**SER 216 – Summer B 2018 – Group 3**

Test Case Document

Revision 1.0

August 01, 2018

Document Number: TC.GD.041011.1.0

Team Name: Team 3

Project Sponsor: ASU SER218 Stratego

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Open up the project in Eclipse and run the Server.java file in the ./src/edu.asu.stratego/ directory. | The file runs successfully without any errors, and the server object should start, confirmed by the following message in the console:  Server started @ 192.168.0.137  Waiting for incoming connections...  (The IP address values can be different as long as they stay consistent throughout.) |
| 2 | Verify that the server is actually working by opening the command prompt/terminal and pinging the IP address displayed in the result of step 1. | The ping result should produce the final following output:  Ping statistics for 192.168.0.137:  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss) |

**FUNCTIONAL VERIFICATION TESTING**

**TEST CASE 1: Test to verify that the Stratego Server starts successfully.**

* Method of Testing: Manual
* IDE: Eclipse Oxygen 3
* Feature / Requirement:

**TEST CASE 2: Test to verify that the Stratego Client for Player 1 logs into the server session successfully.**

* Method of Testing: Manual
* IDE: Eclipse Oxygen 3
* Feature / Requirement: The server must be started before running the following test.

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Run the Client.java file in the ./src/edu.asu.stratego/ directory. | The file runs successfully without any errors, and the connection scene GUI should be displayed asking for the player’s nickname and the server’s IP address. |
| 2 | Enter the player’s nickname and the server’s IP address, and then click on the ‘Enter Battlefield’ button. | The GUI accepts the data and loads a waiting scene GUI displaying the following message:  Waiting for an opponent… |
|  |  | The following message is displayed in the console confirming that the server acknowledged player 1 joining the session:  Session 1: Player 1 has joined the session |

**TEST CASE 3: Test to verify that the Stratego Client for Player 2 logs into the server session successfully.**

* Method of Testing: Manual
* IDE: Eclipse Oxygen 3
* Feature / Requirement: The server must be started and player 1 should be logged in before running the following test.

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Run again the Client.java file in the ./src/edu.asu.stratego/ directory. | The file runs successfully without any errors, and the connection scene GUI should be displayed asking for the player’s nickname and the server’s IP address. |
| 2 | Enter the player’s nickname and the server’s IP address, and then click on the ‘Enter Battlefield’ button. | The GUI accepts the data and loads player 2’s board setup GUI.  (this output will be expected only if a player 1 was loaded before executing this test) |
|  |  | The following message is displayed in the console confirming that the server acknowledged player 2 joining the session:  Session 1: Player 2 has joined the session |
|  |  | Player 1’s waiting scene GUI disappears, and its setup GUI loads up instead.  (this output will be expected only if a player 1 was loaded before executing this test) |
|  |  |  |
|  |  |  |

**TEST CASE 4: Test to verify that the Setup GUI automatically sets up the boards and loads the game after 5 minutes.**

* Method of Testing: Manual
* IDE: Eclipse Oxygen 3
* Feature / Requirement: A server session has been started and 2 players have joined it

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | If the players do not setup the board themselves within 5 minutes of entering the setup GUI, the program will randomically set it up for them. | Both setup GUIs will disappear, and the game board GUI will be displayed with all piece properly setup. |

**TEST CASE 5: Test to verify that the Setup GUI allows the user to properly setup the board when within the 5 minutes time limit.**

* Method of Testing: Manual
* IDE: Eclipse Oxygen 3
* Feature / Requirement: A server session has been started and 2 players have joined it

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Click on one of the piece identified by a number. | The piece gets highlighted. |
| 2 | Move across the layout of the board to place the highlighted piece and ensure the squares’ board turns green or red based on the possibility of placing a piece in that position. | The squares’ board turns green or red according to the placement rules for the game of Stratego (the squares around the lakes should be highlight red for all pieces during setup). |
| 3 | Click on a green-highlighted square to place a piece. | The following results should be expected simultaneously:   * The piece gets positioned in the selected square; * The piece’s count should decrease by one. |
| 4 | Click on a different piece and position it in the same location as a previously placed one. | The following results should be expected simultaneously:   * The piece gets positioned in the selected square; * The piece’s count should decrease by one; * The replaced piece’s count should increase by one. |
| 5 | Position the last component of a specific piece. | The following results should be expected simultaneously:   * The piece gets positioned in the selected square; * The piece’s count should go to zero; * The piece representation of it in the top setup area turns gray/shaded. |
| 6 | Complete the setup for one of the players | The setup GUIs will display a “Waiting for opponent…” message until the other player completes the setup. |
| 7 | Complete the setup for all players. | Both setup GUIs will disappear, and the game board GUI will be displayed with all piece setup. |

**TEST CASE 6: Test to verify that the Game GUI works according to the game’s rules and the code specifications.**

* Method of Testing: Manual
* IDE: Eclipse Oxygen 3
* Feature / Requirement: A server session has been started and 2 players have joined it and setup the respective board.

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Hover over one of the pieces of the player that needs to perform a move to see if it can be moved. | * If the piece can be moved, the square the piece is located on will be highlighted green; * If the piece is cannot be moved, the square the piece is located on will be highlighted red. |
| 2 | Click on one of the pieces of the player that needs to perform a move. | * If the piece can be moved, the game will highlight the squares the piece can be moved to depending on the piece-type; * If the piece is could be moved, but it is stuck behind others and cannot perform a move, the square the piece is located on will be highlighted red; * If the piece is one of those that can never be moved and it’s in a position that looks like it could move, the game will not show any possible move and not allow the player to move it. |
| 3 | Hover and/or click on one of the pieces of the player that does not needs to perform a move. | The square the piece is located on will be highlighted red for any piece. |
| 4 | Pick one of the pieces that can perform a move and move it. | The following results should be expected:   * The piece moves to the selected square; * An arrow of the player’s color (red/blue) will show the move performed; * Both players boards will be updated; * The turn passes to the other player. |
| 5 | Position a piece into to perform an attack and attack. | The following results should be expected:   * The GUIs update showing the pieces’ type on both players’ GUI; * The attack gets executed and the following updates take place:  1. The winning piece moves to the loosing piece’s position, if the winner is the attacked, or it stays in its position, if the winning piece is the defender; 2. The loosing piece gets taken out of the game and disappears; 3. If the result of the attach is a tie, both pieces get taken out of the game and disappear. |
| 6 | Try to perform a move with a Bomb. | No move will be allowed because bombs cannot be moved. |
| 7 | Try to perform a move with a Flag. | No move will be allowed because the flag cannot be moved. |
| 8 | Perform enough moves to catch the opponent’s flag. | The game GUI updates showing all remaining pieces of both colors on both players’ GUI, and no more moves will be allowed. The game ends. |

**TEST CASE 7: Test to verify that the program handles the closing of either player’s window.**

* Method of Testing: Manual
* IDE: Eclipse Oxygen 3
* Feature / Requirement: A server session has been started and 2 players have joined it and are setting up, or have completed the setup of, the respective board.

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Close either player’s GUI clicking on the ‘X’ closing the Java window. | The following error handling message gets displayer in the console:  Session 1: Error occurred during network I/O  (The error is handled in the ServerGameManager.java file). |

**UNIT/INTEGRATION TESTING**

**TEST CASE 8: Test to verify that player color is properly inferred from opponent color.**

* Method of Testing: TestNG
* IDE: Eclipse Oxygen 3
* Feature / Requirement: Two player objects have been initialized.

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Close either player’s GUI clicking on the ‘X’ closing the Java window. | The following error handling message gets displayer in the console:  Session 1: Error occurred during network I/O  (The error is handled in the ServerGameManager.java file). |

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Set a player as the opponent |  |
| 2 | Set opponent piececolor to BLUE |  |
| 3 | Call waitForOpponent() |  |
| 4 | Call getColor() for player |  |
| 5 | Compare player color to expected color | getColor() returns the player color as RED |

**TEST CASE 9: Test to verify that player’s turn defaults with no move selected.**

* Method of Testing: TestNG
* IDE: Eclipse Oxygen 3
* Feature / Requirement: Two player objects have been initialized and GameStatus is in progress.

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Set a player as the player |  |
| 2 | Set player pieceColor to RED |  |
| 3 | setMoveColor to RED |  |
| 4 | Call playGame() |  |
| 5 | Call getStatus() on player |  |
| 6 | Compare expected status to move status of player | getStatus() returns the move status of player as NONE\_SELECTED |

**TEST CASE 10: Test to verify that a move is in bounds.**

* Method of Testing: TestNG
* IDE: Eclipse Oxygen 3
* Feature / Requirement: n/a

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Call isInBounds() with parameters (1,9) | isInBounds returns true |

**TEST CASE 11: Test to verify that a move is out of bounds.**

* Method of Testing: TestNG
* IDE: Eclipse Oxygen 3
* Feature / Requirement: n/a

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Call isInBounds() with parameters (1,10) | isInBounds returns false |

**TEST CASE 12: Test to verify that a specified square on board is a lake square.**

* Method of Testing: TestNG
* IDE: Eclipse Oxygen 3
* Feature / Requirement: n/a

|  |  |  |
| --- | --- | --- |
| **S.N** | **EXECUTION STEPS** | **EXPECTED RESULTS** |
| 1 | Call isLake() with parameters (4,7) | isLake returns true |

Aside from the proper execution of the game and the proper coding standards addressed by the bugs list and the related tools used to discover them, there isn’t any more testing required since no initial specification was provided regarding speed/performance testing and cross-compatibility between operating systems (especially since the game runs on Eclipse, and the tests would be testing Eclipse performance and cross- compatibility rather than the Game’s).