Test Manual

Battleship Game

By: Aaron Zachariah

Contents

Testing Scope

The tests for this application aim to test all functionality provided and ensure that the correct effect is generated given some correct input. These tests must also show that the application is able to handle incorrect input and will fail with an appropriate error message or exception. Consequently, given some invalid input, the application must fail safely, and alert the user with an informative message via the GUI or the terminal.

Testing Strategy

There is a variety of user input and features that can either function correctly or cause an error. To make sure that the application handles each case in the correct manner, there must be a concrete strategy on the various aspects of the program execution that must be tested.

User Input

Input validation is important for this application, since initialization and network configuration relies on getting useful user input. Each aspect of user input must be tested, first to ensure that the application reacts normally when given valid input, and also to ensure that an appropriate error is throws when given faulty input. See the section on “Test Scenarios” for more information.

Game Functions

The game offers several functions to the user that allows the user to play the game, but these functions may not always be enabled during every step of the game. Certain functions should be disabled or enabled based on the state of the game. For each function, there should be normal functionality for the correct states, and no functionality for the incorrect states. Thus, both cases must be tested for each function.

Testing Scenarios/Test Cases

Below is a section detailing several types of testing scenarios, their input, and their expected result. These testing scenarios aim to test all the features of the Battleship application, and test various types of valid and invalid input. When testing certain aspects of user input, make sure all other aspects of user input is correct, to prevent certain problems from covering other problems. These scenarios cover the UI (View), as the Model is tested separately. Please see the java files for testing in the /test/java directory

Configuration Input

To start the game, there is user input that must happen so the application can configure what player the user is, and how the user will connect to another user.

Connect/Host

First the program asks the user whether they want to connect or host a game. Below are test cases regarding input for this prompt.

Test Cases:

1. Test system behavior when “c” is entered
2. Test system behavior when “h” is entered
3. Test system behavior when anything else is entered

Connecting as Host

If you choose host, you must enter the port value

Test Cases:

1. Test system behavior when a valid port value is entered
2. Test system behavior when an invalid port value is entered
3. Test system behavior when nothing is entered
4. Test system behavior when a non-integer value is entered

Connecting as Client

If you choose to connect to a host, you must enter the address and a port value separated by a colon

Test Cases:

1. Test system behavior when a valid address and port is entered
2. Test system behavior when a valid address and port is not separated by a colon
3. Test system behavior when an invalid address and valid port is provided
4. Test system behavior when a valid address and invalid port is provided
5. Test system behavior when neither address nor port are valid
6. Test system behavior when only the address is provided

7. Test system behavior when only the port is provided
8. Test system behavior when nothing is provided

Button Functionality

When the main window appears, there will be a tool bar with useful buttons. These buttons may only be meant to function only when the game has reached a certain state.

Ready Button

This button is used to let the game know you are done setting your board

Test Cases:

1. Test system behavior when ready is clicked in the initial state
2. Test system behavior when ready is clicked during player1's turn
3. Test system behavior when ready is clicked during plaer2's turn

Randomize

This button will re-randomize the game board

Test Cases:

1. Test system behavior when randomize is clicked in the initial state
2. Test system behavior when randomize is clicked during player1's turn
3. Test system behavior when randomize is clicked during player2's turn
4. Test system behavior when randomize is clicked after ready

Fire

This button allows the user to fire at the enemy board by giving coordinates

Test Cases:

1. Test system behavior when fire is clicked during the initial state
2. Test system behavior when fire is clicked during the player's turn
3. Test system behavior when fire is clicked during the other player's turn
4. Test system behavior when correct coordinated are input for fire
5. Test system behavior when the coordinate values are swapped
6. Test system behavior when nothing is input as coordinates
7. Test system behavior when the letter input is lowercase
8. Test system behavior when the letter input is uppercase
9. Test system behavior when the input is too long
10. Test system behavior when the input is too short

Endgame Stage Testing

When a game is finished, the scoreboard will change to include the winner's point, and each user is prompted with the option to replay the game.

Scoreboard

A Text label to display the current score, where each game won is worth one point.

Test Cases:

1. Test system behavior when player1 wins
2. Test system behavior when player2 wins

Replay Prompt

When the game finishes, the user will be asked if they wish to replay the game

Test Cases:

1. Test system behavior when the user types "n"
2. Test system behavior when the user types "y"
3. Test system behavior when the user types invalid input
4. Test system behavior when one user types "y" and one types "n"
5. Test system behavior when both users type "y"
6. Test system behavior when both users type "n"