How Our OOD Will Work

Case One: Board Creation

1. When the game begins, the main class **FortressDefense** will create the I**nitializer** object with the user specified tank number argument (command-line argument 1) or the default tank number (5) and the board dimensions (10X10).
2. The **Initializer** object creates a **BoardMaker** object with the given game board information.
3. The **BoardMaker** object uses the game board information to create the **GameBoard** object. Then the BoardMaker uses a nested for-loop to create new **BoardCell** on the **GameBoard.** The coordinate values of each cell will be derived from the index values at which it is created (for rows we may want to use a map from integers to letters). All cells will be stored in a map belonging to the GameBoard, where the key is the coordinate of the cell.
4. When the basic **GameBoard** is completed, the **BoardMaker** will proceed to create tanks with the placeTanks() method. These tanks will be kept in an list, which is an attribute of the GameBoard. Tanks will be assigned ID numbers; the first tank will be "1", the second "2", and so on. Each tank will be initiated with 4 hitpoints. Then it will assigned a set of cell coordinates.

- When assigning coordinates, placeTanks() will first choose a random position on the board. If that position is occupied by a tank it will choose another random position (it if cannot find an empty position after n attempts it will give up), returning an error. After the first position is assigned, it will then choose a random spot of radius 1 from itself. If that spot is occupied or off the board, it will choose again (again, it will give up after a certain number of tries). It will also consider adjacent *corners*, but will only accept such a position if one of its components/cells *has a side adjacent to that position*.[1] For instance, in the diagram below, the sequence X1, X2, X3, X4 is possible. Once X3 has been placed, the position of X4 is a possible option. First, it is corner-adjacent to the position of X3, so it may randomly come up. Second, it is side-adjacent to the position of X2, so if it does come up randomly, it will be selected.

X1			
X2	X4		
X3			

- placeTanks() will continue in this fashion until 4 cells have been selected. Whenever a cell is selected, the BoardMaker will set its isTankCell as true.

1

5. Once all tanks have been placed, the **BoardMaker** will pass the **GameBoard** back to the **Initializer** which will use it as a parameter when creating the **MoveProcesser**.

Case Two: Player Move

1. When the user inputs a move, the **UI** will first pass the value to **MoveProcessor'**s calculateMoveResult(). the method will then call the moveChecker() in the **GameBoard** object to check if the move is illegal (i.e., if it is not a **GameBoard** coordinate).

- If the move is illegal, calculateMoveResult() will return an immutable constant indicating an illegal move, which will cause the **UI** to request that the user enter another move.
- If the move is legal, the value of the move will be passed to the **GameBoard's** checkForHit() method. checkForHit() will check if the value is a key in its map of **BoardCells**. It will have that cell call setHasFiredAt() to set the value of hasFiredAt to 'true'.
- Next, checkforHit() calls the cell's isTankCell() method. If isTankCell() returns true, checkForHit will find the tank that has the coordinate of the hit cell in its myCells list. If no such tank exists, it will return an error. Otherwise, it will call the tank's decreaseDamage() method and set isMyLivingCell() to 'false'. Finally, checkForHit() will return 'true' to calculateMoveResult().
- If isTankCell returns false, checkForHit() will simply return 'true' to calculateMoveResult().

2. Finally, if calculateMoveResult() received 'true' it will in turn return an immutable string constant indicating a hit to the **UI**. The **UI,** in turn, will print "Hit" on a line in the console. Otherwise, it will return an immutable string constant indicating a miss to the **UI**. The **UI**, in turn, will print "Miss" on a line in the console.