Overview:
In our game's design, two of the core functionalities include dynamically placing forts on the game board and processing user moves. These actions are key to our Object-Oriented Design. The Game and FortManager classes are important in these processes.

FortManager class:
In the FortManager class, it maintains a list of Fort objects, the game board and the current Cell id. The relationship between FortManager and the game board is direct association. FortManager will “has-a” game board as member variable.

When a Game instance is created, it initializes a game board with a defined board size and a FortManager responsible for managing forts on the board. The Game class calls place/initialize forts method on the FortManager, passing the desired number of forts as determined by game configuration. FortManager will interact with the game board and update the status of placed fort.

Inside FortManager, the method/responsibility of the placing the forts on board, is to iterate over the number of forts to be placed, creating a new Fort instance for each. For each fort, we attempt to find a position on the game board. It randomly selects starting coordinates and checks if the fort can be placed without overlapping existing forts or extending beyond the board's limits. If a valid position is found, then we update the corresponding Cell objects on the game board to update the fort's placement, add them to the list, marking them as occupied and assigning a unique identifier.

FortManager is also keeping track of the list of alive forts, number of damages forts and if all forts have been destroyed, these are used to update the game score and status.

Game class:
In the Game class, it maintains the game board, the FortManager and the opponent’s score. The Game “has-a” game board and “has-a” FortManager, and it “uses” the text UI. When the game is initialized, the FortManager will be called and place the forts on the game board.

The Game class contains a method to check if the user's input represents a valid coordinate on the board. It ensures the input is correctly associated with the board's bounds when interact with text UI.

Game class is also responsible to check if the cell has been shot or not, which checks if the targeted Cell is part of a fort (isOccupied) and updates the game state accordingly.
If the cell was part of a fort, we mark the cell as hit and notifies the corresponding Fort via the FortManager to increment its damaged cells count. If the cell is hit, the Game class will update the opponent's total score and check if the fort is being destroyed after couple moves. The game continually checks the conditions for ending the game, either all forts are destroyed, or the opponent reaches the maximum score.