BattleShip Game Testing Manual

1 Test Plan

The underlying data representation will be mainly tested by unit tests. For the GUI part, the focus is on verifying whether data bindings are working correctly and if target behaviors are reached.

2 Test Strategy

2.1 Unit Test

A Java framework, JUnit5, has been used within the process. You may consult `com.luox6.battleship.test` for the written test cases.

2.1.1 Testing Scenarios

Coordinate Coordinate represents a 2D point on the plane:

- Correctly represent 2D point
- Ability to convert from string representation
- Equality implementation

Line Line represents a straight line consisting of two coordinates:

- Correctly represent a line with direction (horizontal or vertical)
- Overlap helpers (between line)
- IsBetween helpers (point on the line)
- Coordinate sequence
- Correctly calculate line length

Cell Cell represents a single cell on the map:

- Correctly represent Coordinate with discoverable attribute
- Ability to set ship ID and status

GameMap GameMap represents the status of cells:

- Row/col length getters
- Class constructor
- Ability to mark cells
- Detect ships' statuses (alive or dead)
- Validate coordinate input

The above tests the model class's construction, mutations, and expected string representation.

Run Unit Test You may use your favorite IDE integration with JUnit. Please consult the JUnit manual for more details. However, if you would like to use `ConsoleLauncher`, after compiling the program following the instructions in `readme.md`, try the following command under the project root directory for unit test results:

```
$ sh scripts/test.sh
```

Or for Windows platform:

```
.\scripts\test.cmd
```

Please consult `test-log.txt` for the complete log.

2.2 Manual Test

Unit tests cannot cover some corner cases, such as:

- Network Protocol
- IO/Input validation
- GUI functionality
- Other scenarios

2.2.1 Testing Scenarios

Server Client Init

- Player name check
- Report error if failed to connect
- Client successfully syncs settings from server
- Server will wait for incoming connections

Before Game Ready

- Successfully set ships following instructions on the dialogs
- Report ready status when clicking the ready button
- Report error if user tries editing ships after ready
- Report error if ships are out of the map or overlap with existing ships
- Report info if all ships are set

Game Play

- Turn remaining time updated every second
- Game ends when one side is out of the given time
- Mark ship on self/enemy map each time a player makes a move
- Game ends when all ships are marked
- Warn user if tried making a move during enemy turn

Notice:

- ± 2 seconds of display time difference is acceptable
- Please dismiss any dialog as quickly as possible during moves

Game Ended

- Prompt correct win/lose message
- Ability to restart the game and increase score accordingly

Notice:

- Scoreboard will update if both sides accept the next game

Toolbar

- `Ready` button will set the game state correctly
- `Setting` button will show warning and follow the config panel

Configuration

- Setting persistence (after application closed)
- Setting update will be effective after restarting the program
- Cell color changes accordingly
- Time limits will sync with the client even though local settings are different

Warnings

- At any time, text fields that expect a number but fail to parse will give a warning
- At any time, if the action of a text field fails, the value will be restored to its previous value