# ***DESIGN DOCUMENT***

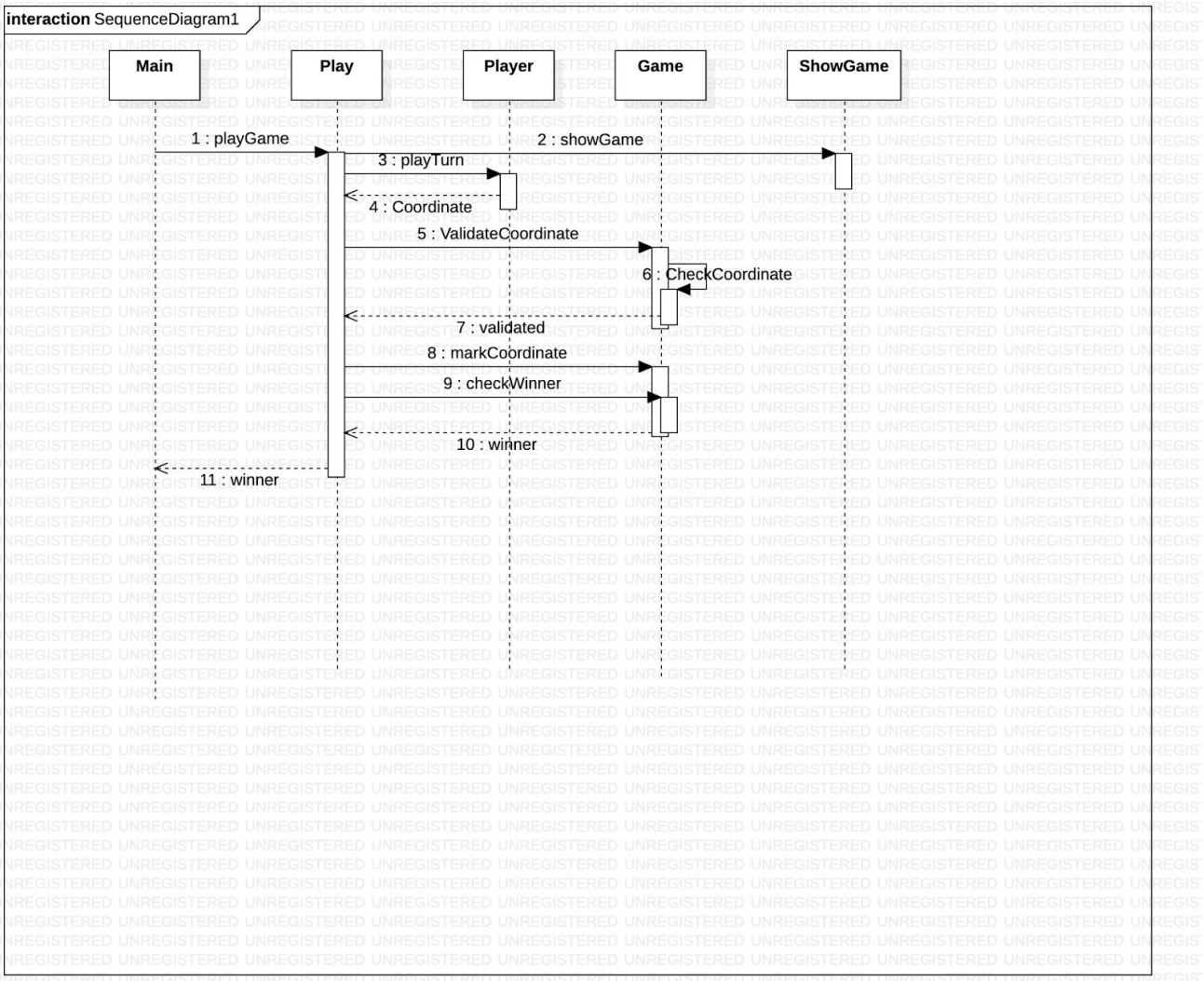
PROJECT GAME DESIGN

SECTION I: First Iteration Game Project Design

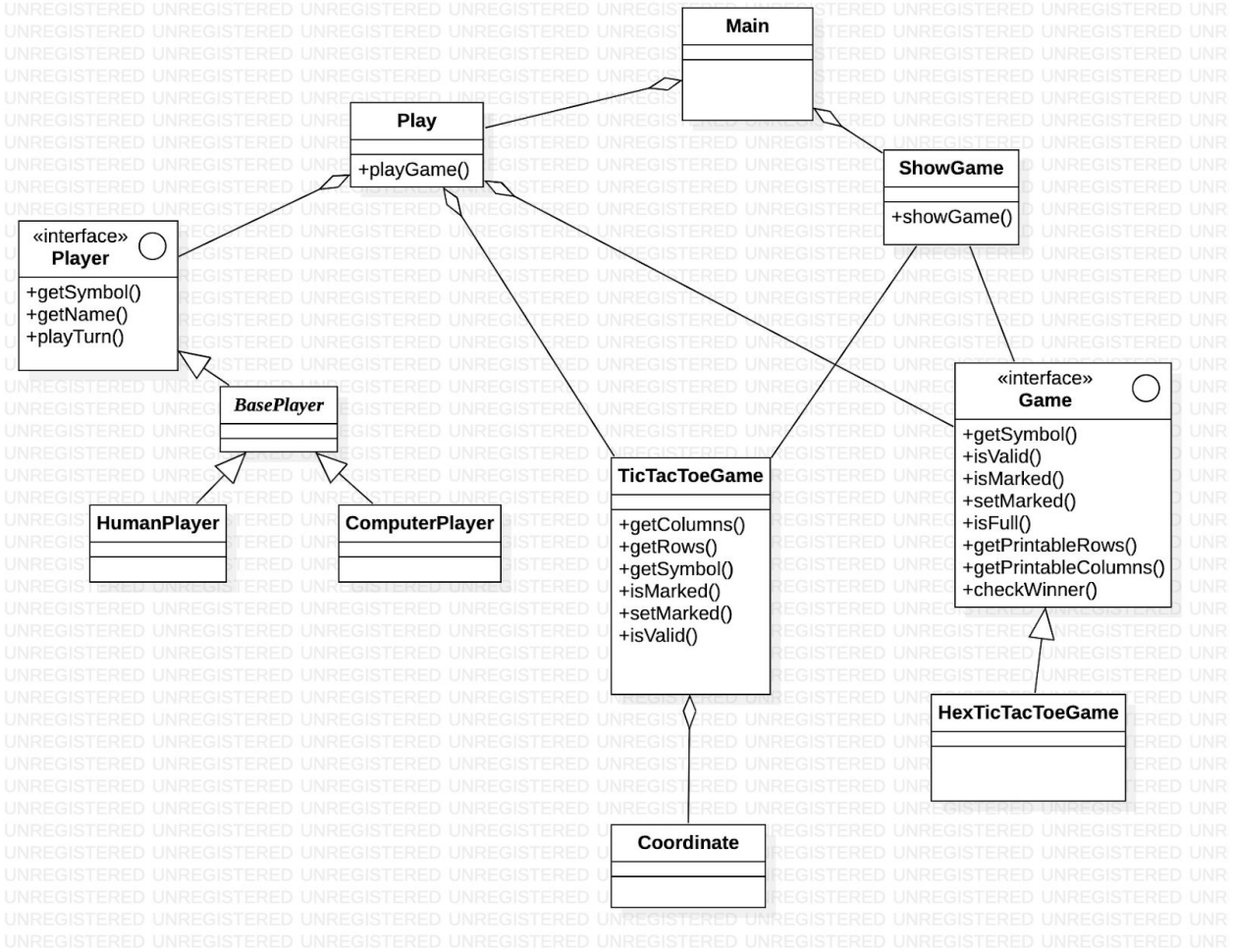
[Till Phase III or Phase IV Submitted Upto 29th January, 2020]

PART I: Design Diagrams

Sequence Diagrams



Class Diagrams




PART II : Common Design/Choices and Conventions/Assumptions and Detailed Descriptions etc.

Design Choices

1) The game can be extended to any number of rows and columns.
2) The game can be made any number of levels deep.
3) Games can be played by any number of players. We just have to pass a list of players to Play class. By default only two players are passed.
4) Player is the interface which lists methods that have to be implemented by all players.
5) Game interface provides methods for all games although in this version only hex

tictoegame implements Game interface. It is made to provide common methods for all board games and in the next version rectangular tic tac toe game will also implement game interface.

PART III : Feature Specific Design/Choices and Conventions/Assumptions

GameDesign v2.0 - Requirement I

Completed - 1. Tic-Tac-Toe consists of 3x3 Square Cells

Used 2D array of Coordinates to represent the game grid the grid size and how much level deep game should be can be specified by the user.

Completed - 2. Game Between Two Humans

A list of players is passed to Play class for playing. To simulate a game between two humans, a list containing two human players can be passed.

Completed - 3. Game Between Human and Machine

Machine uses random selection of coordinates but the machine move can be optimised later.

Completed - 4. Winning Criteria - 3 Cells in Row/Column/Diagonal are in the Same State.

Checked each row, column, diagonal to check whether they are in the same state.

Completed - 5. Announce Winning Player

GameDesign v2.0 - Requirement II

Completed - 6. Enhanced Tic-Tac-Toe Game Consist of 9x9 Squares...

Used a mathematical model to map coordinates to various subgames

Completed - 7. Enhanced Tic-Tac-Toe will continue to expand in depth levels...

Completed - 8. Extend Game to 4x4 Board

Feature Specific Design Decision?

Not Completed - 10. Storing and Retrieving Game State

Feature Specific Design Decision?

Not Completed - 11. Store Players Game Statistics: Leaderboard

Feature Specific Design Decision?

GameDesign v3.0 - Requirement III

Not Completed - 12. Super Tic-Tac-Toe Game Extends Enhanced Tic-Tac-Toe Game...

Completed - 13. Design Winning and Losing Criterias On All Edges...

All edges are checked for winning condition.

Not Completed - 14. Incorporate Irregular shaped Hexagonal Boards

Feature Specific Design Decision?

GameDesign v4.0 - Requirement IV

Not Completed - 15. Incorporate Biased Game Board

Not Completed - 16.Incorporate Connect Four Game In Design

Partially Completed - 17. Discover Newer Abstract Types

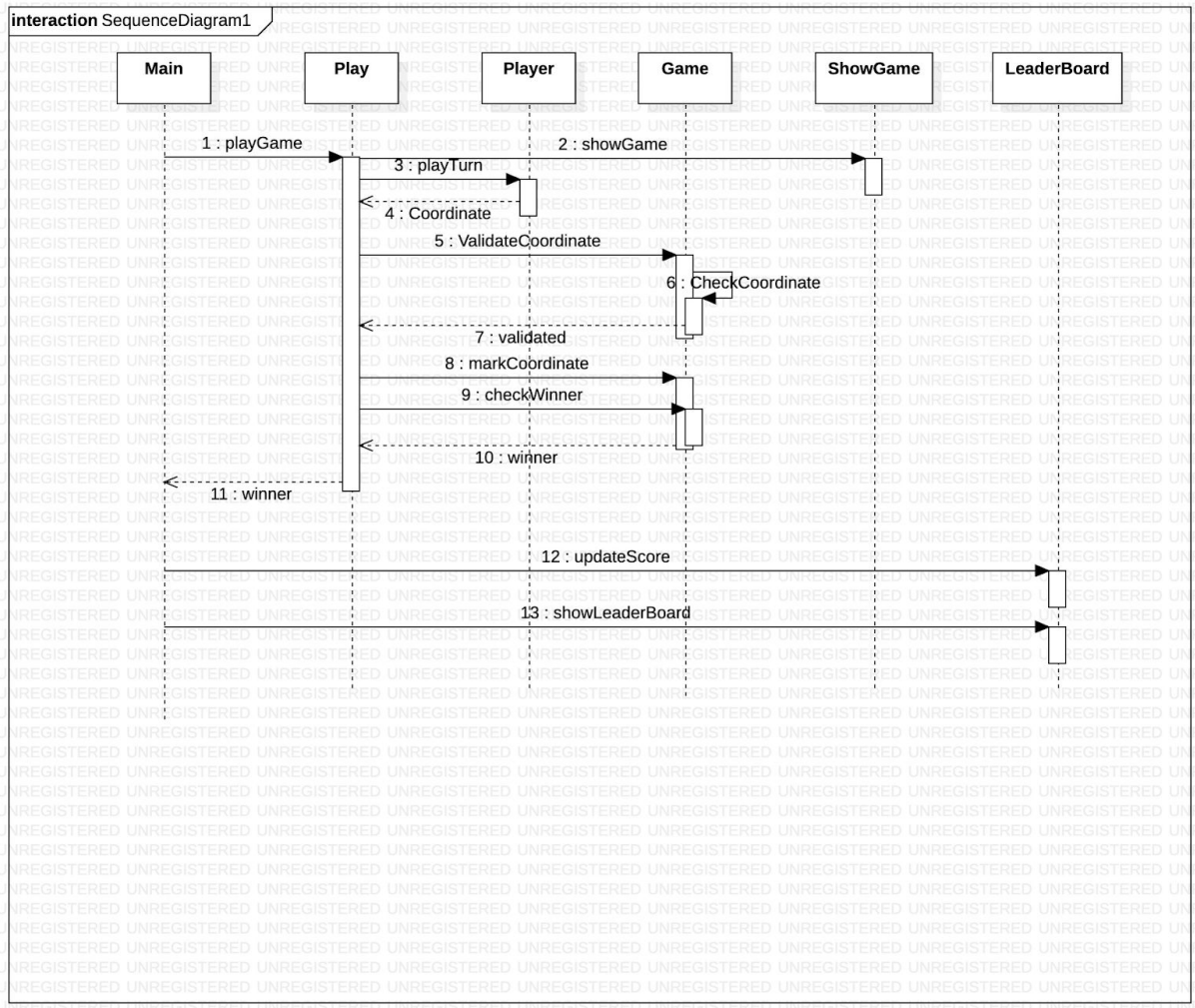
Partially Completed- 18. Refactor and Reuse Code In Both Games

SECTION II: Second Iteration[Refactoring/Redesign] Game Project Design

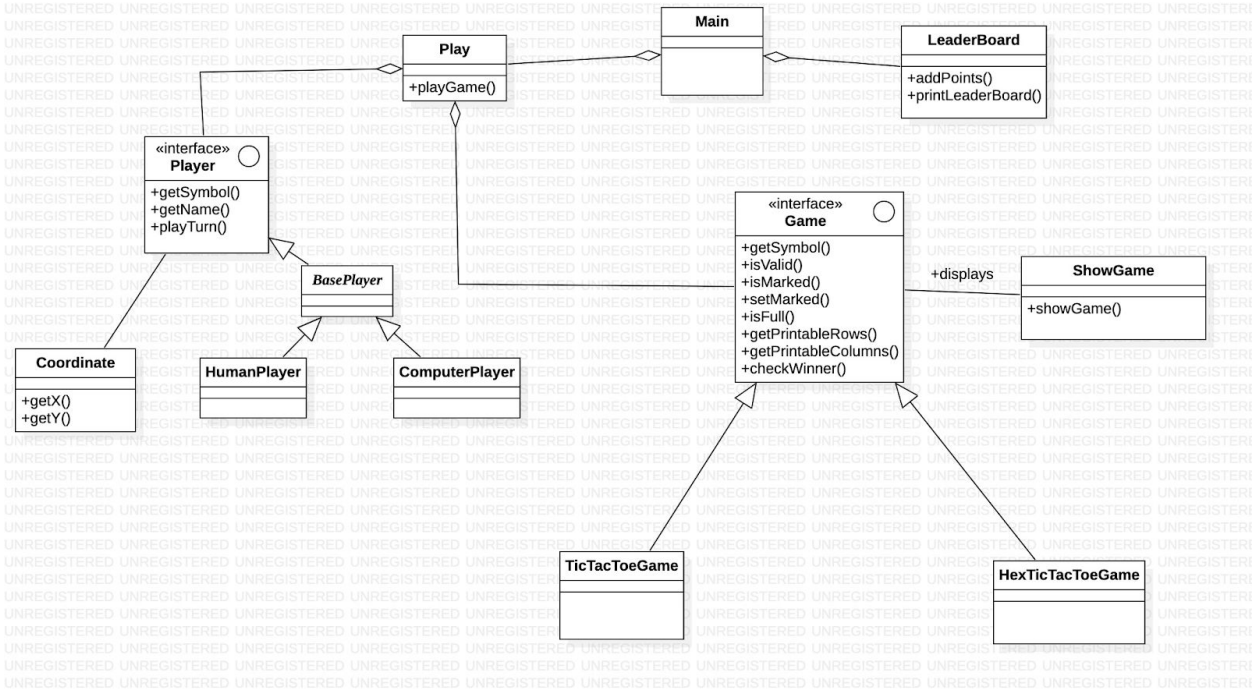
[Till Phase III or Phase IV Submitted Upto 03rd February, 2020]

PART I : Common Design/Choices, Conventions and Assumptions

Sequence Diagrams




Class Diagrams




PART II : Common Design/Choices and Conventions/Assumptions and Detailed Descriptions etc.

Game interface provides common methods for both rectangular TicTacToe and Hexagonal TicTacToe games.
The LeaderBoard is added to keep track of scores of various players.
Code is reused in TicTacToe and HexTicTacToe most of the code is the same.
Both HexTicTacToe and Rectangular TicTacToe are designed in a way that they can be extended in multiple ways in the future.
The operations which can be performed on a game are well encapsulated in Game class and internal implementation of game board is hidden from outside only provided methods can be used to read or modify the state of the game.
Play class is used to carry out a single game between two players and returns the result to the Main class. The main class then updates the leaderboard and decides whether to play the game again or not calling the playgame method of Play class in the former case.

PART III: Feature Specific Design/Choices, Conventions and Assumptions

GameDesign v2.0 - Requirement I

Completed - 1. Tic-Tac-Toe consists of 3x3 Square Cells

Completed - 2. Game Between Two Humans

Same as section 1.

Completed - 3. Game Between Human and Machine

Implemented HumanPlayer and MachinePlayer classes each class overriding playTurn() method of the base class.

Completed - 4. Winning Criteria - 3 Cells in Row/Column/Diagonal are in the Same State.

Completed - 5. Announce Winning Player

After each move call is made to check the winner method to check whether the player has won or not.

GameDesign v2.0 - Requirement II

Completed - 6. Enhanced Tic-Tac-Toe Game Consist of 9x9 Squares...

Implemented using game consisting of subgames each basic grid point also acts as a subgame of dimension 1*1.

Completed - 7. Enhanced Tic-Tac-Toe will continue to expand in depth levels...

Each game contains subgames and thus can be expanded to any number of depth levels.

Completed - 8. Extend Game to 4x4 Board

The game is implemented internally to have dynamic grid size thus can be extended to 4x4, 5x5 etc.

Not Completed - 10. Storing and Retrieving Game State

Completed- 11. Store Players Game Statistics: Leaderboard

LeaderBoard is maintained as a separate class thus can be customised in the future. It can be further extended to display the leaderboard in various ways without affecting any other classes. LeaderBoard class expects the winning player and no of points and stores the points and displays the leaderboard.

GameDesign v3.0 - Requirement III

Completed - 12. Super Tic-Tac-Toe Game Extends Enhanced Tic-Tac-Toe Game...

Exactly the same as Super Tic-Tac-Toe just the logic to calculate winner and little other changes are there.

Completed - 13. Design Winning and Losing Criterias On All Edges...

Not Completed - 14. Incorporate Irregular shaped Hexagonal Boards

GameDesign v4.0 - Requirement IV

Not Completed - 15. Incorporate Biased Game Board
Not Completed - 16.Incorporate Connect Four Game In Design
Completed - 17. Discover Newer Abstract Types
Game type is taken as base type for both HexTicTacToe and TicTacToe
Completed - 18. Refactor and Reuse Code In Both Games
The code in both games is almost the same and can be extracted as a base abstract class for both with both games overriding certain methods if necessary.

SECTION III: GameDesign Project Feature and Test Cases Implementation Status and Description

GameDesign v1.0 - Requirement I

Completed - 1. Tic-Tac-Toe consists of 3x3 Square Cells
TestCases:
Test1 : Test Generation of 3x3 Board
Test.java
Test2 : Test Initialisation of 3x3 Board
Test.java
Test3 : Test Cell Generation
Test.java

Completed - 2. Game Between Two Humans
TestCases:
Test1 : Test.java

Completed - 3. Game Between Human and Machine
TestCases:
Test1 : ComputerTest.java

Completed - 4. Winning Criteria - 3 Cells in Row/Column/Diagonal are in Same State.
TestCases:
Test1 : Test.java
checkColumn()
checkRow()
checkDiagonal()

Completed - 5. Announce Winning Player

GameDesign v2.0 - Requirement II

Completed - 6. Enhanced Tic-Tac-Toe Game Consist of 9x9 Squares...
TestCases:
Test1 : Test.java

Completed - 7. Enhanced Tic-Tac-Toe will continue to expand in depth levels...
TestCases:
Test1 : Test.java

Completed - 8. Extend Game to 4x4 Board
TestCases:
Test1 : Test.java

Not Completed - 9. Human Player is Biased...
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

Not Completed - 10. Storing and Retrieving Game State
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

Completed - 11. Store Players Game Statistics: Leaderboard
TestCases:
Test1 : Test.java HexTest.java

GameDesign v3.0 - Requirement III

Completed - 12. Super Tic-Tac-Toe Game Extends Enhanced Tic-Tac-Toe Game...
TestCases:
Test1 : HexTest.java

Completed - 13. Design Winning and Losing Criterias On All Edges...

TestCases:

Test1 : HexTest.java

checkRow()

checkDiagonal()

Not Completed - 14. Incorporate Irregular shaped Hexagonal Boards

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

GameDesign v4.0 - Requirement IV

Not Completed - 15. Incorporate Biased Game Board

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

Not Completed - 16. Incorporate Connect Four Game In Design

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

Not Applicable - 17. Discover Newer Abstract Types

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

Completed - 18. Refactor and Reuse Code In Both Games

TestCases:
Test1 : Test.java HexTest.java

*SECTION III: How to Run/Test Your Code?*

Describe How To Run Your Code ( commands)

```
git clone https://github.com/manansingla4/FKApplyDesign.git
cd FKApplyDesign
git checkout Dev
cd src
javac -cp .:junit.jar *.java
java Main
```

To run Test Cases :

```
java -cp .:junit.jar:hamcrest.jar org.junit.nner.JUnitCore Test
java -cp .:junit.jar:hamcrest.jar org.junit.nner.JUnitCore HexTest
java -cp .:junit.jar:hamcrest.jar org.junit.nner.JUnitCore ComputerTest
```

Can I Run Test.java to test your whole source code?

No

## Why design is better

The classes are developed using future needs in mind and can be extended to a wide variety of board games if necessary. The state is well encapsulated inside each class and is accessible only through public methods. All classes are loosely coupled.