Name: BinaryBuffs
Team Members: Sahand Setareh, Tahmina Ahmad, Divya Bhat, Tanvi Gopalabhatla

---

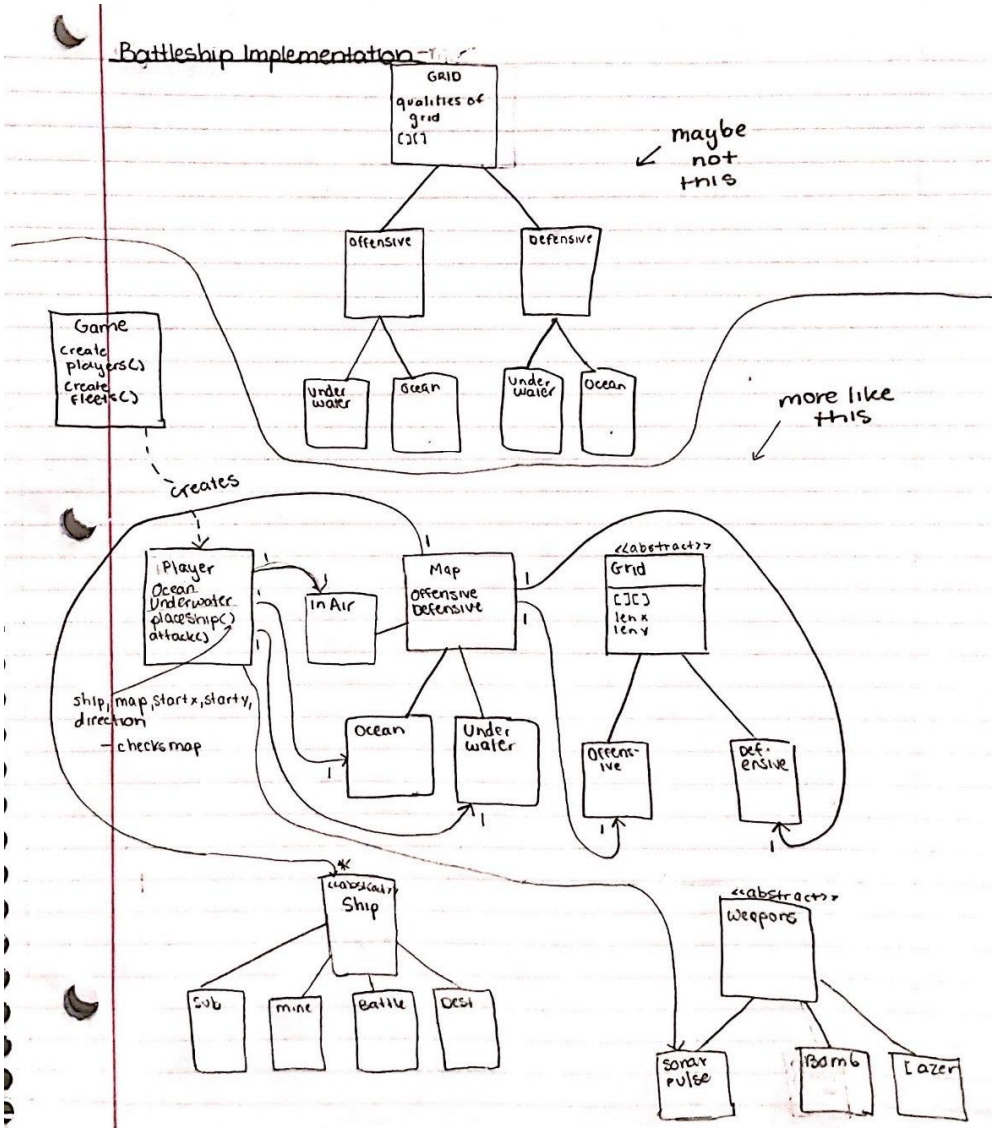
User stories

- Player Perspective: As a user of this game, I will have the capabilities of adding a submarine at the surface or below the surface, so that if I sink my opponents submarines, I would increase my chances of winning and if my submarines are hit by my opponent, my chances of winning decrease.
- Player Perspective: As a user of this game, once I sink at least one opponent ship, I will have the capabilities of using a new weapon called the space laser, and have the ability to penetrate the water and be able to hit both a surface ship and a sub that is placed below it at the same time, which will act as an attack against my opponent throughout the game.
- Player Perspective: As a user of this game, I should be able to move my fleet. When the command to move is given, each ship will move one position, depending on whether I choose N, S, W, or E to move them based on directions. I should be allowed to undo and redo moves, but I will not be allowed to move a ship if it is at the edge of my board. This will allow me to have more strategy and options when I'm playing the game, which will allow me to increase my chances of winning.
- Creator Perspective: In order to create this game through Java, as the developer, we would implement the appropriate classes and methods allowing us to simulate the Battleship game according to the requirements and features we were given.

Planning game

- Implement all aspects of move fleet feature
    - Wrote separate methods for multi-layer undoMove, redoMove, reverseMove, playerMoveFleet, and moveFleet
        - Used stacks as main data structure for implementation
- Refactoring code
    - Make code more extensible for additional ships, weapons, make it more loosely coupled
    - Implement Map abstract class
        - Inherited by ocean map and underwater map for a player
        - Keeps track of ships and locations
    - Extract Grid class
        - Objects for defensive and offensive grid
    - Separate out Ship subclasses

        - Implement Submarine ship which can be placed on either map
        - Created interfaces for armored and submersible ships (Decorator pattern)
    - Separate out Weapon subclasses
        - Create Bomb class
        - Implement Space Laser weapon
            - Create Space Laser class
    - Implement design principles
        - Encapsulate what varies principle
        - Extensibility
        - Strategy pattern
- Plan out functionality for Game class
    - User fleet creation, view menu of possible choices, etc.



Time estimates and actual time spent

- Estimate:
    - 4 hours a week
    - 10 hours total for this milestone
- Time Spent This Week:
    - Monday, March 15:
        - 2 hours
    - Wednesday, March 17:
        - 2 hours
    - Thursday, March 18:
        - 3 hours

    - Friday, March 19:
        - 2 hours
    - Saturday, March 20:
        - 12.5 hours
    - Sunday, March 21
        - 3 hours

Reevaluate project risks

- Goal: Iteration 4:
    - Reviewing expectations, tracking progress, and completing remaining coding portions of project, like the 3 additional required features
    - This is a checkpoint to see what we have gotten done and how closely we are following the initial expectations and requirements set during iteration 1, 2, and 3
    - We will be testing new features and working on other requirements given to us by the instructors for Milestone 4.
- Evaluation:
    - We got the 3 additional features from the milestone 4 writeup and then implemented them in our code. We had to add new attributes and methods in our classes as well as do major refactoring so that our code adhered to design principles.
    - Added strategy method for ship and method implementation. All of our code is now loosely coupled and we encapsulated what varied. Additionally, our code has been refactored to become easily extensible.
    - Goal for next iteration: From our original expectations, we will be:
        - Implement a GUI for our battleship game.
        - Wrapping up the rest of the project assuming that it is the last milestone/iteration.
        - We initially said we would do our final round of testing and wrap up our code. However, since we have been testing throughout and will continue to test as we implement new features, we may not need the final round of testing.

Meeting minutes

- Public Link: https://docs.google.com/document/d/1l77NK2v-QxKcIluM39wImUMLsDqf4aaz7iqc5WtMgxo/edit?usp=sharing

Resolved outstanding issues from last time:

- Completely reorganized and re-coded most classes in our project in order to prepare for additional features, a multi layer grid (underwater and above ground), new weapons, and new functionalities.
    - The ship coordinates are no longer baked into ship
    - Ships are abstracted → no longer embedded in ship class
- Code now abides by a design principle of being loosely coupled where our components have little or no knowledge of the definitions of other separate components.
- Created test classes for each of our major functionalities.
    - Each functionality that is incorporated into the program has its own corresponding test class, so that the tests aren't placed in a single class which causes confusion