Tests

The following methods are used when testing the program.

Test plan

The underlying data representation will be mainly tested by the unit test. For the GUI part, it is mostly focused on whether the data bindings are working, or if target behaviors are reached.

Test strategy

Unit Test

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

Testing scenarios

Coordinate Coordinate represents a 2D point on the plane

- correctly represent 2D point
- be able 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
- be able to set ship id and status

GameMap GameMap represents the status of cells

- row/col length getters
- class constructor
- be able to mark cells
- detect ships' statuses (alive or dead)
- validate coordinate input

The above have tested model class's construction, mutations, and expected string representation.

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

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

or windows platform

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

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

Manual Test

However, the unit test cannot cover some corner cases, e.g.:

- Network Protocol
- IO/Input validation
- GUI functionality
- …

Testing scenarios

Server Client Init

- player name check
- report error if you failed to connect
- client successfully sync setting from server
- server will wait for incoming connection

Before game ready

- successfully set ship following instructions on the dialogs
- report ready status when clicking ready button
- report error when user tried editing ship after ready
- report error if ship will out of map or overlap with existing ship
- report info if all ship are set

Game play

- turn remaining time updated every second
- game ended when one side is out of given time
- mark ship on self/enemy map each time a player make a move
- game ended when all ships are marked
- warn user if tried making move during enemy turn

Notice:

- +-2 seconds of display time difference is acceptable

- please dismiss any dialog as quick as possible during moves

Game ended

- prompt correct win/lose message
- be able to restart game and increase score accordingly

Notice:

- score board will update if both sides accept next game

Toolbar

- `Ready` button will set game state correctly
- `Setting` button will show warning and follow 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 client even though local settings are different

Warnings

- at any time, textfields that expected number but failed to parse will give warning
- at any time, if the action of textfield failed, the value will be restored to its previous value