“Wiki”

The below sections contain what our wiki should contain.

Updated user stories for Final Submission

1. As a player, I want the system to keep track of where my ships are.
2. As a player, I want to be able to choose a coordinate and for the system to tell me whether I hit a ship or missed it.
3. As a player, I want the system to tell me when my ship has sunk.
4. As a player, I want to be able to use the sonar pulse in place of a turn twice in a game, once I have sunk at least one of the opponent’s ships.
5. As a player, if I hit the captain’s quarters of one of the opponent’s ships, I want the whole ship to sink, unless the quarters is armored, in which case I want the ship to sink if I hit it twice.
6. As a player, I want to be able to use the space lazer once my enemy’s ship has sunk.
7. As a player, I want to be able to move my fleet of ships to any direction (N,S,E or W) and have the option to undo this move as many times as I want.
8. As a player, I want to know if I’ve won the game.
9. As a player, I want to be able to place a mine on my opponent's board before they place their ships.
10. As a player, I want to be able to continue playing even if my battleship is sunk.
11. As a player, I want to be able to place my ships using inputs on the command line.
12. As a player, I want to be able to play battleship using inputs on the command line.

Planning game/CRC card iterations

We made a few changes to our CRC cards as suggested to us in our previous interview grading session. Here are our new CRC cards.

Player	
Initialize player grid Keep track of player grid Keep track of player ships Keep track of Sonar pulses used	Grid Ship Coordinate

Grid	
Handle adding ships to grid Keep track of the status of	Ship Coordinate

each ship Handle hitting ships Handle printing the grid Handle sonar pulse Able to move fleet one unit	

Ship	
Size, type, location Pieces hit Sunk/not sunk	Coordinate

Coordinate	
Hold a coordinate on an x-y axis Hold a value that indicates whether the ship at that coordinate is hit or not Hold a value that indicates whether piece is captain's quarters or not	Ship Direction

Game	
Handle turn-taking Take user input Keep track of players	Ship Coordinate Command Grid Direction Player

Direction	
Hold cardinal direction to move fleet Move coordinates different directions Undo moves	Command Coordinate

Weapon	
Hold name and availability of weapon	Game

Command (interface)	
Define interface for command (moves)	Game

MoveFleet	
Implementation of Command; move fleet in a single direction Undo a move	Command Direction

Time estimates and time spent

We estimated that creating all of our classes and testing them would take around 4 hours total.
Milestone 2: It took a bit longer than that, about 5 or so hours across all of our workloads.
Milestone 3**:** It took 2 hours combined (Tuesday's time allotted) and 2 hours per person to complete the requirements of this milestone.
Milestone 4: It took about 4-5 hours longer than expected to complete. 5-6 hours of work in group along with at least 2 hours each to complete.
Milestone 5: It took a shorter amount of time to complete this milestone as we already have a good framework. We expected to take around 5 hours to complete, but instead took only 2 hours.
Final Submission: This was the most time consuming part of the project. The I/O in the main function was more involved than we expected, and the final documentation took up a decent amount of time. We expected everything to take about 7-8 hours to complete but it really took 12 hours.

Project risks

We weren't totally sure about what this part of the rubric meant, so we went with "risks" in terms of the extreme programming reading. Using the Risk Index from here, we sort our user stories by risk. Since the base requirements of the game are fairly straightforward, we only identified *low* and *medium* levels of risk based on this index.

Low (0-1)

1. (0) As a player, I want to be able to choose a coordinate and for the system to tell me whether I hit a ship or missed it.
    a. Completeness: 0
    b. Volatility: 0
    c. Complexity: 0
2. (1) As a player, I want the system to tell me when my ship has sunk.
    a. Completeness: 0

    b. Volatility: 0
    c. Complexity: 1
3. (1) As a player, I want to know if I've won the game.
    a. Completeness: 0
    b. Volatility: 0
    c. Complexity: 1
4. (1) As a player, I want to be able to place a mine on my opponent's board before they place their ships.
    a. Completeness: 0
    b. Volatility: 0
    c. Complexity: 0

Medium (2-4)

5. (2) As a player, I want the system to keep track of where my ships are.
    a. Completeness: 0
    b. Volatility: 0
    c. Complexity: 2
6. (3) As a player, if I hit the captain's quarters of one of the opponent's ships, I want the whole ship to sink, unless the quarters is armored, in which case I want the ship to sink if I hit it twice.
    a. Completeness: 0
    b. Volatility: 1
    c. Complexity: 2
7. (3) As a player, I want to be able to use the space lazer once my enemy's ship has sunk.
    a. Completeness: 1
    b. Volatility: 0
    c. Complexity: 2
8. (3) As a player, I want to be able to continue playing even if my battleship is sunk.
    a. Completeness: 0
    b. Volatility: 0
    c. Complexity: 2
9. (3) As a player, I want to be able to use the sonar pulse in place of a turn twice in a game, once I have sunk at least one of the opponent's ships.
    a. Completeness: 0
    b. Volatility: 1
    c. Complexity: 2
10. (3) As a player, I want to be able to place my ships using inputs on the command line.
    a. Completeness: 0
    b. Volatility: 1
    c. Complexity: 2
11. (4) As a player, I want to be able to move my fleet of ships to any direction (N,S,E or W) and have the option to undo this move as many times as I want.
    a. Completeness: 0
    b. Volatility: 2

    c. Complexity: 2
12. (4) As a player, I want to be able to play battleship using inputs on the command line.
    a. Completeness: 0
    b. Volatility: 2
    c. Complexity: 2

Meeting minutes

10th Feb 2021
Meeting with TA. Following suggestions:
- Add to Log class - Responsibility - Determine game status (i.e. winner / loser)
- Add a new class for the abilities players can use in their turn - “Abilities” - Heal and Sonar included in that

16th Feb 2021
Implemented following tests using JUnit:
- All tiles in grid EMPTY at initialization
- Name, length and sunk (boolean) of ship

TODO:
- Implement set coordinates of ship - John
- Each subtype of ship and ask Dwight about expectation from milestone 2- Sagarika
- Implement Player class - Filip
- Implement Main class - Elizabeth

23rd Feb 2021
Use cases:
- Place ships on their grid
- Take a turn
- Hit the captain’s quarters
- Sonar Pulse

Divided groups for HW2:
- Sagarika and Filip
- Elizabeth and John

24th Feb 2021
Milestone-2 meeting with TA:
Issues:
- Cannot implement the wiki (solution suggested: update in Documents folder in git)
- Test coverage - coordinates testing issue

2nd Mar 2021

During meeting:

- Merge branch that has captain's quarters functionality into master
- Planning sonar
- Update Wiki
- Testing - write tests for new features: captain's quarters and sonar pulse
- Implementation

TODO:

- Clarify about captain's quarters
- Add tests - sonar
- Add captain's quarters and sonar
- Update wiki
- Add tag

9th March 2021

Meeting with TA about milestone3, following comments:

- Decompose ship and make it scalable
- Main ship class to be abstract
- Coordinates to be in a single place - not hard coded in ship

Decided Ship implementation to follow the Factory design pattern, each subclass to implement the abstract class "Ship" to address the TA's comments

Following assumptions about the new requirements for milestone 4:

- The shape of the submarine to be in 2D, i.e - depth must remain the same throughout

16th March 2021

Command design pattern to be implemented to execute the move fleet, notes for the implementation:

- moveFleet's previous direction as the "button"
- Command to be an interface with two methods - 'execute' and 'undo'
- Each 4 directions could have been separate classes, to reduce repetitiveness, we will implemented a class direction and move according to the direction
- New "Direction" class would allow expansion

Following assumptions about the new requirements for milestone 4:

- If single ship cannot move because it is located on the edge, the whole fleet does not move
- Moving in any direction would mean moving one unit on the grid in that direction

- Requirements said ‘Undo’ can be used multiple times for any “move fleet” command, logically this would mean doing ‘undo’ twice would get you to the initial position, using trice would be the same as using it once and so on.

1st April 2021

Implemented a new mine feature, where the player can place a “mine” on the enemy’s grid prior to the placement of the ships. If the enemy places a ship on the mine tile, it will count as a hit.

Implemented a new lifeboat feature. If the player’s battleship is sunk, a 1x1 “lifeboat” is spawned at a random valid location on the grid.

Divided two new features:
- Filip: lifeboat
- John: mine

4th April 2021

Debugging “mine” tests and writing wiki.

20th April 2021

Starting implementing the main function. The goal of this meeting was to set up the i/o to be able to take in both player’s inputs for their names as well as the coordinates of the ships they want to place. We completed the name input functionality but only got about halfway done with the coordinate system.

22nd April 2021

Finished the main I/O implementation and added logic for winning the game (if one player has sunk all of the ships). We also started working on the documentation for the final submission.

23rd April 2021

Debugging I/O and adding sonar ability in the player input.

24th April 2021

Completed documentation for final submission. The documentation goes into depth on the development process, requirements and specifications, architecture and design, as well as personal reflections from each group member. We also added comments for each public class and method using the Javadoc standard.

25th April 2021

Last minute debugging of code, such as fixing any tests that may have failed due to new changes. We also finished adding comments for the public classes and methods, and updated the README and all the documents needed for the final submission.