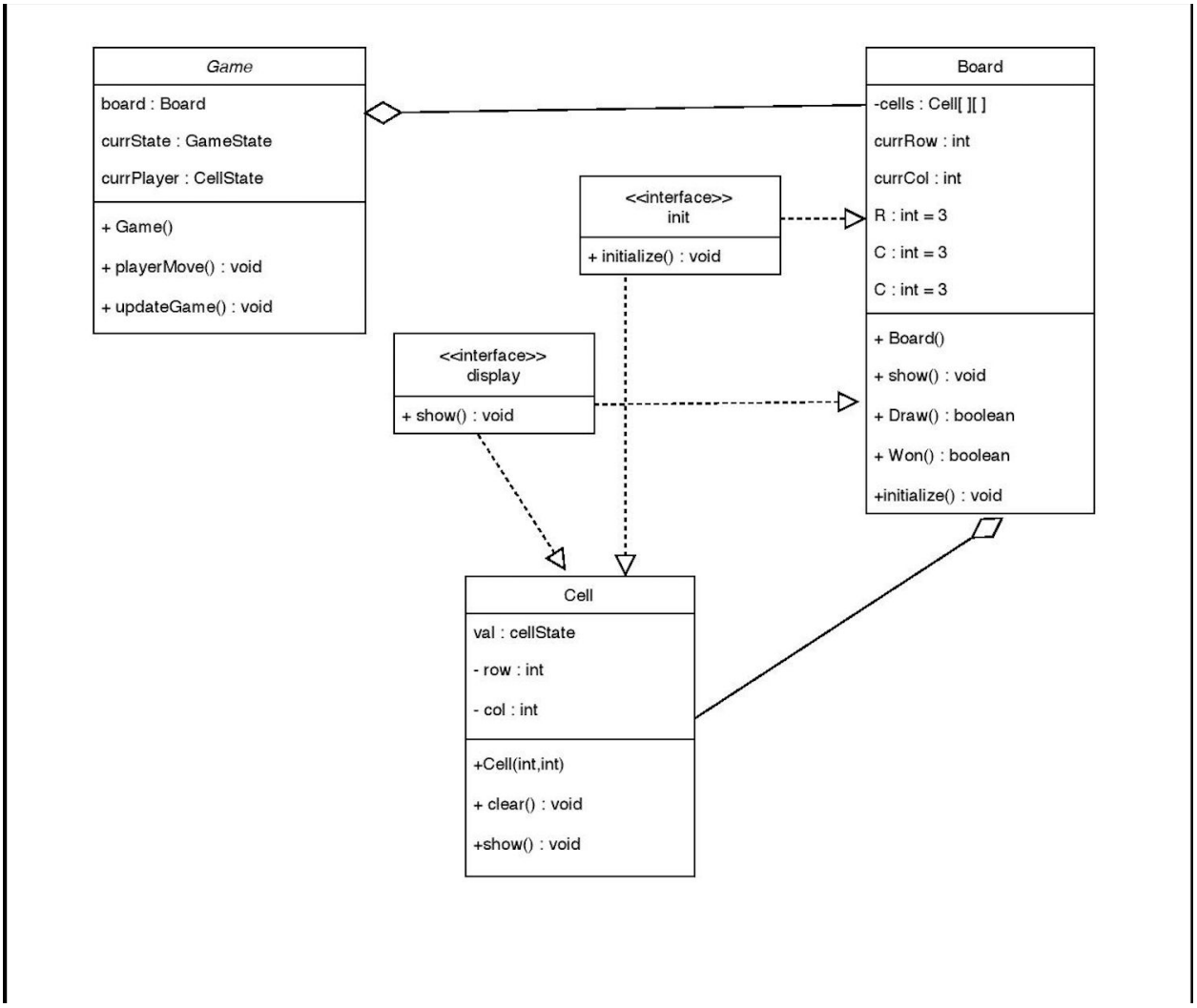
Design Document

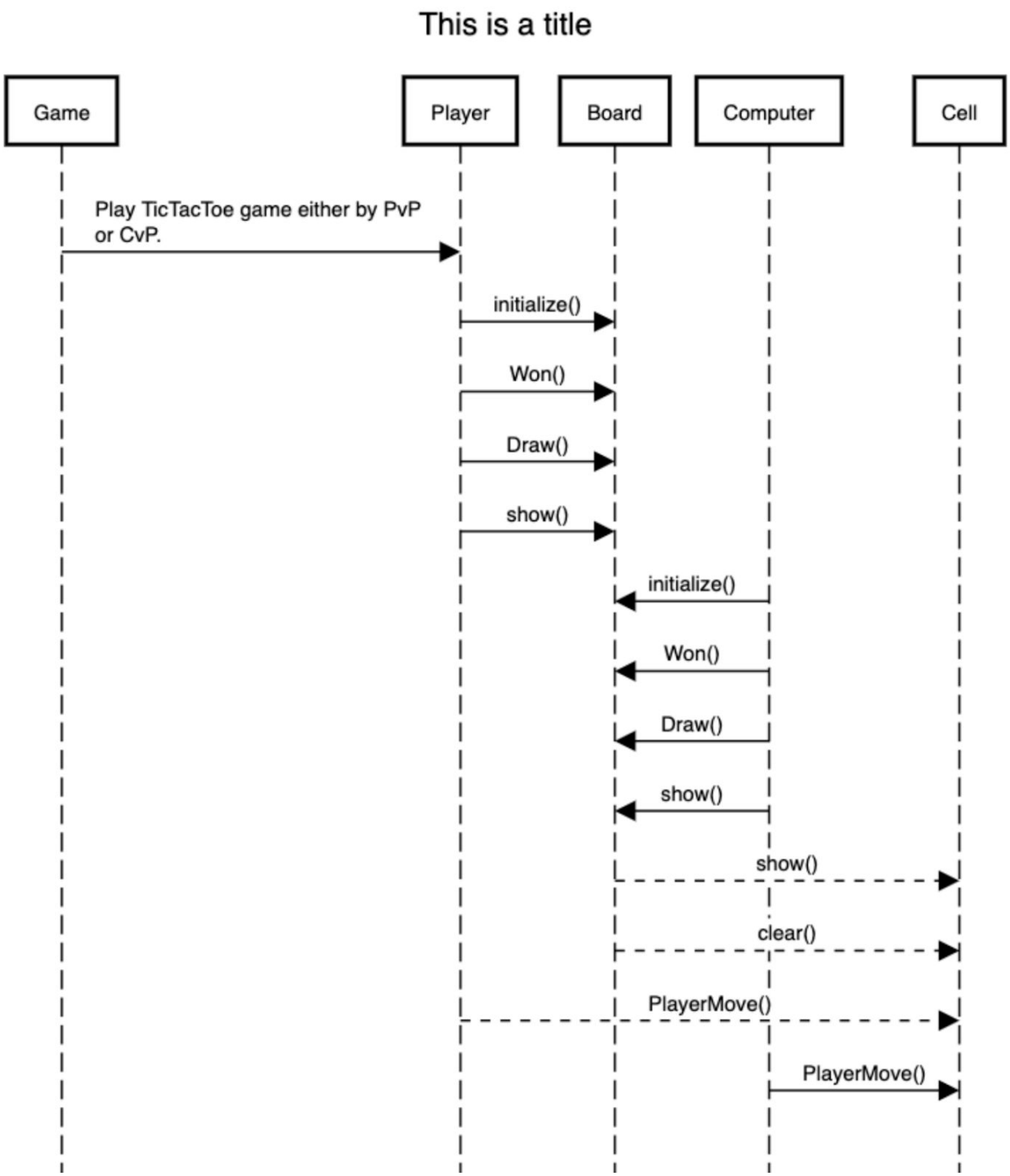
Name: Shreyansh Chaudhary
Emp ID: 138913

Section I: First Iteration Game Project Design

Part I: Class Diagram



Sequence Diagram:



Part II: Design decisions and assumptions

My main focus was on 2 things: Broader scope and loose coupling. I tried in such a way that the code is more general such that people can choose whichever grid they want to play, the dimensions and so on, without making any change in the code or without using different classes for different grids.
The player will first choose the grid he wants to play on like square or hexagon. Then he will decide the dimension of the grid.
The player also gets a choice to revert back his chance and also to see the leaderboard stats.
Assumption: In the 9*9 or any other n*n grid, one assumption which is used is that once the basic grid is won by some player that all the empty spaces of that grid is filled with X's or O's accordingly to avoid false chances.

Part III:

GameDesign v2.0: Requirement I

Completed : 1.Tic-Tac-Toe consists of 3x3 Square Cells.
-Basic 2D array is used for storing the chances. Size of the array is dynamic. Matching criteria used is for loop to check the winner.
Completed : 2.Game between 2 humans.
- 2 types of players are involved. Their chance is updated using "enum".
Completed : 3.Game between human and machine.
- Machine's choice is generated using random number generation keeping check of the boundary conditions.

Completed : 4.Winning Criteria - 3 Cells in Row/Column/Diagonal are in same State.
- Used for loop for finding the winner so that it can be extended easily for bigger dimensions.
Completed : 5.Announce Winning Player.
- Enums are used to check the winning player and stop the game as soon as someone wins.

GameDesign v2.0: Requirement II
Completed : 6.Enhanced TicTacToe Game Consist of 9x9 Sqs.
- Already incorporated. Just changed the grid size to 9x9. Base grid size remains as it is(can be changed too).
Not Completed : 7.Enhanced Tic-Tac-Toe will continue to expand in depth levels…
Completed : 8.Extend game to 4x4 board.
- Already incorporated. Just changed the base grid size to 4x4.
Completed : 9.Human player is biased.
- Added the functionality to undo the chance.
Completed : 10.Storing and retrieving game states.
- Leaderboard class is implemented that will handle all the related stuff.

GameDesign v2.0: Requirement III
Completed : 12.Super Tic-Tac-Toe Game Extends Enhanced Tic-Tac-Toe Game…
- Design separate class for Hexagon board game and mapped the indices accordingly.
Completed : 13.Design Winning and Losing Criterias On All Edges...

- Total 3 winning possibilities of left-right and both the diagonals.

Not Completed : 14.Incorporate Irregular shaped Hexagonal Boards

GameDesign v2.0: Requirement IV

Completed : 15.Incorporate biased game boards

- Added Revert back the move option.

Not Completed : 16.Incorporate Connect Four in the game

Completed : 17.Refactor and Reuse the code in both the games

- Reused the game in both games which is evident from the interfaces as well as the Board and Cell classes are the same.

Section II - Same as Section I

Section III - How to run/test your code

```
create an empty Package directory called "learnJava"
run command          - mkdir learnJava
                          - cd learnJava

create an empty directory  within learnJava called "ClassFiles"
run command          - mkdir ClassFiles
```

return to the directory where downloaded
run command - cd ..

Compile file
run command - javac f<number>.java -d ClassFiles

Run file
run command - java -cp ClassFiles/ learnJava.f<number>

where number = number of feature in the dev branch

Can I Run Test.java to test your whole source code?
Yes. Rather run f5.java. It contains everything required.

Are you providing all Input/Output files to run Test Code using Test.java?
No. The I/O is via terminal (real time).

----------xxxxxxxxxxxxxxx----------
-