|  |  |
| --- | --- |
| **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: 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: 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 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: TestIRV.java  Functions: testRunElection()  Helper Functions: initializeTestIRV() |
| **Results: Pass x\_\_\_ Fail\_\_\_\_\_\_\_\_** |  |
|  |  |
| **Preconditions for Test:**   * testBallot arraylist of arraylist<integer> is initialized * IRV Voting System is initialized | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step**  **#** | **Test Step**  **Description** | **Test**  **Data** | **Expected**  **Result** | **Actual**  **Result** | **Notes** |
| 1 | initializeTestIRV() | Arraylist<ArrayList<Integer>> of ballots, String[] candidates, | Initialized IRV Voting System | Initialized IRV Voting System | Initializer |
| 2 | testRunElection() – tests the only public method in IRV class | IRV Voting System | Winner = “Sasuke” | Function returned winner that == “Sasuke” | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

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

After test is run, the public method RunElection is verified to work and since this method calls upon and uses private methods those are verified to work as well

|  |  |
| --- | --- |
| **Project Name: Project 1: Voting System Team#14** | |
| **Test Stage: Unit \_\_ 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<mailto:example@gmail.com> | 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 \_\_ 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: The project #, name of your system, and the team#

Test Stage: Indicate whether it is a unit test or a system test.

Test Date: The date the test was performed.

Test Case ID#: A unique ID is required. Decide on a naming convention and use numbering. Example: Ballot\_Shuffle\_1

Name(s) of Testers: List the names of anyone involved in running this test case.

Test Description: Describe briefly the test objective.

Automated: Indicate if the test is completely automated or being checked manually. (If you have methods running the tests and checking results, select “yes”. If you are manually checking results, indicate manual by selecting the “no.”)

**Results:** Indicate if the test passed or failed.

**Step #:** You will be listing the test steps in order. This number is the step number in the process.

**Test Step Description:** Details of the test step.

**Test Data:** What the test data will be for this step. Be clear on what the input data will be. If using a specific file, be clear on the name.

**Expected Result:** What result are you expecting from the program component or system.

**Actual Result:** What result were returned based on the test.

**Post condition for Test:** What will be true after the test has been run? Has the state of the system changed in any way?

**Notes:** Comments and notesfor you and your team members.