID#: IR_001

Name(s) of Testers: Trevor

Test Description:

IR test Description with a copy of the example file given on Canvas. Should print the winner to screen. Audit and media files should be created in /src directory.

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

The test file is “IR.txt” and it is stored in the /testing directory inside our repository.

Automated: yes___ no _X__

Results: Pass __X___ Fail_____

Preconditions for Test: The “IR.txt” exists and is correctly formatted.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the application	Open terminal. Navigate to the working directory. Use “java SunshineApp.java” to run the application	GUI should pop up on the user’s screen with a window to input the file name.	GUI popped up on the user’s screen with a window to input the file name.	
3	Enter the File containing election information	Where prompted by the GUI, enter “IR.txt” into the search bar.	The system should accept the file and run the election. A winner will be displayed to the GUI.	The election run was successful. Rosen won the election and his name and party were displayed on the GUI.	
4	Check for Media/Audit File	Open the /src directory and search for the audit and media files	They should both contain their relevant information.	Upon opening them, both files contained the needed information.	

Post condition(s) for Test:

The GUI displays text that tells some information about the election. It shows that the election was an IR election. It also shows the number of ballots cast and the number of candidates that ran during the election. Finally, the winner, their party, and their votes are displayed.

Audit and media files were created in the /src directory. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit __ System X **Test Date: 5/2/2021**

Test Case ID#: IR_002 **Name(s) of Testers: Trevor**

Test Description:

IR test Description with a CSV file that results in a 4 way tie after the initial distribution. I will run this file multiple times to ensure that a winner is chosen randomly due to the tiebreakers.

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

The test file is “IR4way.txt” and it is located in the /testing directory of our repository.

Automated: yes___ no _X__

Results: Pass __X___ Fail_____

Preconditions for Test: The “IR4waytie.txt” exists and is correctly formatted.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the application	Open terminal. Navigate to the working directory. Use “java SunshineApp.java” to run the application	GUI should pop up on the user’s screen with a window to input the file name.	GUI popped up on the user’s screen with a window to input the file name.	
3	Enter the File containing election information	Where prompted by the GUI, enter “IR4way.txt” into the search bar.	The system should accept the file and run the election. A random winner will be displayed to the GUI. This occurs because of the four way tie.	The election was run multiple times and a different winner was chosen on different runs. This implies that the winner chosen is random which is good.	
4	Check for Media/Audit File	Open the /src directory and search for the audit and media files	They should both contain their relevant information.	Upon opening them, both files contained the needed information.	

Post condition(s) for Test:

The GUI displays text that tells some information about the election. It shows that the election was an IR election. It also shows the number of ballots cast and the number of candidates that ran during the election. Finally, the winner, their party, and their votes are displayed.

Audit and media files were created in the /src directory. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit __ System X **Test Date: 5/2/2021**

Test Case ID#: IR_003 **Name(s) of Testers: Trevor**

Test Description:

IR test Description with a CSV file that results in a 2 way tie after the vote distribution. A winner should be chosen at random using the flipCoin() method.

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

The test file is “IR2waytie.txt” and it is located in the /testing directory of our repository.

Automated: yes___ no _X__

Results: Pass __X___ Fail_____

Preconditions for Test: The “IR2waytie.txt” exists and is correctly formatted.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the application	Open terminal. Navigate to the working directory. Use “java SunshineApp.java” to run the application	GUI should pop up on the user’s screen with a window to input the file name.	GUI popped up on the user’s screen with a window to input the file name.	
3	Enter the File containing election information	Where prompted by the GUI, enter “IR2waytie.txt” into the search bar.	The system should accept the file and run the election. A random winner will be displayed to the GUI. This occurs because of the four way tie.	The election was run multiple times and a different winner was chosen on different runs. This implies that the winner chosen is random which is good.	
4	Check for Media/Audit File	Open the /src directory and search for the audit and media files	They should both contain their relevant information.	Upon opening them, both files contained the needed information.	

Post condition(s) for Test:

The GUI displays text that tells some information about the election. It shows that the election was an IR election. It also shows the number of ballots cast and the number of candidates that ran during the election. Finally, the winner, their party, and their votes are displayed.

Audit and media files were created in the /src directory. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit __ System X **Test Date: 5/2/2021**

Test Case ID#: IR_004 **Name(s) of Testers: Trevor**

Test Description:

IR test Description with a CSV file that results in a ballot being deleted. The original ballot is owned by candidate 4 with the only alternate vote being placed for candidate 1. Candidate 1 is eliminated first. Then, candidate 4 is eliminated. This causes the ballot in question to be deleted since it votes for no more candidates.

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

The test file is “IReliminateballot.txt” and it is located in the /testing directory of our repository. The entire system is being tested.

Automated: yes___ no _X__

Results: Pass __X___ Fail_____

Preconditions for Test: The “IReliminateballot.txt” exists and is correctly formatted.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the application	Open terminal. Navigate to the working directory. Use “java SunshineApp.java” to run the application	GUI should pop up on the user’s screen with a window to input the file name.	GUI popped up on the user’s screen with a window to input the file name.	
3	Enter the File containing election information	Where prompted by the GUI, enter “IReliminateballot.txt” into the search bar.	The GUI will display information regarding the winner of the election. Upon checking the audit file we can see that the ballot was lost after candidate 4 is eliminated.	The GUI displayed the relevant information and the audit file did indicate that the ballot was deleted. This was a success.	
4	Check for Media/Audit File	Open the /src directory and search for the media and audit files that have been generated.	They should both contain their relevant information.	Upon opening them, both files contained the needed information.	

Post condition(s) for Test:

The GUI displays text that tells some information about the election. It shows that the election was an IR election. It also shows the number of ballots cast and the number of candidates that ran during the election. Finally, the winner, their party, and their votes are displayed.

Audit and media files were created in the /src directory. The system has not changed

Project Name: Project 1: Voting System Team# 5

Test Stage: Unit __ System X **Test Date: 5/2/2021**

Test Case ID#: OPL_001 **Name(s) of Testers: Trevor**

Test Description:

OPL election test with a CSV that was copied off the example on Canvas.

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

The test file is “OPL.txt” and it is located in the /testing directory of our repository.

Automated: yes___ no _X__

Results: Pass __X___ Fail_____

Preconditions for Test: The “OPL.txt” exists and is correctly formatted.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the application	Open terminal. Navigate to the working directory. Use “java SunshineApp.java” to run the application	GUI should pop up on the user’s screen with a window to input the file name.	GUI popped up on the user’s screen with a window to input the file name.	
3	Enter the File containing election information	Where prompted by the GUI, enter “OPL.txt” into the search bar.	The system should accept the file input and run the election. The winners of seats should be shown to the screen.	The election was successfully run. All winners were displayed to the screen.	
4	Check for Media/Audit File	Open the /src directory and search for the audit and media files.	They should both contain their relevant information.	Upon opening them, both files contained the needed information including intermediate test descriptions.	

Post condition(s) for Test:

The GUI displays the names and parties of all candidates that were awarded a seat. The media and audit files were created. The information in the media file matches the information that is displayed to the user via the GUI. The audit file contains information regarding intermediate steps including vote assignments and seat awards.

Both the audit and media files are created and stored in the /src directory. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit __ System X **Test Date: 5/2/2021**

Test Case ID#: OPL_002 **Name(s) of Testers: Trevor**

Test Description:

OPL election test in which a party wins more seats than it has candidates to assign them to. It needs to reassign seats to a different party.

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

The test file is “OPLTOOMANYSEATS.txt” and it is located in the /testing directory of our repository.

Automated: yes___ no _X__

Results: Pass __X___ Fail_____

Preconditions for Test: The “OPLTOOMANYSEATS.txt” exists and is correctly formatted.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the application	Open terminal. Navigate to the working directory. Use “java SunshineApp.java” to run the application	GUI should pop up on the user’s screen with a window to input the file name.	GUI popped up on the user’s screen with a window to input the file name.	
3	Enter the File containing election information	Where prompted by the GUI, enter “OPLTOOMANYSEATS.txt” into the search bar.	The system should accept the file input and run the election. The winners of seats should be shown to the screen.	The election was successfully run. All winners were displayed to the screen.	
4	Check for Media/Audit File	Open the /src directory and search for the audit and media files.	They should both contain their relevant information. The media file should show that a seat was temporarily awarded and then redistributed.	Upon opening them, both files contained the needed information including intermediate test descriptions.	

Post condition(s) for Test:

The GUI displays the names and parties of all candidates that were awarded a seat. The media and audit files were created. The information in the media file matches the information that is displayed to the user via the GUI. The audit file contains information regarding intermediate steps including vote assignments and seat awards.

Audit and media files are created and stored in the /src directory.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit __ System X **Test Date: 5/2/2021**

Test Case ID#: OPL_003 **Name(s) of Testers: Trevor**

Test Description:

OPL Election test in which a party is awarded a seat. However, the top 2 candidates in that party are tied so a tiebreaker must be used to determine who gets the seat. File will be passed in multiple times to ensure the tiebreaker is random.

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

The test file is “OPLTIE.txt” and it is located in the /testing directory of our repository.

Automated: yes___ no _X__

Results: Pass __X___ Fail_____

Preconditions for Test: The “OPLTIE.txt” exists and is correctly formatted.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the application	Open terminal. Navigate to the working directory. Use “java SunshineApp.java” to run the application	GUI should pop up on the user’s screen with a window to input the file name.	GUI popped up on the user’s screen with a window to input the file name.	
3	Enter the File containing election information	Where prompted by the GUI, enter “OPLTIE.txt” into the search bar.	The system should accept the file input and run the election. The winners of seats should be shown to the screen.	The election was successfully run. All winners were displayed to the screen. Multiple runs resulted in differing winners which is good.	
4	Check for Media/Audit File	Open the /src directory and search for the audit and media files.	They should both contain their relevant information. The audit file should indicate that a tie has occurred.	Upon opening them, both files contained the needed information including intermediate test descriptions. Audit file showed that a tie occurred.	

Post condition(s) for Test:

The GUI displays the names and parties of all candidates that were awarded a seat. The media and audit files were created. The information in the media file matches the information that is displayed to the user via the GUI. The audit file contains information regarding intermediate steps including vote assignments and seat awards.

Audit and Media files were created and placed in the /src directory.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit __ System X **Test Date: 5/2/2021**

Test Case ID#: OPL_004 **Name(s) of Testers: Trevor**

Test Description:

OPL Election in which there are more seats available than candidates running.

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

The test file is “OPLTOOMANYSEATS2.txt” and it is located in the /testing directory of our repository.

Automated: yes___ no _X__

Results: Pass __X___ Fail_____

Preconditions for Test: The “OPLTOOMANYSEATS2.txt” exists and is correctly formatted.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the application	Open terminal. Navigate to the working directory. Use “java SunshineApp.java” to run the application	GUI should pop up on the user’s screen with a window to input the file name.	GUI popped up on the user’s screen with a window to input the file name.	
3	Enter the File containing election information	Where prompted by the GUI, enter “OPLTOOMANYSEATS2.txt” into the search bar.	The system should accept the file input and run the election. The winners of seats should be shown to the screen. There will be an extra seat	The election was successfully run. All winners were displayed to the screen.	
4	Check for Media/Audit File	Open the /src directory and search for the audit and media files.	They should both contain their relevant information. The media file should show that a seat was temporarily awarded and then redistributed.	Upon opening them, both files contained the needed information including intermediate test descriptions.	

Post condition(s) for Test:

The GUI displays the names and parties of all candidates that were awarded a seat. The media and audit files were created. The information in the media file matches the information that is displayed to the user via the GUI. The audit file contains information regarding intermediate steps including vote assignments and seat awards.

Audit and media files are created and stored in the /src directory

Project Name: Project 1: Voting System Team# 5

Test Stage: Unit _X_ System __ **Test Date: 5/2/2021**

Test Case ID#: SUNSHINE_TEST_001 **Name(s) of Testers: Tom**

Test Description:

Unit test for the SunshineApp Class. The constructor is tested.

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

The test file is “TestSunshineApp.java” and it is located in the /src directory of our repository.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: The environment is configured correctly for JUnit testing. The “SunshineApp.java” and “TestSunshineApp.java” exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the test.	Junit should respond with a green “Passed” message	Junit responded that the unit test was passed.	

Post condition(s) for Test:

The Junit GUI has displayed a “Passed” message. The system has not changed.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit _X_ System __ **Test Date: 5/2/2021**

Test Case ID#: CANDIDATE_TEST_001 **Name(s) of Testers: Tom**

Test Description:

Unit test for the Candidate Class. The constructor is tested. Also tested are getName(), getParty(), getBallots(), getNumBallots(), and getParty() are tested.

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

The test file is “TestCandidate.java” and it is located in the /src directory of our repository.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: The environment is configured correctly for JUnit testing. The “Candidate.java” and “TestCandidate.java” exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit GUI has displayed a “Passed” message. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit _X_ System __ **Test Date: 5/2/2021**

Test Case ID#: CANDIDATE_TEST_002 **Name(s) of Testers: Tom**

Test Description:

Unit test for the Candidate Class. The addBallot function is tested. Also tested is getNumBallots().

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

The test file is “TestCandidate.java” and it is located in the /src directory of our repository.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: The environment is configured correctly for JUnit testing. The “Candidate.java” and “TestCandidate.java” exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework displayed a pass message. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit _X_ System __ **Test Date: 5/2/2021**

Test Case ID#: CANDIDATE_TEST_003 **Name(s) of Testers: Tom**

Test Description:

Unit test for the Candidate Class. The deleteBallot function is tested. Also tested is getNumBallots(), addBallot(), and getBallots().

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

The test file is “TestCandidate.java” and it is located in the /src directory of our repository.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: The environment is configured correctly for JUnit testing. The “Candidate.java” and “TestCandidate.java” exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit _X_ System __ **Test Date: 5/2/2021**

Test Case ID#: BALLOT_TEST_001 **Name(s) of Testers: Tom**

Test Description:

Unit test for the Ballot Class. The Ballot constructor is tested. Also tested is getChoices().

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

The test file is “TestBallot.java” and it is located in the /src directory of our repository.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: The environment is configured correctly for JUnit testing. The “Ballot.java” and “TestBallot.java” exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit _X_ System __ **Test Date: 5/2/2021**

Test Case ID#: BALLOT_TEST_002 **Name(s) of Testers: Tom**

Test Description:

Unit test for the Ballot Class. The getChoices() function is tested.

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

The test file is “TestBallot.java” and it is located in the /src directory of our repository.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: The environment is configured correctly for JUnit testing. The “Ballot.java” and “TestBallot.java” exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit X_ System _ **Test Date: 5/2/2021**

Test Case ID#:
OPL_TEST_PROCESS_FILE_001 **Name(s) of Testers: Cole**

Test Description:

Tests that an election is properly processed for the opl election. Checks that it properly gets the number of seats, ballots, candidates, and that it properly creates and assigns candidates to their parties.

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

The testProcessFile is stored in the TestOplElection class in src, and uses methods from OplElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit properly compiles and TestOplElection.java, OplElection.java, Party.java, and Candidate.java exist. The “OPL.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OplElection object	OPL.txt	myElection is not null	myElection is not null	
2	Call processFile() method	OPL.txt	processFile runs with no errors	processFile runs with no errors	
3	Check that the file was processed correctly	OPL.txt	processFile correctly assigns the number of balltos, seats, candidates, etc.	processFile correctly assigns the number of balltos, seats, candidates, etc.	

Post condition(s) for Test:

The correct number of seats, ballots, candidates are obtained, as well as creating the proper Candidate objects and adding them to their correct party.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
OPL_TEST_CONSTRUCTOR_001 **Name(s) of Testers: Cole**

Test Description:

Tests that the constructor properly creates an OplElection object and assigns the class variables properly.

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

The testAssignBallots method is stored in the TestOplElection class in src, and uses methods from OplElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestOplElection.java, OplElection.java, Candidate.java, and Ballot.java exist. The “OPL.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OplElection Object	OPL.txt	myElection is not null	myElection is not null	
2	Check that file assigns class variables	OPL.txt	The file is properly read and assigns the right values	The file is properly read and assigns the right values	

Post condition(s) for Test:

Creates the candidates properly and is able to correctly assign ballots to those candidates.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
OPL_TEST_ASSIGN_BALLOTS_001 **Name(s) of Testers: Cole**

Test Description:

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

The testConstructor method is stored in the TestOplElection class in src, and uses the constructor method from OplElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestOplElection.java and OplElection.java exist. The “OPL.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OplElection object	OPL.txt	myElection is not null	myElection is not null	
2	Call processFile() method	OPL.txt	The file is properly processed	The file is properly processed	
3	Check that candidates were created and ballots assigned	OPL.txt	The candidates were properly created and assigned the proper ballots	The candidates were properly created and assigned the proper ballots	

Post condition(s) for Test:

The proper candidates were created and assigned to the right party, with the correct ballots assigned to each candidate.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
OPL_TEST_RUN_ELECTION_001 **Name(s) of Testers: Cole**

Test Description:

Tests that the runElection method in OplElection runs properly and correctly runs the election.

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

The testRunElection method is stored in the TestOplElection class in src, and uses methods from OplElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestOplElection.java, OplElection.java, Party.java, and Candidate.java exist. The “OPL.txt” file is placed in the same directory as TestOplElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OplElection object	OPL.txt	myElection is not null	myElection is not null	
2	Call runElection() method	OPL.txt	runElection() runs and doesn't throw any errors	runElection() runs and doesn't throw any errors	
3	Check that correct winners are declared	OPL.txt	The correct winners are declared	The correct winners are declared	

Post condition(s) for Test:

The correct winners were declared from the election file.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
OPL_TEST_RUN_ELECTION_TIE_001 **Name(s) of Testers: Cole**

Test Description:

Runs an OPL election where two candidates tie within a party that was awarded a seat. A coin flip is used to determine which candidate is awarded the seat.

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

The testRunElectionTie method is stored in the TestOplElection class in src, and uses methods from OplElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestOplElection.java, OplElection.java, Party.java, and Candidate.java exist. The “OPLTIE.txt” file is placed in the same directory as TestOplElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OplElection object	OPLTie.txt	myElection is not null	myElection is not null	
2	Call runElection() method	OPLTie.txt	runElectionTie() runs and doesn't throw any errors	runElectionTie() runs and doesn't throw any errors	
3	Check that correct winners are declared	OPLTie.txt	The correct winners are declared	The correct winners are declared	

Post condition(s) for Test:

The correct winners were declared from the election file.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
OPL_TEST_TOOMANYSEATS_001 **Name(s) of Testers: Cole**

Test Description:

Runs an OPL election where a party is awarded more seats than they have candidates. The seats are reassigned to other parties.

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

The testRunElectionTooManySeats method is stored in the TestOplElection class in src, and uses methods from OplElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestOplElection.java, OplElection.java, Party.java, and Candidate.java exist. The “OPLTOOMANYSEATS.txt” file is placed in the same directory as TestOplElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OplElection object	OPLTOOMANYSEATS.txt	myElection is not null	myElection is not null	
2	Call runElection() method	OPLTOOMANYSEATS.txt	runElectionTooManySeats() runs and doesn't throw any errors	runElectionTooManySeats() runs and doesn't throw any errors	
3	Check that correct winners are declared	OPLTOOMANYSEATS.txt	The correct winners are declared	The correct winners are declared	

Post condition(s) for Test:

The correct winners were declared from the election file.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
OPL_TEST_TOOMANYSEATS2_001 **Name(s) of Testers: Cole**

Test Description:

Runs an OPL election where there are more available seats than candidates and every candidate is awarded a seat.

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

The testRunElectionTooManySeats2 method is stored in the TestOplElection class in src, and uses methods from OplElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestOplElection.java, OplElection.java, Party.java, and Candidate.java exist. The “OPLTOOMANYSEATS2.txt” file is placed in the same directory as TestOplElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create OplElection object	OPLTOOMANYSEATS2.txt	myElection is not null	myElection is not null	
2	Call runElection() method	OPLTOOMANYSEATS2.txt	runElectionTooManySeats() runs and doesn't throw any errors	runElectionTooManySeats() runs and doesn't throw any errors	
3	Check that correct winners are declared	OPLTOOMANYSEATS2.txt	The correct winners are declared	The correct winners are declared	

Post condition(s) for Test:

The correct winners were declared from the election file.

Project Name: Project 1: Voting System Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#: PARTY_TEST_001 **Name(s) of Testers: Cole**

Test Description:

Tests the constructor of the party class and that class variables are properly assigned.

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

The testConstructor method is stored in the TestParty class in src and uses methods from Party

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestParty.java, Party.java, exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create Party object	Char X	Party name X is created	Party name X is created	

Post condition(s) for Test:

The correct winners were declared from the election file.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#: PARTY_TEST_002 **Name(s) of Testers: Cole**

Test Description:

Tests that addCandidate() method works and that a candidate is properly added.

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

The testAddCandidate method is stored in the TestParty class in src and uses methods from Party

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestParty.java, Party.java, Candidate.java exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create Party object	char X	myParty is not null	myParty is not null	
2	call addCandidate	Candidate(John, X)	addCandidate properly adds the candidate to the ArrayList	addCandidate properly adds the candidate to the ArrayList	

Post condition(s) for Test:

The candidate was added to the ArrayList in Party.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#: PARTY_TEST_003 **Name(s) of Testers: Cole**

Test Description:

Tests that the rankCandidates method works properly, and orders the candidates in order based on their amount of votes.

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

The testRankCandidates method is stored in the TestParty class in src and uses methods from Party

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestParty.java, Party.java, Candidate.java exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create Party object	char X	myPartyn is not null	myParty is not null	
2	Create Candidate objects and add them to Party object	Candidate(John, X) Candidate(Jolly, X)	Candidates are properly added to the party	Candidates are properly added to the party	
3	Call rankCandidates method	party.candidates	The candidates are ranked in correct order	The candidates are ranked in the correct order.	

Post condition(s) for Test:

The candidates in the Party are ranked in the correct order according to their amount of ballots

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#: PARTY_TEST_004 **Name(s) of Testers: Cole**

Test Description:

Tests that the getter and setter for remainder works properly.

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

The testGetterSetterRemainder method is stored in the TestParty class in src and uses methods from Party

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestParty.java, Party.java exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create Party object	char X	myParty is not null	myParty is not null	
2	Call setRemainder and getRemainder	6	setRemainder and getRemainder properly work	setRemainder and getRemainder properly work	

Post condition(s) for Test:

setRemainder and getRemainder properly work.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#: PARTY_TEST_005 **Name(s) of Testers: Cole**

Test Description:

Tests that addSeats and addSeat methods work.

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

The testAddSeats method is stored in the TestParty class in src and uses methods from Party

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestParty.java, Party.java exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create Party object	char X	myParty is not null	myParty is not null	
2	Call addSeats() and addSeat()	6	addSeats() and addSeat() both add the correct number of seats	addSeats() and addSeat() both add the correct number of seats	

Post condition(s) for Test:

Both addSeats() and addSeat() add the correct number of seats to the total.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#: PARTY_TEST_006 **Name(s) of Testers: Cole**

Test Description:

Tests that assignSeats() method works and properly declares winners for the party.

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

The testAssignSeats method is stored in the TestParty class in src and uses methods from Party

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestParty.java, Party.java Candidate.java exist.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Create Party object	char X	myParty is not null	myParty is not null	
2	Create Candidate objects and add them to the party	Candidate(John, X) Candidate(Jolly, X)	The candidates are properly created and added to the party	The candidates are properly created and added to the party	
3	Call the assignSeats() method	party.ArrayList	The correct candidate is awarded a seat	The correct candidate is awarded a seat	

Post condition(s) for Test:

The correct candidate was awarded the seat in the party.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
IR_TEST_CONSTRUCTOR_001 **Name(s) of Testers: Ioana**

Test Description:

Tests that the runElection method in IrElection runs properly and correctly runs the election.

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

The testConstructor method is stored in the TestIrElection class in src, and uses methods from IrElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestIrElection.java, IrElection.java, and Candidate.java exist. The “IR.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
IR_TEST_PROCESS_FILE_001 **Name(s) of Testers: Ioana**

Test Description:

Tests that the runElection method in IrlElection runs properly and correctly runs the election.

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

The testProcessOneFile method is stored in the TestIrElection class in src, and uses methods from IrElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestIrElection.java, IrElection.java, and Candidate.java exist. The “IR.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
IR_TEST_ASSIGN_BALLOTS_001 **Name(s) of Testers: Ioana**

Test Description:

Tests that the runElection method in IrElection runs properly and correctly runs the election.

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

The testAssignBallots method is stored in the TestIrElection class in src, and uses methods from IrElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestIrElection.java, IrElection.java, and Candidate.java exist. The “IR.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
IR_TEST_RUN_ELECTION_001 **Name(s) of Testers: Ioana**

Test Description:

Tests that the runElection method in IrElection runs properly and correctly runs the election.

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

The testRunElection method is stored in the TestIrElection class in src, and uses methods from IrElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestIrElection.java, IrElection.java, and Candidate.java exist. The “IR.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
IR_TEST_RUN_ELECTION_2WAYTIE_001 **Name(s) of Testers: Ioana**

Test Description:

Tests that the runElection method in IrElection runs properly and correctly runs the election.

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

The testRunElectionIR2WayTie method is stored in the TestIrElection class in src, and uses methods from IrElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestIrElection.java, IrElection.java, and Candidate.java exist. The “IR.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System
Team# 5

Test Stage: Unit X System __ **Test Date: 5/2/2021**

Test Case ID#:
IR_TEST_ELIMINATE_BALLOT_001 **Name(s) of Testers: Ioana**

Test Description:

Tests that the testEliminateBallot method in IrElection runs properly and correctly runs the election.

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

The testEliminateBallot method is stored in the TestIrElection class in src, and uses methods from IrElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestIrElection.java, IrElection.java, and Candidate.java exist. The “IR.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

Project Name: Project 1: Voting System

Team# 5

Test Stage: Unit X System __

Test Date: 5/2/2021

Test Case ID#:
TEST_AUDIT_SERVICE_001

Name(s) of Testers: Ioana

Test Description:

Tests that the runElection method in IrElection runs properly and correctly runs the election.

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

The testRunElection method is stored in the TestIrElection class in src, and uses methods from IrElection.

Automated: yes__X_ no ___

Results: Pass __X___ Fail_____

Preconditions for Test: JUnit compiles properly and TestIrElection.java, IrElection.java, and Candidate.java exist. The “IR.txt” file is placed in the same directory as TestIrElection.java.

Step #	Test Step Description	Test Data	Expected Result	Actual Result	Notes
1	Compile all files.	Open terminal. Use javac to compile /src files.	Compiles with no errors	Compiles with no errors	
2	Run the Unit Test	Use the Junit framework to run the unit test.	Junit window should show a “Passed” message	Junit showed a “Passed” message	

Post condition(s) for Test:

The Junit framework indicated that the test was passed. The system has not changed.

The PBI, the Task Description (from Sprint Log) with Unique Testing Number:	Test 1: PBI 5 Processing multiple files for IR
Team Member(s) Responsible:	Trevor
Inputs:	IR.txt and IR2.txt
Tests:	We run the election with multiple inputted files.
Outputs:	The GUI will display relevant information to the screen including number of ballots, votes received, and the winner. Both audit and media files are produced.
Passed or Failed	Pass
Date:	5/2/2021

The PBI, the Task Description (from Sprint Log) with Unique Testing Number:	Test 2: PBI 5 Processing multiple files for OPL
Team Member(s) Responsible:	Cole
Inputs:	OPL.txt and OPL2.txt
Tests:	We run the election with multiple inputted files.
Outputs:	The GUI will display relevant information to the screen including number of ballots, votes received, and the winner. Both audit and media files are produced.
Passed or Failed	Pass
Date:	5/2/2021

The PBI, the Task Description (from Sprint Log) with Unique Testing Number:	Test 3: PBI 7 Invalidating ballots for IR
Team Member(s) Responsible:	Ioana
Inputs:	IRInvalidBallots.txt
Tests:	We run the election with a file that contains ballots that should be marked as invalid. We also run a JUnit test.
Outputs:	The GUI will display relevant information to the screen including number of ballots, votes received, and the winner. Both audit and media files are produced. It shows the number of ballots invalidated. The JUnit framework gives a pass message.
Passed or Failed	Pass
Date:	5/2/2021

The PBI, the Task Description (from Sprint Log) with Unique Testing Number:	Test 4: PBI 5 Allowing the system to process PO ballots.
Team Member(s) Responsible:	Trevor
Inputs:	PO.txt
Tests:	We run the election with a PO file. No information is printed to the screen. Therefore, much of this testing is done within the JUnit framework. testConstructor() testProcessOneFile() testMultipleFiles()
Outputs:	JUnit framework indicates a pass. The terminal gives no errors.
Passed or Failed	Pass
Date:	5/2/2021