Testing Manual for BattleShip

Introduction

This document provides detailed instructions for testing the BattlleShip The goal is to ensure that all features work as expected and the application is free of defects.

Test Strategy for `BattleShip`

Objectives

The main objectives of the testing strategy are to:

- Ensure the `GameOfLifeApplication` meets its functional requirements.
- Verify the application behaves correctly under various conditions and edge cases.
- Validate the user interface for usability and accessibility.
- Confirm the application performs efficiently under different load conditions.

Test Environment

- **Operating System**: Windows 10, macOS, or Linux
- **Java Version**: JDK 11 or later
- **JavaFX Version**: 11 or later

Setup Instructions

1. **Install Java and JavaFX**:
    - Ensure you have Java Development Kit (JDK) installed. You can download it from here.
    - Install JavaFX SDK from here.
2. **Compile the Application**:
    - Make sure you are in the correct directory:

      Something like so:

      src/edu/rpi/cs/csci4963/u24/wangn4/hw03/battleship/

```
javac *.java
```

Run the Application:

```
java --module-path /path/to/javafx-sdk/lib --add-modules
javafx.controls,javafx.fxml
```

```
edu.rpi.cs.csci4963.u24.wangn4.hw03.battleship.BattleShipApplication
```

Note: edit the `/path/to/javafx-sdk/lib` with your path to the JavaFx sdk

Test Cases:

Functional Tests:

Application Launch:

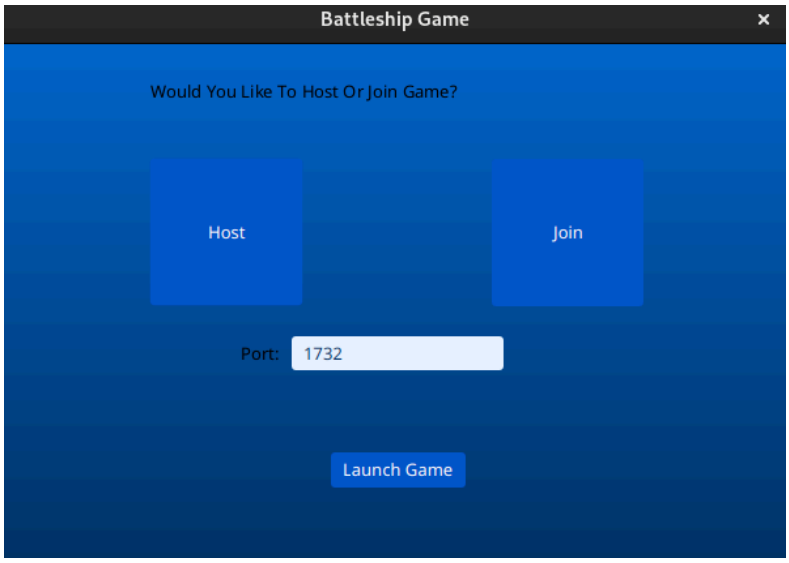
- **Test Case**: Launch the BattleShippplication.
- **Steps**:
    1. Open a terminal or command prompt.
    2. Navigate to the directory containing the compiled classes.
    3. Run the application using the command provided in the setup instructions.
- **Expected Result**: The application window should open without errors.

  Main window should look like so:

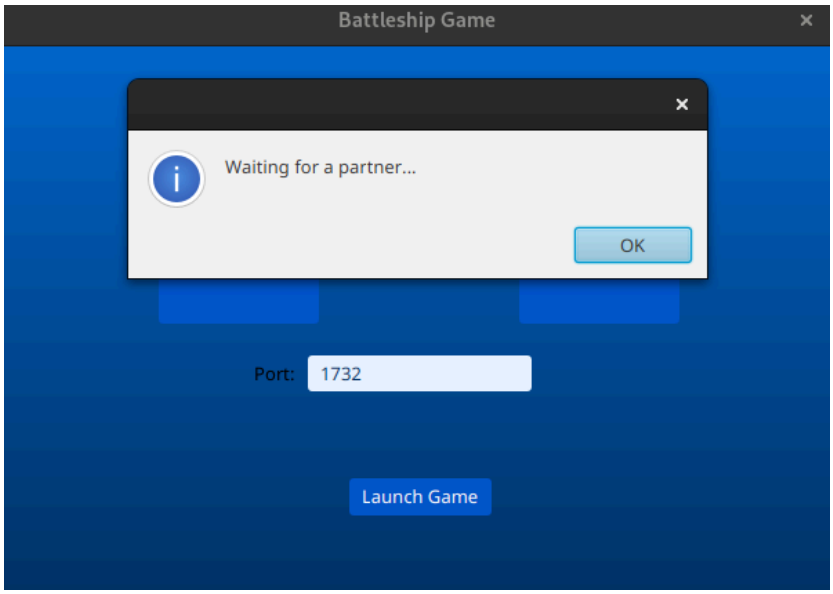
Host:

Test Case: Click the host button.

Expected Result:



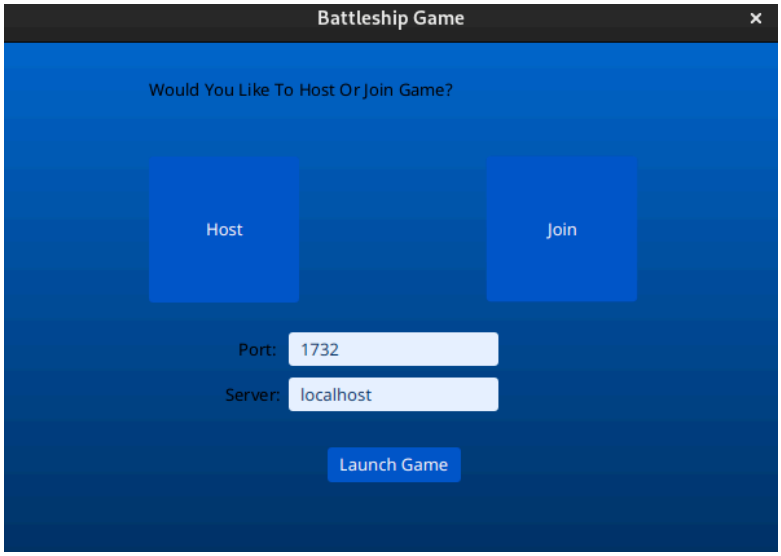
Test Case Host (continued): Click Launch Game
Expected Result:



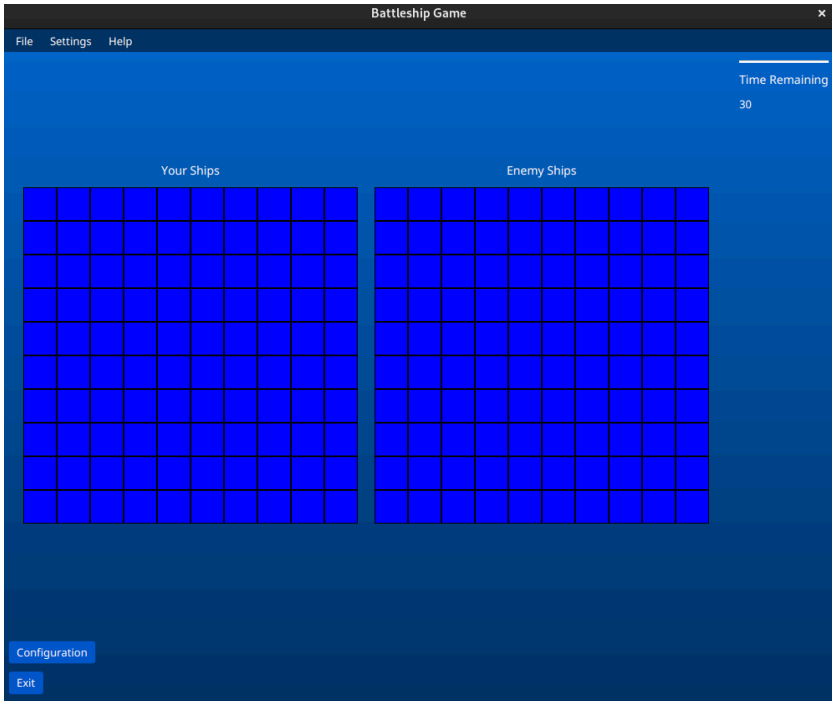
Join Game:

Test Case: In a second window, open the game and Click Join

Expected Result:



Test Case Join Game (continued): Click Launch Game
Expected Result: Both players should see this



Exit Functionality:

Test Case: Verify the functionality of Exit in menu.

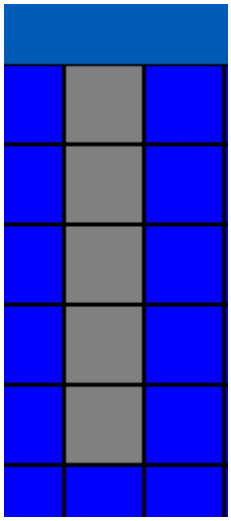
Steps: Click Exit located in the Menu -> File

Expected Result: Program closes successfully.

Place Ships:

Test Case: Place a ship by clicking on Your Ships

Expected for Left Click:



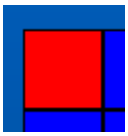
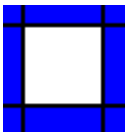
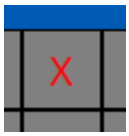
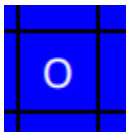
Expected for Right Click:

Place Shot:

Test Case 1: Take a shot by left clicking on enemy grid.

Test Case 2: Get shot at by the enemy clicking on their enemy grid.

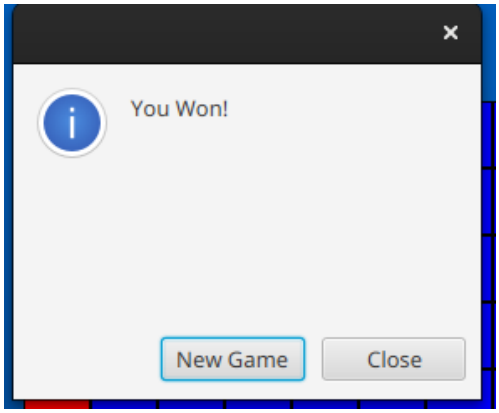
Expected Results:

			
Red - Your Shot Hit	**White - Your Shot Miss**	**X - Enemy Hit You**	**O - Enemy Missed You**

Game End:

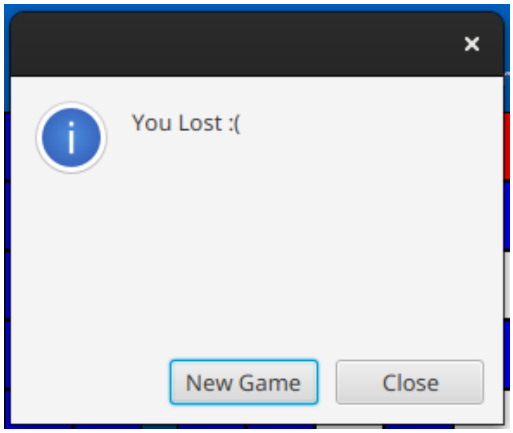
Test Case 1: Player 1 Sinks all ships by shooting

Expected Result:



Test Case 2: Player 2 Got all their ships

Expected Result:



J-Unit Test Cases:

J-Unit test cases are located in the BattleShipModelTests.java located in the src -> test file path.

Testing Suite Description:

As part of the comprehensive testing suite for the Battleship game, several key tests were conducted to ensure thorough validation of the application's functionality and robustness.

Firstly, boundary value testing was performed on the grid size and ship placement logic to verify that the game handles edge cases correctly. This included tests for placing ships on the edges or corners of the grid and attempting to place ships partially out of bounds, both horizontally and vertically, ensuring proper error handling and user feedback.

Secondly, concurrency tests were implemented to simulate multiple players interacting with the game simultaneously, particularly focusing on network-related functionalities such as hosting and joining games. These tests helped identify potential synchronization issues or race conditions that could arise during real-time gameplay.

Thirdly, usability tests were designed to evaluate the user interface, ensuring it was intuitive and accessible. These tests verified that all interactive elements, such as buttons and menus, responded correctly to user actions and that the game provided appropriate visual and auditory feedback for different in-game events (e.g., ship hits, misses, and game end conditions).

Additionally, performance tests were conducted to measure the game's responsiveness and stability under varying load conditions, such as a large number of simultaneous players or extended gameplay sessions. These tests ensured that the game maintained an optimal performance level without lag or crashes.

Lastly, tests for verifying the correctness of game rules and mechanics were included. These tests ensured that ships could only be placed in valid positions, that the game correctly tracked hits and misses, and that the win/loss conditions were accurately determined. Complete game scenarios were simulated to ensure that the game logic functioned as intended from start to finish.

By incorporating these tests, the testing suite for the Battleship game was comprehensive, helping to identify and resolve potential issues and ensuring a robust and enjoyable user experience.