|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team# 14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/4/18** |
| **Test Case ID#: 1** | **Name(s) of Testers: Meghann** |
| **Test Description:** This unit test tests the methods of the IRV ballot class after instantiating and initializing a known IRV ballot. Additionally, since the IRV ballot class extends the ballot class, methods from that class are also verified. | |
| **Automated: yes \_x\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Stored in: Project1/src/test/votingsystems/TestIRVBallot.java  Functions: testIsExhausted(), testGetNextVote(), testGetID()  Helper Functions: testBallotInitialize() |
| **Results: Pass \_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * An IRV Ballot is initialized with votes and an id | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | testBallotInitialize – a ballot used for testing is initialized | Arraylist of votes, ID | IRVBallot is initialized | IRV Ballot is initialized | Method called before each test |
| 2 | Test getNextVote | IRV ballot | 1 / testVotes.get(0) | 1 | pass |
| 3 | Test isExhausted | Irv ballot | False | False | Pass |
| 4 | Test getID – tests if ID of ballot is retrieved | IRV Ballot | 4 | 4 | Pass |

**Post condition(s) for Test:**

The correct vote is provided and saved as an int and a Boolean is returned showing if the number of votes is exhausted

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team# 14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/4/18** |
| **Test Case ID#: 2** | **Name(s) of Testers: Meghann** |
| **Test Description:**  This unit test tests the methods of the IRV candidate class after instantiating and initializing a known IRV ballot and candidate. Additionally, since the IRV candidate class extends the candidate class, methods from that class are also verified. | |
| **Automated: yes x\_\_\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Stored in: Project1/src/test/votingsystems/TestIRVCandidate.java  Functions: testAddBallot(), testEliminate(), testIsEliminated(), testGetNumVotes()  Helper Functions: initializeTestBallot() |
| **Results: Pass x\_\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * testBallot and Candidate are initialized with dummy data | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | initalizeTestBallot – adds votes to a vote arraylist and initializes IRVballot | Arraylist of votes, irv ballot | Initialized IRVBallot with votes and id | Initialized IRVBallot | Initializer |
| 2 | testAddBallot – tests the add ballot function of IRVCandidate | IRV Ballot, IRV Candidate, ArrayList for eliminated candidate’s ballot | Ballot added == ballot added to eliminated ballots | Equivalence holds true | Pass |
| 3 | testEliminate – test eliminate function of IRV Candidate | Irv ballot, IRV Candidate, array of IRVBallots | Array of ballots added == array of ballots eliminated | For one ballot, result is equivalent | Pass; may need to expand to test for multiple ballots |
| 4 | testIsEliminated – tests if candidate reflects eliminated after being eliminated | IRV Ballot, IRV Candidate | testCandidate.isEliminated() == true | testCandidate.isEliminated() == true | pass |
| 5 | testGetName – tests method from candidate.java | IRV candidate | “Jenny” | “Jenny” | pass |
| 6 | testGetNumVotes – tests method from candidate class | IRV Ballot and candidate | 1 | 1 | pass |

**Post condition(s) for Test:**

After test is run, each public method of IRVCandidate class and Candidate class are verified to work for at least one ballot.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team# 14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/4/18** |
| **Test Case ID#: 3** | **Name(s) of Testers: Meghann** |
| **Test Description:**  This unit test tests the methods of the IRV class through creating mock election files and known audit file results. | |
| **Automated: yes x\_\_\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  *Stored in:* Project1/src/test/votingsystems/TestIRV.java  *Functions:* testIRVRandomWinner(), testRunElectionEfficiency(), testIRVMajorityPopularVote(), testIRVTenThousandVotes(), testIRVOneVote(), testIRVOneCandidate(), testIRVNoMajorityPopularVote(), testIRVConsequentialTieTwoCandidates()  *Helper Functions:* testFileAuditPair(), testFileAuditPairRandomMsg() |
| **Results: Pass x\_\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * The election files for each specific case of an election are created and stored in the testing directory * The expected audit files for each specific case are created and stored in the testing directory | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | testRunElectionEfficiency – tests the time it takes to run the election for 100,000 votes in milliseconds | Test time in milliseconds | Test time is < 8 minutes | Test time < 8 minutes | 8 minutes = 480,000 milliseconds  Pass |
| 2 | testIRVRandomWinner – test if the election outputs the correct audit file when a winner is chosen at random | randomWinner.txt – election file for a random winner | The election winner is one of the three expected candidates  This string is expected in the audit file: "NOTE: This elimination was the result of a random toss due to a consequential tie in least amount of votes.” | The election winner is one of the three expected candidates  The expected string was found in the audit file | Pass |
| 3 | testIRVMajorityPopularVote – test if the election outputs the correct audit file when there is a winner by majority popular vote | majorityPopularVote.txt – election file for a winner by majority vote  majorityPopularVoteAudit.txt – expected audit file to be produced | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 4 | testIRVTenThousandVotes – test if the election outputs the correct audit file when there is 10,000 ballots | tenThousandVotes.txt – election file with 10,000 ballots  tenThousandVotesAudit.txt – expected audit file | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 5 | testIRVOneVote() – test if the election outputs the correct audit file when there is only one ballot | oneVote.txt – election file with one ballot and multiple candidates  oneVoteAudit.txt – expected audit file | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 6 | testIRVOneCandidate() – test if the election outputs the correct audit file when there is only one candidate | oneCandidate.txt – election file with one candidate  oneCandidateAudit.txt – expected audit file | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 7 | testIRVNoMajorityPopularVote() – test if the election outputs the correct audit file when there is no winner | noMajorityPopularVote – election file with no winner by majority popular vote  noMajorityPopularVoteAudit.txt – expected audit file | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 8 | testIRVConsequentialTieTwoCandidates() – test if the election outputs the correct audit file when there is a Tie between two candidates | consequentialTieTwoCandidates.txt – election file with a tie between two candidates  consequentialTieTwoCandidatesAudit.txt – an expected audit file | The election winner is one of the two expected candidates  This string is expected in the audit file: "NOTE: This elimination was the result of a random toss due to a consequential tie in least amount of votes.” | The election winner is one of the two expected candidates  This string is expected in the audit file: "NOTE: This elimination was the result of a random toss due to a consequential tie in least amount of votes.” | Pass |

**Post condition(s) for Test:**

After test is run, audit files are produced for each method testing a certain type of election.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/11/18** |
| **Test Case ID#: 4** | **Name(s) of Testers: Jake Nippert** |
| **Test Description:**  **The Mariah File Chooser GUI allows the user to select a file or open a file by name manually and initializes the processing of this election file.** |  |
| **Automated: yes\_\_\_ no \_x\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  **Project1/testing/MariahGUI/FileChooser\***  **Project1/testing/MariahGUI/ManualEntry\***  **Project1/testing/MariahGUI/Snow\***  **Project1/testing/MariahGUI/NoGUI\*** |
| **Results: Pass \_\_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test: No command line arguments are passed in to MariahEP when running main.** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | Run java MariahEP | N/A | Opens GUI and with system file chooser and a manual entry | Opens GUI and with system file chooser and a manual entry | Pass |
| 2 | Open file from system file chooser | Election File | Processes election file in main | Processes election file | Pass |
| 3 | Open file from manual file name entry | Election File | Processes election file in main | Processes election file | Pass |
| 4 | Click snow button on header label (Repeat 2-3 with expected outputs) | Election File | Snows in background and processes election file in main | Snows in background and processes election file in main | Pass |
| 5 | NoGUI command line | N/A | No GUI is displayed | No GUI is displayed | PASS |

**Post condition(s) for Test:**

Election file is not changed. State of system is changed as an Audit file will have been produced and the election results GUI will be displayed.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/11/18** |
| **Test Case ID#: 5** | **Name(s) of Testers: Jake Nippert** |
| **Test Description:**  **The Mariah Election Results GUI allows the user to view the immediate results of the election as well as open the audit file that was produced.** |  |
| **Automated: yes\_\_\_ no \_x\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  **Project1/testing/MariahGUI/ElectionResults\*** |
| **Results: Pass \_\_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test: The command line argument “NoGUI” was NOT passed in to MariahEP when running main.** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | Run java MariahEP | Election File | Results are placed in the GUI Election Results | Results are placed in the GUI Election Results | Pass |
| 2 | Clicking Open Audit File makes a system call open of the audit file | N/A | Opens the correct Audit File | Opens the correct Audit File | Pass |
| 3 | Exiting returns the user to the File Chooser GUI to restart the process | N/A | User returns to File Chooser GUI | User returns to File Chooser GUI | Pass |
| 4 | Repeat Unit Test Case ID #8 | N/A | Passes all Test ID #4 Use Cases | Passes all Test ID #4 Use Cases | Pass |

**Post condition(s) for Test:**

Election file is not changed. State of system is changed as an Audit file will have been produced and the election results GUI will be displayed.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/12/18** |
| **Test Case ID#: 6** | **Name(s) of Testers: Christine Tsai** |
| **Test Description:**  These unit tests ensure that the Auditor is correctly initialized and appends strings to the correct parts of the auditor file. It also makes sure that the audit file created contains the correct information. |  |
| **Automated: yes\_x\_\_ no \_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Project1/testing/Auditor/TestAuditor.java  testAuditProcess(), testAuditResult(), testAuditSetup(), testAuditStructure(), testCreateAuditFile() |
| **Results: Pass \_\_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:** No preconditions**.** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | initializeTestAuditor() – initializes an Auditor object for testing | Arraylist of random test strings | The Auditor object used for testing is initialized | The Auditor object used for testing is initialized |  |
| 2 | Test auditProcess | Arraylist of random test strings | Only audit process is present in audit file | Only audit process is present in audit file | Pass |
| 3 | Test auditResult | Arraylist of random test strings | Only audit result is present in audit file | Only audit result is present in audit file | Pass |
| 4 | Test auditSetup | Arraylist of random test strings | Only audit setup is present in audit file | Only audit setup is present in audit file | Pass |
| 5 | Test auditStructure | Arraylist of random test strings | Process, result and setup are present in correct order and separated properly | Process, result and setup are present in correct order and separated properly | Pass |
| 6 | Test createAuditFile | Arraylist of random test strings | Audit file is created on system | Audit file is created on system | Pass |
| 7 | testAuditStructureSPR – test audit structure with set up, process and results | Arraylist of random test strings | The output would be the audit’s set up, process, and string in the respective order | The output would be the audit’s set up, process, and string in the respective order | pass |
| 8 | testAuditStrucutreSP – test audit structure with set up and process | Arraylist of random test strings | The output would be the audit’s set up and process in the respective order | The output would be the audit’s set up and process in the respective order | pass |
| 9 | testAudStructureSR – test audit structure with set up and results | Arraylist of random test strings | The output would be the audit’s set up and results in the respective order | The output would be the audit’s set up and results in the respective order | pass |
| 10 | testAuditStucturePR – test audit structure process and result | Arraylist of random test strings | The output would be the audit’s process and results in the respective order | The output would be the audit’s process and results in the respective order | pass |
| 11 | testCreateAuditFile – test creating an auto file | Arraylist of random test strings | An audit file is created | An audit file is created | pass |

**Post condition(s) for Test:**

System state is unchanged because test audit files that are produced are deleted after all lines are read.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team# 14** | |
| **Test Stage: Unit \_\_ System \_x\_** | **Test Date: 11/4/18** |
| **Test Case ID#: 7** | **Name(s) of Testers: Meghann, Cassandra, Jake** |
| **Test Description:**  This test runs all of our unit tests and essentially tests proper functionality of the driver class. | |
| **Automated: yes x\_\_\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  *Stored in:* Project1/src/test/main/TestMariahEP.java  *Functions:* testMainIRVRandomWinner(), testMainIRVMajorityPopularVote(), testMainIRVTenThousandVotes(), testMainIRVOneVote(), testMainIRVOneCandidate(), testMainIRVNoMajorityPopularVote(), testMainIRVConsequentialTieTwoCandidates()  *Helper Functions:* testFileAuditPair(), testFileAuditPairRandomMsg() |
| **Results: Pass x\_\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * The election files for each specific case of an election are created and stored in the testing directory * The expected audit files for each specific case are created and stored in the testing directory | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 |  |  |  |  |  |
| 2 | testMainIRVRandomWinner – test if the election outputs the correct audit file when a winner is chosen at random | randomWinner.txt – election file for a random winner | The election winner is one of the three expected candidates  This string is expected in the audit file: "NOTE: This elimination was the result of a random toss due to a consequential tie in least amount of votes.” | The election winner is one of the three expected candidates  The expected string was found in the audit file | Pass |
| 3 | testMainIRVMajorityPopularVote – test if the election outputs the correct audit file when there is a winner by majority popular vote | majorityPopularVote.txt – election file for a winner by majority vote  majorityPopularVoteAudit.txt – expected audit file to be produced | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 4 | testMainIRVTenThousandVotes – test if the election outputs the correct audit file when there is 10,000 ballots | tenThousandVotes.txt – election file with 10,000 ballots  tenThousandVotesAudit.txt – expected audit file | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 5 | testMainIRVOneVote() – test if the election outputs the correct audit file when there is only one ballot | oneVote.txt – election file with one ballot and multiple candidates  oneVoteAudit.txt – expected audit file | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 6 | testMainIRVOneCandidate() – test if the election outputs the correct audit file when there is only one candidate | oneCandidate.txt – election file with one candidate  oneCandidateAudit.txt – expected audit file | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 7 | testMainIRVNoMajorityPopularVote() – test if the election outputs the correct audit file when there is no winner | noMajorityPopularVote – election file with no winner by majority popular vote  noMajorityPopularVoteAudit.txt – expected audit file | The audit file produced matches the expected audit file exactly | The audit file produced matches the expected audit file exactly | Pass |
| 8 | testMainIRVConsequentialTieTwoCandidates() – test if the election outputs the correct audit file when there is a Tie between two candidates | consequentialTieTwoCandidates.txt – election file with a tie between two candidates  consequentialTieTwoCandidatesAudit.txt – an expected audit file | The election winner is one of the two expected candidates  This string is expected in the audit file: "NOTE: This elimination was the result of a random toss due to a consequential tie in least amount of votes.” | The election winner is one of the two expected candidates  This string is expected in the audit file: "NOTE: This elimination was the result of a random toss due to a consequential tie in least amount of votes.” | Pass |
|  |  |  |  |  |  |
| 9 | testMainOPLVOneSeatOneWinner – test an election where there is one seat and one winner with six candidates | oneSeatOneWinner election file | Naruto (Senju) | Naruto (Senju) | pass |
| 10 | testMainOPLVOneSeatOneWinnerOneVote –  test an election where there is one seat and one winner with six candidates with only one vote cast | oneSeatOneWinnerOneVote election file | Naruto (Senju) | Naruto (Senju) | pass |
| 11 | testMainOPLVSixSeatsSixCandidatesEqual – test an election where there is six seats and all candidates receive one vote | sixSeatsSixCandidatesEqual election file | All candidates win a seat | All candidates win a seat | pass |
| 12 | testMainOPLVConsequentialPartyTieTwoCandidates – test an election where there is a consequential tie between candiates on the same party | consequentialPartyTieTwoCandidates | One of the two candidates that are tied win the seat with a random flip | One of the two candidates that are tied win the seat with a random flip | pass |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 13 | testMainOPLVtwoSeatsFiveCandidatesTenVotes - test an election where there is two seats with five candidates and the remaining seat goes to the party with the largest remainder. | twoSeatsFiveCandidatesTen  Votes | Election Winners: Naruto (Senju)  Sakura (Akatsuki) | Election Winners: Naruto (Senju)  Sakura (Akatsuki) | pass |
| 14 | testMainOPLVtwoSeatsFiveCandidatesUnequal –  Test an election where there are two seats with five candidates and the party with the largest remainder does not have a candidate to fill seat so the seat is allocated to the next party with the next highest largest remainder. | twoSeatsFiveCandidatesUneq  ual | Election Winners: Naruto (Senju)  Sai (Anbu) | Election Winners: Naruto (Senju)  Sai (Anbu) | pass |

**Post condition(s) for Test:**

After test is run, audit files are produced for each method testing a certain type of election.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team# 14** | |
| **Test Stage: Unit x System \_\_** | **Test Date: 11/11/18** |
| **Test Case ID#: 8** | **Name(s) of Testers: Cassandra Chanthamontry** |
| **Test Description:** This unit test tests the methods of the OPLV ballot class after instantiating and initializing a known OPLV ballot. Additionally, since the OPLV ballot class extends the ballot class, methods from that class are also verified. | |
| **Automated: yes \_x\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Stored in: Project1/src/test/votingsystem/TestOPLVBallot.java  Functions: testOPLVTestVotes(), testOPLVTestIDs()  Helper Functions: none |
| **Results: Pass \_x\_\_ Fail\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * An OPLV Ballot is initialized with votes and an id | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | testOPLVBallot – testing default constructor fails | Calling the OPLVBallot default constructor | Illegal argument error | Illegal argument error | pass |
| 2 | Test testOPLVTestVotes – testing the getVote function | OPLV Ballot | A ballot with 0, 1, 2, 3, and 4 added respectively have that amount of vote. | 0, 1, 2, 3, and 4 votes. | pass |
| 3 | Test testOPLVTestIDS – testing the getID function | OPLV Ballot | A ballot with ID, 0, 1, 2, 3, and 4 as their ID, will return their respective ID. | Ballot 0, 1, 2, 3, and 4 have their numerical value as their ID. | Pass |

**Post condition(s) for Test:**

The system does not change.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team# 14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/11/18** |
| **Test Case ID#: 9** | **Name(s) of Testers: Cassandra Chanthamontry** |
| **Test Description:**  This unit test tests the methods of the OPLV candidate class after instantiating and initializing a known OPLV ballot, party, candidate. Additionally, since the OPLV candidate class extends the candidate class, methods from that class are also verified. | |
| **Automated: yes x\_\_\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Stored in: Project1/src/test/votingsystems/TestOPLVCandidate.java  Functions: testOPLVCandidate(), testOPLVCandidateName(), testOPLVParty(), testOPLVCandidateCastZeroVote(), testOPLVCandidateCastOneVote(), testOPLVCandidateCastFiveVotes()  Helper Functions: initializeTestCandidate() |
| **Results: Pass x\_\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * Party and Candidate are initialized with dummy data | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | initalizeTestCandidate – Intitalize a candidate with party | OPLV Candidate | Intialize OPLVCandidate | Initialized OPLVCandidate | Initializer |
| 2 | testOPLVCandidate– test the default constructor fails | OPLV Candidate | Illegal Argument error | Illegal Argument error | Pass |
| 3 | testOPLVCandidateName – test if candidate name is the same when using getName function | OPLV Candidate, OPLV Party | “TestCandidate” | “TestCandidate” | Pass |
| 4 | testOPLVCandidateParty – test if party name is the same when using getParty function | OPLV Candidate, OPLV Party | “TestParty” | “TestParty” | Pass |
| 5 | testOPLVCandidateCastZeroVote– test that the default vote count for the candidate and respective party is 0 | OPLV Candidate, OPLV Party | 0 vote for both candidate and party | 0 vote for both candidate and party | pass |
| 6 | testOPLVCandidateCastOneVote – test that one vote for candidate also cast one vote for the respective party | OPLV Candidate, OPLV Party | 1 vote for both candidate and party | 1 vote for both candidate and party | pass |
| 7 | testOPLVCandidateCastFiveVotes – test that five votes for a candidate also cast five votes for respective party | OPLV Candidate, OPLV Party | 5 vote for both candidate and party | 5 vote for both candidate and party | pass |

**Post condition(s) for Test:**

After test is run, each public method of OPLVCandidate class and Candidate class are verified to work for at least one ballot.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team# 14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/11/18** |
| **Test Case ID#: 10** | **Name(s) of Testers: Cassandra Chanthamontry** |
| **Test Description:**  This unit test tests the public methods of the IRV class after instantiating and initializing known IRV ballots and IRV Voting System. Additionally, since the IRV class extends the VotingSystem class, methods from that class are also verified. | |
| **Automated: yes x\_\_\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Stored in: Project1/src/test/votingsystems/TestParty.java  Functions: testPartyName(), testPartyCastZeroVote(), testPartyCastOneVote(), testPartyCastFiveVotes(), testPartyZeroVotes(), testPartyOneSeat(), testGetWinningCandidatesNumSeatsEqualsNumCandidates(),  testGetWinningCandidatesLessNumSeatsThanNumCandidates(),  testRankCandidatesZeroCandidate(), testRankCandidatesOneCandidate(), testRankCandidatesTwoCandidates(), testRankCandidatesFiveCandidate()  Helper Functions: initializeTestParty() |
| **Results: Pass x\_\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * OPLVCandidate arraylist of arraylist<OPLVCandidate > is initialized | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | initializeTestParty() – initialized a party with candidates | Party, Arraylist OPLVCandidate | Initialized OPLV party and candidate | Initialized OPLV party and candidate | Initializer |
| 2 | testParty() – test that default constructor is not used | Party, Arraylist OPLVCandidate | Illegal Argument error | Illegal Argument error | Pass |
| 3 | testPartyName() – test if party name is the same when using getName function | Party, Arraylist OPLVCandidate | “TestParty” | “TestParty” | Pass |
| 4 | testPartyCastZeroVote – test if party default vote is 0 | Party, Arraylist OPLVCandidate | 0 | 0 | Pass |
| 5 | testPartyCastOneVote – test if party has one vote when one vote is casted | Party, Arraylist OPLVCandidate | 1 | 1 | Pass |
| 6 | testPartyCastFiveVotes – test if party has five votes when five votes are casted | Party, Arraylist OPLVCandidate | 5 | 5 | Pass |
| 7 | testPartyZeroSeats – test if default seat number is 0 | Party, Arraylist OPLVCandidate | 0 | 0 | Pass |
| 8 | testPartyOneSeat – test if when allocated one seat, one seat is returned with function getNumSeats | Party, Arraylist OPLVCandidate | 1 | 1 | Pass |
| 9 | testGetWinningCandidatesNumSeatsEqualsNumCandidates – test if there is same amount of seats and candidate, the function getWinningCandidates will return the whole candidates | Party, OPLVCandidate array | [TestCandidate1, TestCandidate2, TestCandidate3, TestCandidate4, TestCandidate5] | [TestCandidate1, TestCandidate2, TestCandidate3, TestCandidate4, TestCandidate5] | pass |
| 10 | testGetWinningCandidatesLessNumSeatsThanNumCandidates – test if there is less seats than candidates, only winning candidates are returned | Party, OPLVCandidate | The number of candidates returned from getWinningCandidates is equal to the number of seats indicated | The number of candidates returned from getWinningCandidates is equal to the number of seats indicated | pass |
| 11 | testRankCandidatesZeroCandidate() – when there is zero candidates that are ranked, returns zero candidates | Party, Arraylist OPLVCandidate | [] | [] | pass |
| 12 | testRankCandidateOneCandidate() – when there is one candidate that are ranked, returns that one candidate | Party, Arraylist OPLVCandidate | [TestCandidate1] | [TestCandidate1] | pass |
| 13 | testRankCandidatesTwoCandidates() – test ranking two candidates returns two candidates in the correct ranking | Party, Arraylist OPLVCandidate | [TestCandidate1, TestCandidate2] | [TestCandidate1, TestCandidate2] | pass |
| 14 | testRankCandidatesFiveCandidates() -test ranking 5 candidates, give back the five candidates in the correct rank | Party, Arraylist OPLVCandidate | [TestCandidate1, TestCandidate2, TestCandidate3, TestCandidate4, TestCandidate5] | [TestCandidate1, TestCandidate2, TestCandidate3, TestCandidate4, TestCandidate5] | pass |

**Post condition(s) for Test:**

After test is run, the Party and its method are verified

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team# 14** | |
| **Test Stage: Unit x System \_\_** | **Test Date: 11/13/18** |
| **Test Case ID#: 11** | **Name(s) of Testers: Cassandra Chanthamontry** |
| **Test Description:** This unit test test the methods of the OPLV class through creating mock election files and known audit files | |
| **Automated: yes \_x\_ no \_\_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Stored in: TestOPLV.java  Functions: testOPLV(), testRunElectionTwice(), testRunElectionEfficiency(), testOPLVOneSeatOneWinner(), testOPLVOneSeatOneWinnerOneVote(), testOPLVSixSeatsSixCandidateEqual(), testOPLVConsequentialPartyTieTwoCandidates(), testOPLVConsequentialPartyTieThreeCandidates(), testOPLVconsequentialTie(), testOPLVtwoSeatsFiveCandidatesTenVotes(), testOPLVTwoSeatsFiveCandidatesUnequal()  Helper Functions: votingsystemFromFile(), OPLVBallotsFromFile(), CandidatePartyPairsSeparator(), testFileAuditPair(), testFileAuditPairRandomMsg() |
| **Results: Pass \_x\_\_ Fail\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * The election files for each specific case of an election are created and stored in the testing directory * The expected audit files for each specific case are created and stored in the testing directory | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | testOPLV - test that default constructor tea is not used | OPLV | Illegal argument error | Illegal argument error | Pass |
| 2 | testRunElectionTwice – test that when election is run twice, error message shows up | Votingsystem, OPLV | Error message | Error message | pass |
| 3 | testRunElectionEfficiency – test 100, 000 ballots are processed under 8 minutes | String of candidates and parties, VotingSystem | The time to parse through the ballot and calculate results is under 8 minutes | The time to parse through the ballot and calculate results is under 8 minutes | pass |
| 4 | testOPLVOneSeatOneWinner – test an election where there is one seat and one winner with six candidates | oneSeatOneWinner election file | Naruto (Senju) | Naruto (Senju) | pass |
| 5 | testOPLVOneSeatOneWinnerOneVote –  test an election where there is one seat and one winner with six candidates with only one vote cast | oneSeatOneWinnerOneVote election file | Naruto (Senju) | Naruto (Senju) | pass |
| 6 | testOPLVSixSeatsSixCandidatesEqual – test an election where there is six seats and all candidates receive one vote | sixSeatsSixCandidatesEqual election file | All candidates win a seat | All candidates win a seat | pass |
| 7 | testOPLVConsequentialPartyTieTwoCandidates – test an election where there is a consequential tie between candiates on the same party | consequentialPartyTieTwoCandidates | One of the two candidates that are tied win the seat with a random flip | One of the two candidates that are tied win the seat with a random flip | pass |
| 8 | testOPLVConsequentialPartyTieThreeCandidates   * Test an election where there is a consequential tie between three candidates on the same party | consequentialPartyTieThreeCandidates | One of the three candidates who are tied win the seat with a random flip | One of the three candidates who are tied win the seat with a random flip | pass |
| 9 | testOPLVconsequtialTie – test an election where there is a consequential tie between candidates | consequentialCandidateTie | The 4 certain winners and the randomly chosen fifth winner is returned | The 4 certain winners and the randomly chosen fifth winner is returned | pass |
| 10 | testOPLVtwoSeatsFiveCandidatesTenVotes - test an election where there is two seats with five candidates and the remaining seat goes to the party with the largest remainder. | twoSeatsFiveCandidatesTenVotes | Election Winners: Naruto (Senju), Sakura (Akatsuki) | Election Winners: Naruto (Senju), Sakura (Akatsuki) | pass |
| 11 | testOPLVtwoSeatsFiveCandidatesUnequal –  Test an election where there are two seats with five candidates and the party with the largest remainder does not have a candidate to fill seat, so the seat is allocated to the next party with the next highest largest remainder. | twoSeatsFiveCandidatesUnequal | Election Winners: Naruto (Senju), Sai (Anbu) | Election Winners: Naruto (Senju), Sai (Anbu) | pass |

**Post condition(s) for Test:**

After test is run, audit files are produced for each method testing a certain type of election.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/27/18** |
| **Test Case ID#: 12** | **Name(s) of Testers: Jake Nippert** |
| **Test Description:**  **The Mariah File Chooser GUI allerts the user to select a different file or open a different file by name manually when the previously selected file was not in a standardized election file format.** |  |
| **Automated: yes\_\_\_ no \_x\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  **Project1/testing/MariahGUI/InvalidFilePrompt\*** |
| **Results: Pass \_\_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test: A txt file exists that is not a valid standardized IRV or OPLV file.** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | Run java MariahEP | N/A | Opens GUI and with system file chooser and a manual entry | Opens GUI and with system file chooser and a manual entry | Pass |
| 2 | Open invalid file from system file chooser | Invalid File Path | Alert user of invalid file | Alters user of invalid file | Pass |
| 3 | Close alert | N/A | Alert is no longer displayed | Alert is no longer displayed | Pass |
| 4 | Run Test Case #4 | N/A | Unit tests pass | Unit tests pass | Pass |

**Post condition(s) for Test:**

Incorrect file is not changed. No audit file or invalid ballots file is created.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 11/27/18** |
| **Test Case ID#: 12** | **Name(s) of Testers: Jake Nippert** |
| **Test Description:**  **The Mariah File Chooser GUI alerts the user to select a different file or open a different file by name manually when the previously selected file was not in a standardized election file format.** |  |
| **Automated: yes\_\_\_ no \_x\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  **Project1/testing/InvalidFiles/\*** |
| **Results: Pass \_\_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test: A text file exists that will generate at least every possible error message for an exception.** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | Run java MariahEP | N/A | Opens GUI and with system file chooser and a manual entry | Opens GUI and with system file chooser and a manual entry | Pass |
| 2 | Open each text file to produce every possible error message for an invalid file | Invalid File Path | Alerts user of invalid file with the correct error message | Alerts user of invalid file with the correct error message | Pass |

**Post condition(s) for Test:**

The GUI continues running and is functional.

|  |  |
| --- | --- |
| **Project Name: Project 2: Voting System Team# 14** | |
| **Test Stage: Unit x System \_\_** | **Test Date: 11/29/18** |
| **Test Case ID#: 13** | **Name(s) of Testers: Cassandra Chanthamontry** |
| **Test Description:** This test the createQuickPrintSum function in both IRV and OPL through creating a mock election file | |
| **Automated: yes \_\_ no \_x\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Stored in: Project2/testing/QuickPrintSummary  Functions: createQuickPrintSum |
| **Results: Pass \_x\_\_ Fail\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * GUI needs to be run and user must select Print Report | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | In the GUI clicking File, then Print Report calls createQuickPrintSum to print the quick summary report | N/A | A Quick Print summary was printed. Refer to project2/testing/QuickPrintSummary for images | A Quick Print summary was printed. Refer to project2/testing/QuickPrintSummary for images | Pass |

**Post condition(s) for Test:**

After test is run, the creation of a quick summary that can be printed is verified.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 12/08/18** |
| **Test Case ID#: 14** | **Name(s) of Testers: Christine Tsai** |
| **Test Description:**  These unit tests check that IRV methods are working properly, specifically those associated with the ballot validation process (performBallotValidation(), isBallotValid()). There are a variety of test cases where certain ballots are invalid, which affects the final election results. |  |
| **Automated: yes\_x\_\_ no \_\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Project1/src/test/votingsystems/TestIRV.java   * testIRVEvenCandidatesNoInvalidBallots() * testIRVEvenCandidatesSomeInvalidBallots() * testIRVEvenCandidatesAllButOneInvalidBallots() * testIRVEvenCandidatesAllInvalidBallots() * testIRVOddCandidatesNoInvalidBallots() * testIRVOddCandidatesSomeInvalidBallots() * testIRVOddCandidatesAllButOneInvalidBallots() * testIRVOddCandidatesAllInvalidBallots() |
| **Results: Pass \_\_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:** The election files and their respective expected audit files must be created and stored in the testing directory. Each test case will have its own election file and expected audit file. | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | testIRVEvenCandidatesNoInvalidBallots() – test an election where **no** ballots are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **even**. | evenCandidatesNoInvalidBallots.txt – the file that would be provided for an election with no invalid ballots and an even number of candidates  evenCandidatesNoInvalidBallotsAudit.txt – the expected audit file from the election  evenCandidatesNoInvalidBallotsInvalidAudit.txt – the expected invalid audit file from the election that holds all the removed invalid ballots | After running the voting system on the provided election file, the contents of the audit file that is produced match that of the expected audit file and the contents of the invalid audit file that is produced match that of the expected invalid audit file. | The produced audit file and the expected audit file have the same contents, and the produced invalid audit file and the expected invalid audit file have the same contents. | Pass |
| 2 | testIRVEvenCandidatesSomeInvalidBallots() – test an election where **some** ballots are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **even**. | evenCandidatesSomeInvalidBallots.txt – the file that would be provided for an election with some invalid ballots and an even number of candidates  evenCandidatesSomeInvalidBallotsAudit.txt – the expected audit file from the election  evenCandidatesSomeInvalidBallotsInvalidAudit.txt – the expected invalid audit file from the election that holds all the removed invalid ballots | After running the voting system on the provided election file, the contents of the audit file that is produced match that of the expected audit file and the contents of the invalid audit file that is produced match that of the expected invalid audit file. | The produced audit file and the expected audit file have the same contents, and the produced invalid audit file and the expected invalid audit file have the same contents. | Pass |
| 3 | testIRVEvenCandidatesAllButOneInvalidBallots() – test an election where **all but one** ballot are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **even**. | evenCandidatesAllButOneInvalidBallots.txt – the file that would be provided for an election with all but one invalid ballot and an even number of candidates  evenCandidatesAllButOneInvalidBallotsAudit.txt – the expected audit file from the election  evenCandidatesAllButOneInvalidBallotsInvalidAudit.txt – the expected invalid audit file from the election that holds all the removed invalid ballots | After running the voting system on the provided election file, the contents of the audit file that is produced match that of the expected audit file and the contents of the invalid audit file that is produced match that of the expected invalid audit file. | The produced audit file and the expected audit file have the same contents, and the produced invalid audit file and the expected invalid audit file have the same contents. | Pass |
| 4 | testIRVEvenCandidatesAllInvalidBallots() – test an election where **all** ballots are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **even**. | evenCandidatesAllInvalidBallots.txt – the file that would be provided for an election with all invalid ballots and an even number of candidates | After running the voting system on the provided election file, the system throws an InvalidFileException error with the message “There are not enough valid ballots.” | The InvalidFileException error is thrown and caught. | Pass |
| 5 | testIRVOddCandidatesNoInvalidBallots() – test an election where **no** ballots are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **odd**. | oddCandidatesNoInvalidBallots.txt – the file that would be provided for an election with no invalid ballots and an odd number of candidates  oddCandidatesNoInvalidBallotsAudit.txt – the expected audit file from the election  oddCandidatesNoInvalidBallotsInvalidAudit.txt – the expected invalid audit file from the election that holds all the removed invalid ballots | After running the voting system on the provided election file, the contents of the audit file that is produced match that of the expected audit file and the contents of the invalid audit file that is produced match that of the expected invalid audit file. | The produced audit file and the expected audit file have the same contents, and the produced invalid audit file and the expected invalid audit file have the same contents. | Pass |
| 6 | testIRVOddCandidatesSomeInvalidBallots() – test an election where **some** ballots are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **odd**. | oddCandidatesSomeInvalidBallots.txt – the file that would be provided for an election with some invalid ballots and an odd number of candidates  oddCandidatesSomeInvalidBallotsAudit.txt – the expected audit file from the election  oddCandidatesSomeInvalidBallotsInvalidAudit.txt – the expected invalid audit file from the election that holds all the removed invalid ballots | After running the voting system on the provided election file, the contents of the audit file that is produced match that of the expected audit file and the contents of the invalid audit file that is produced match that of the expected invalid audit file. | The produced audit file and the expected audit file have the same contents, and the produced invalid audit file and the expected invalid audit file have the same contents. | Pass |
| 7 | testIRVOddCandidatesAllButOneInvalidBallots() – test an election where **all but one** ballot are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **odd**. | oddCandidatesAllButOneInvalidBallots.txt – the file that would be provided for an election with all but one invalid ballot and an odd number of candidates  oddCandidatesAllButOneInvalidBallotsAudit.txt – the expected audit file from the election  oddCandidatesAllButOneInvalidBallotsInvalidAudit.txt – the expected invalid audit file from the election that holds all the removed invalid ballots | After running the voting system on the provided election file, the contents of the audit file that is produced match that of the expected audit file and the contents of the invalid audit file that is produced match that of the expected invalid audit file. | The produced audit file and the expected audit file have the same contents, and the produced invalid audit file and the expected invalid audit file have the same contents. | Pass |
| 8 | testIRVOddCandidatesAllInvalidBallots() – test an election where **all** ballots are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **odd**. | oddCandidatesAllInvalidBallots.txt – the file that would be provided for an election with all invalid ballots and an odd number of candidates | After running the voting system on the provided election file, the system throws an InvalidFileException error with the message “There are not enough valid ballots.” | The InvalidFileException error is thrown and caught. | Pass |

**Post condition(s) for Test:**

The system state is unchanged because test audit files that are produced from running the voting system are deleted after all lines are read.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 12/08/18** |
| **Test Case ID#: 15** | **Name(s) of Testers: Christine Tsai** |
| **Test Description:**  Manually test that the IRV methods are working properly on election files with all invalid ballots. When tested in the GUI, an error message will pop-up indicating that there are not enough valid ballots to proceed with a normal election run. |  |
| **Automated: yes\_\_\_ no \_x\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Project1/src/test/votingsystems/TestIRV.java   * testIRVEvenCandidatesAllInvalidBallots() * testIRVOddCandidatesAllInvalidBallots() |
| **Results: Pass \_\_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:** Each test case’s election file must be created and stored in the testing directory. | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | testIRVEvenCandidatesAllInvalidBallots() – test an election where **all** ballots are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **even**. | evenCandidatesAllInvalidBallots.txt – the file that would be provided for an election with all invalid ballots and an even number of candidates | After opening the GUI and selecting the correct election file to run the voting system on, an error message pops up saying “Selected file is not a standardized IRV or OPLV election file. ERROR: There are not enough valid ballots.” | The GUI is opened and the election file is selected. The correct error message pops up in the GUI. | Pass |
| 2 | testIRVOddCandidatesAllInvalidBallots() – test an election where **all** ballots are invalid, an invalid ballot having fewer than half the candidates ranked, and the number of candidates is **odd**. | oddCandidatesAllInvalidBallots.txt – the file that would be provided for an election with all invalid ballots and an odd number of candidates | After opening the GUI and selecting the correct election file to run the voting system on, an error message pops up saying “Selected file is not a standardized IRV or OPLV election file. ERROR: There are not enough valid ballots.” | The GUI is opened and the election file is selected. The correct error message pops up in the GUI. | Pass |

**Post condition(s) for Test:**

The system state is unchanged because the elections with all invalid ballots are never run. The error message indicates to the user that the election cannot be run.

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit x\_\_ System \_\_** | **Test Date: 12/8/18** |
| **Test Case ID#: 16** | **Name(s) of Testers: Meghann Silagan** |
| **Test Description:**  **This is a manual test to manually test that the Election Table is displaying the expected results based on the IRV election input file.** |  |
| **Automated: yes\_\_\_ no \_x\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Project2\testing\IRV\consequentialTieTwoCandidates.txt  Project2\testing\IRV\majorityPopularVote.txt  Project2\ testing\IRV\noMajorityPopularVote.txt  Project2\ testing\IRV\OneCandidate.txt  Project2\ testing\IRV\OneVote.txt  Project2\ testing\IRV\randomWinner.txt  Project2\ testing\IRV\tenThousandVotes.txt |
| **Results: Pass \_\_x\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test: A txt file exists that is not a valid standardized IRV or OPLV file.** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | Run java MariahEP | N/A | Opens GUI and with system file chooser and a manual entry | Opens GUI and with system file chooser and a manual entry | Pass |
| 2 | Open various test election files (indicated above) for an IRV election | Project2\testing\IRV\consequentialTieTwoCandidates.txt  Project2\testing\IRV\majorityPopularVote.txt  Project2\ testing\IRV\noMajorityPopularVote.txt  Project2\ testing\IRV\OneCandidate.txt  Project2\ testing\IRV\OneVote.txt  Project2\ testing\IRV\randomWinner.txt  Project2\ testing\IRV\tenThousandVotes.txt | Election Table opens with candidates names, number of votes in each round, and votes added/subtracted for each round (data matches that of each test files respective audit file). | Election Table opens with candidates names, number of votes in each round, and votes added/subtracted for each round (data matches that of each test files respective audit file). | Pass |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |

**Post condition(s) for Test:**

These elections will be run as if this weren’t a test. The GUI still runs and a user can view the audit file.

|  |  |
| --- | --- |
| **Project Name: Project 2: Voting System Team# 14** | |
| **Test Stage: Unit x System \_\_** | **Test Date: 12/09/18** |
| **Test Case ID#: 17** | **Name(s) of Testers: Cassandra Chanthamontry** |
| **Test Description:** This is a manual test to manually test that in the terminal only prompts for the filename when running the program with no gui | |
| **Automated: yes \_\_ no \_x\_** | **Indicate where are you storing the tests (what file) and the name of the method/functions being used.**  Stored in: Project2/testing/FilenamePrompt  Functions: runElectionCommandLine  in Project2\ main\src\MariahEP.java |
| **Results: Pass \_x\_\_ Fail\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * Run the program with no gui flag * Pass election’s file absolute path when prompted | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | Running the program on the command line, the terminal only prompts for the file’s name | N/A | Filename is prompted and results from the filename is accurate | Filename is prompted and results from the filename is accurate | Pass |

**Post condition(s) for Test:**

After test is run, the file name prompt in the command line is verified.