How will the board be drawn?

- Our board will be made using a double array or 2-D array of **Cell** type.
- It will be first created with the help of our **GameBoard** class and printed using our **GameTextUI** class.
- Our double array will represent our board in the form of a ten by ten grid of **cells.**
- Our **GameBoard** class, we also consist of a list of all the tanks on board and also a Hashmap mapping our **Tanks** to their position on the **gameboard.**
- Our **GameTextUI** after validating user command will call **GameBoard** to help manipulate a cell instance (takeHit)
- As with a **Game Board** class we can check at a specific position for **Tanks**, otherwise manipulate the empty **cell**.
- Our **cell** class also tells us if a specific **tank** belongs to those particular **cells**, which would help us calculate a **Tank** hit taken based on its **cell** damage. (using **isHit() [Method in Cell Class]**)
- Some cases to consider for isHit to represent cell object :
    - If isHit returns false, i.e. it was not hit by the player, we will print '~', which is fog.
    - If isHit returns true, i.e. it was hit during the turn by the player, then will consider another two cases :
        - Was there a **Tank** present on the **cell t**hat has taken a hit?
        - If YES
            - We will print 'X'
        - If NO
            - We will print ' '

  For cheat mode, we will also assign each **Tank** object a letter.
- If cheat mode is enabled,
    - If isHit returns false, i.e. it was not hit by the user, and there is no **Tank** present, we will print '.' to identify the cell object.
    - If isHit returns false, i.e. it was not hit by the user, and there is a **Tank** present, we will print the letter identifying the tank using our **getTankIdentifier()** method.
    - If isHit returns true, i.e. that the player hit it and there is a **Tank** present, we will print 'X' to identify the damaged **Cell** object of that particular **tank.**

How will the turns work?

- During the user's turn, the user will input a **Cell** coordinate he expects to hit.
- The coordinates will be parsed and validated by **Cell[][].**
- If the coordinates are out of bounds, the user will be asked to give another input.
- Once the user selects an appropriate coordinate, the **fortress** will attempt to hit that **cell**.
- Once it is determined that an actual **tank** has been hit, the **tank's** health will be reduced by 1 in the **removeCell()** function with the specific coordinate and will be set to **tankIdentifier** (lowercase).
- Otherwise, if the user hits a **cell** where there is no tank, it results in a miss.

- Every turn, **GameBoard** is updated with the changes.
- Based on the **tank**'s current health each turn, the **tank**'s damage is revised (**calculateDamage() [Method in Tank Class]**) to reflect these changes where max health has max damage.
- Once a **Tank** loses all its **cells** ( when **damageTaken == tankCells.length() [Method in Tank Class]** ), it will be deemed inactive and will no longer participate in the game.
- As the game progresses, the **fortress**'s health will eventually deteriorate, if the player does not manage to eliminate all **tanks** before they hit 0 the game is lost.
- Otherwise, the user wins and all hit cells are displayed by **GameTextUI** with the appropriate **tank tag** in lowercase (Tank A cells when hit will be converted to an 'a', Tank B to 'b' and so on)