OOD Explanation

1) Random Fort Generation

Random fort generation is handled by the build() method of the FortMapBuilder class. The fort map is represented as a List<List<Field>> data structure where each Field object keeps track of who occupies the land (i.e. fort/opponent ID) and whether or not the field has been hit before.

Since the Field class takes care of updating the occupancy and hit status (int fortID and boolean hit values) of its field, the FortMapBuilder focuses solely on selecting which fields to assign to which fort/opponent.

The FortMapBuilder initially receives a blank map with no forts placed yet, and builds the fort map by randomly placing forts on the given map. In the process of placing the forts, the FortMapBuilder utilizes random field (row and column) selection and DFS algorithm to allow placement of purely random fort shapes in random areas of the map.

- Random field selection is used twice:
    1. when selecting the first block of a fort, and
    2. when selecting a neighboring field to extend the fort. This naturally allows any shape to be built and allows the same logic to be used for different fort sizes (other than 5 which was specified in this assignment).
- The DFS algorithm was used to traverse an unoccupied area and to ensure there was sufficient area available to place a new fort (by finding the maximum connected fields).

Additionally, the FortMapBuilder parametrizes the map configuration (# of forts, size of fort, map size) to allow for a highly flexible fort map generation functionality.

2) User Input Handling

Handling the user's move involves interaction between almost all classes in the program: GameUI, FortMap, Field, OpponentsPointsManager and Utility.

The GameUI class handles user input as part of the game loop. A private method promptNextMove() takes user input and performs validation, prompting for re-entry in case of invalid input, then returns the move as a String in the format of <letter><number>.

This valid move String is then passed to the OpponentsPointsManager object's method calculateResult(). Inside this method, the String is translated into row and column coordinates (of type int). The row letter is translated into int using the Utility class's alphaToInt() method.

The Field object at this coordinate on the map is grabbed. If this Field has not been previously hit, its boolean variable hit is set to true (changing the state of that Field), and the OpponentsPointsManager object updates the state of the opponent whose fort contains that Field (if it is occupied by an opponent fort), and updates the amount of points that the opponent can score per round, which is held in a List<Integer> inside the class.

The program proceeds to print the round result based on the updated opponent state held in the OpponentsPointsManager object.

3) Comments on other classes

OpponentsPointsManager class is responsible for storing, updating, and calculating points along with the knowledge of fortsize-to-point translation. FortMap class is responsible for providing a printable notation of the map aside from storing the map state. GameSystem class was removed after the process of refactoring the code due to a very superficial abstraction where many of the methods were simply a method call of OpponentPointsManager or FortMap objects. Utility class is used for the translation between the numeric fort ID and the alphabetic representation.