DESIGN DOCUMENT

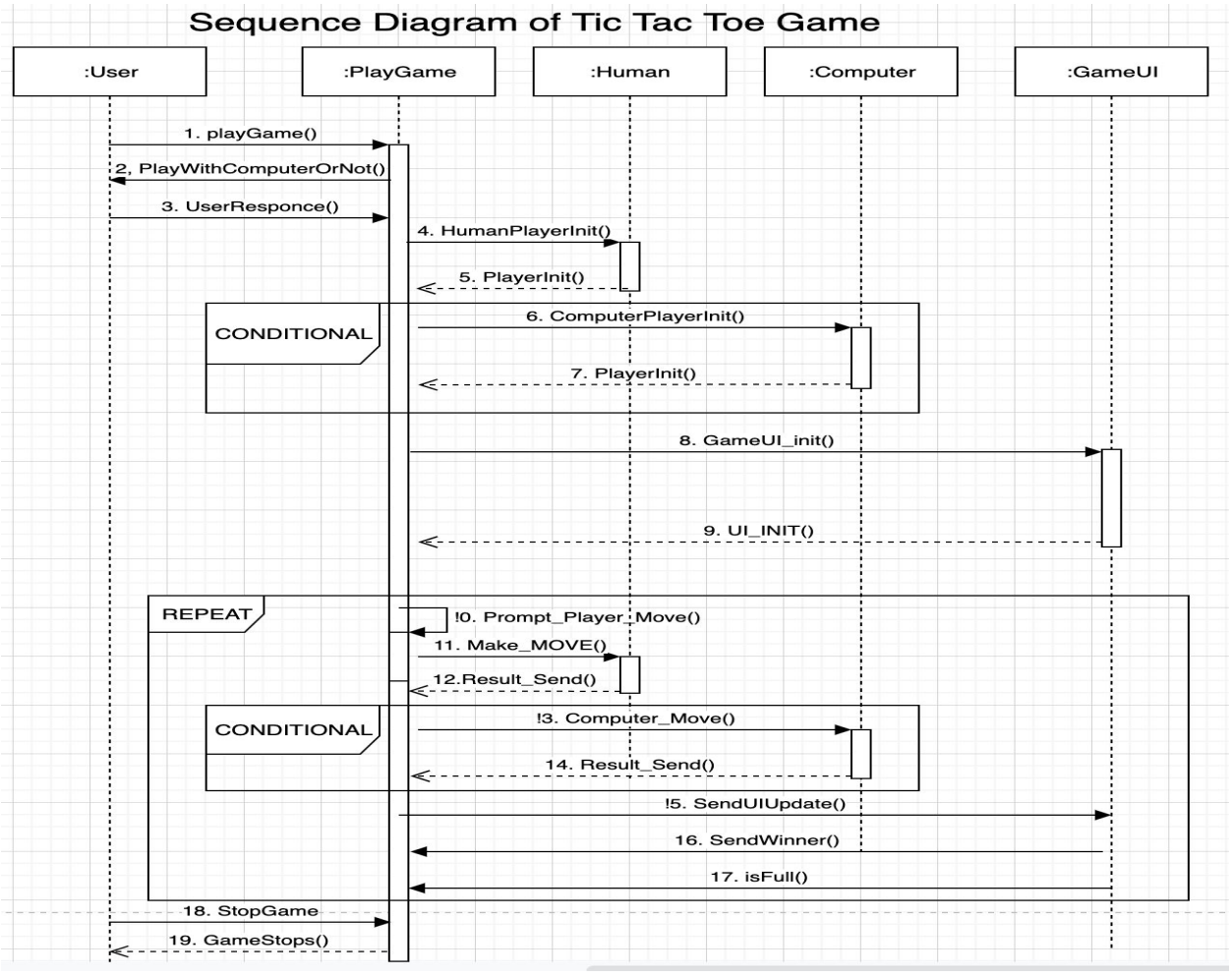
Tic-Tac-Toe Game

EmployeeID - 138952

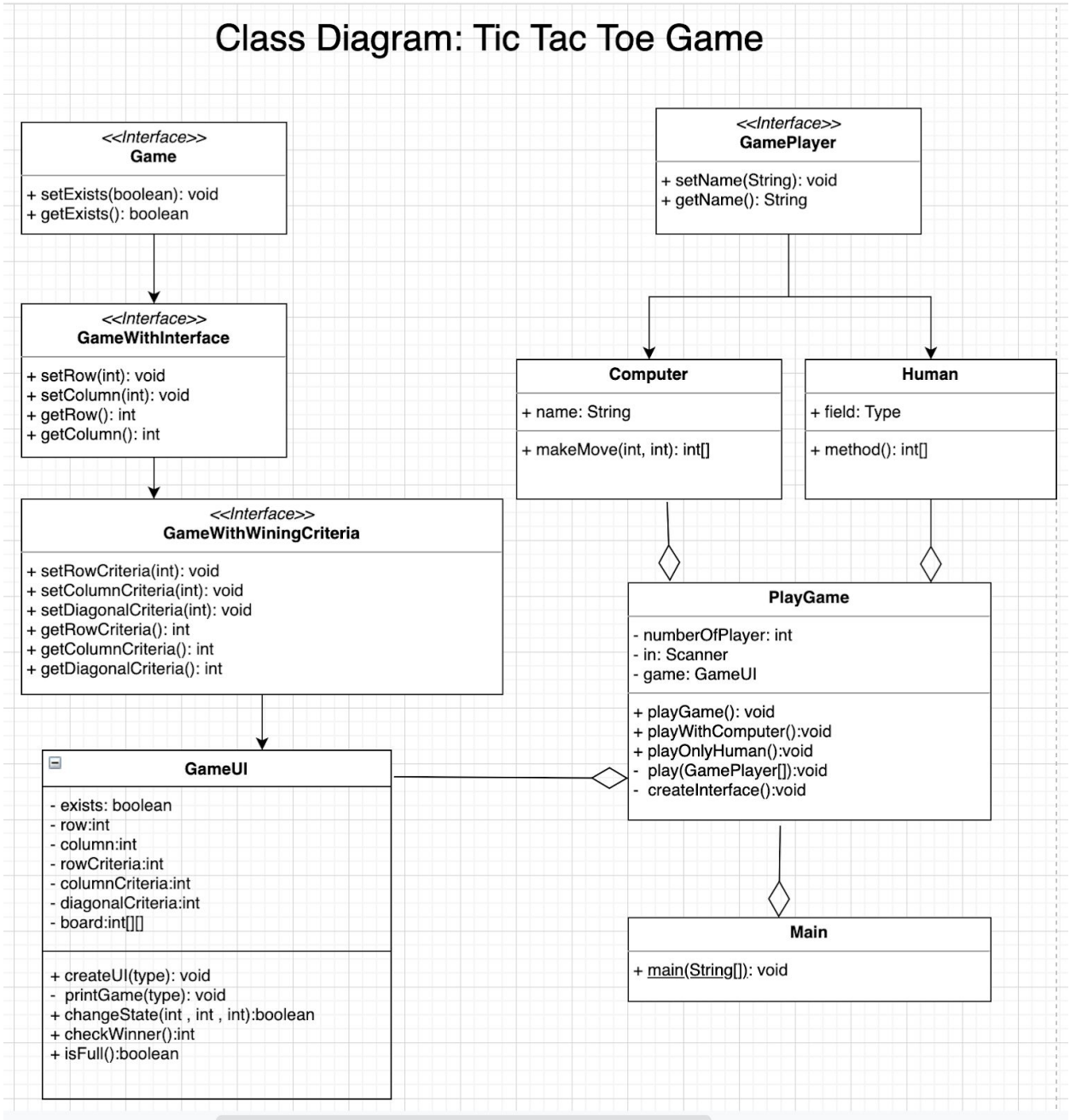
SECTION I: First Iteration Game Project Design
[Till Phase III or Phase IV Submitted Upto 29th January, 2020]

PART I :

Sequence Diagrams



Class Diagrams



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

Common Design/Choices:

1. Abstract type Game with infinite board.
2. Abstract type GameInterface restricts Game with rows and column constraints.
3. Abstract type GameWithWinningCriteria defines GameInterface with winning criteria.
4. GameUI where game rules and interaction with user defined.
5. Abstract type player.
6. Two players Human, Computer which implements player.
7. Importance given to abstraction.

Convention/Assumption:

1. Design was based upon the assumption that there can be any number of players.
2. Any dimension of board.
3. Any winning criteria.
4. Freedom in defining move of computer and human accordingly.

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

GameDesign v2.0 - Requirement I

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

1> Simple design which enhances abstraction.

Completed - 2. Game Between Two Humans

1> Player as interface and human class implements it.

Completed - 3. Game Between Human and Machine

1> Defined Machine as Computer class.

2> It implements Player class and define logic of move.

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

1> Abstract winning criteria which can be modified easily.

Completed - 5. Announce Winning Player

GameDesign v2.0 - Requirement II

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

1> Developed in accordance with feature 7.

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

1> Added Layers in Tic-Tac-Toe such that each cell can be expanded to another level.

Completed - 8. Extend Game to 4x4 Board

1> Dimensions are dynamic.

Completed - 9. Human Player is Biased...

1> Added option of back move only one time.

NotCompleted - 10. Storing and Retrieving Game State

Completed - 11. Store Players Game Statistics: Leaderboard

1> Added Leaderboard class to handle it.

GameDesign v3.0 - Requirement III

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

1> Build separate class HeaxagonalUI which implete GameWithWinningCriteria interface to handle it separately.

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

1> Designed in HexagonalUI separate module to handle it.

NotCompleted- 14. Incorporate Irregular shaped Hexagonal Boards

GameDesign v4.0 - Requirement IV

NotCompleted - 15. Incorporate Biased Game Board

Completed - 16.Incorporate Connect Four Game In Design

1> Separate class Inherit from GameUI(Renames:RectangleUI) Handle it.

Completed - 17. Discover Newer Abstract Types

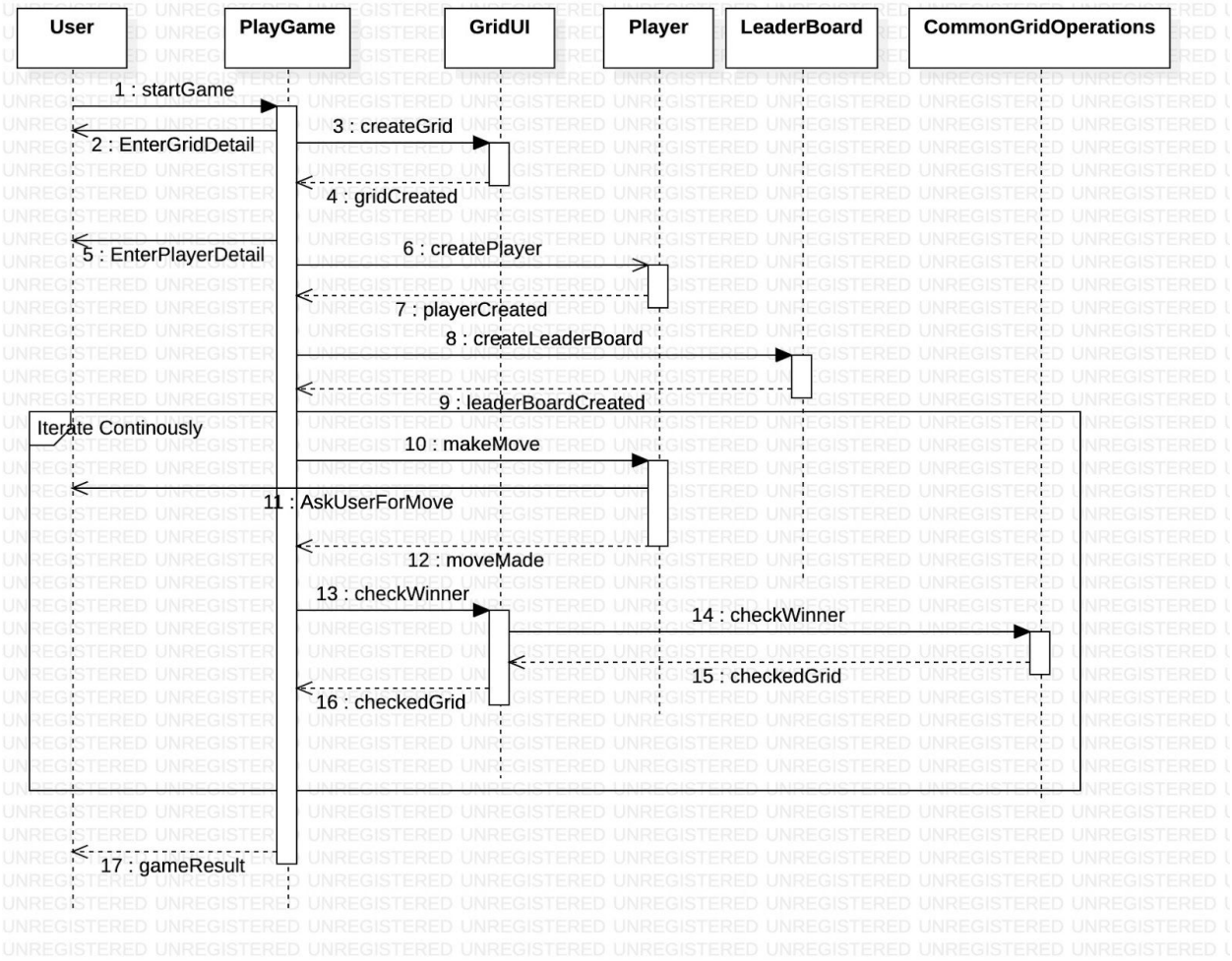
1> Abstraction is used

Completed - 18. Refactor and Reuse Code In Both Games

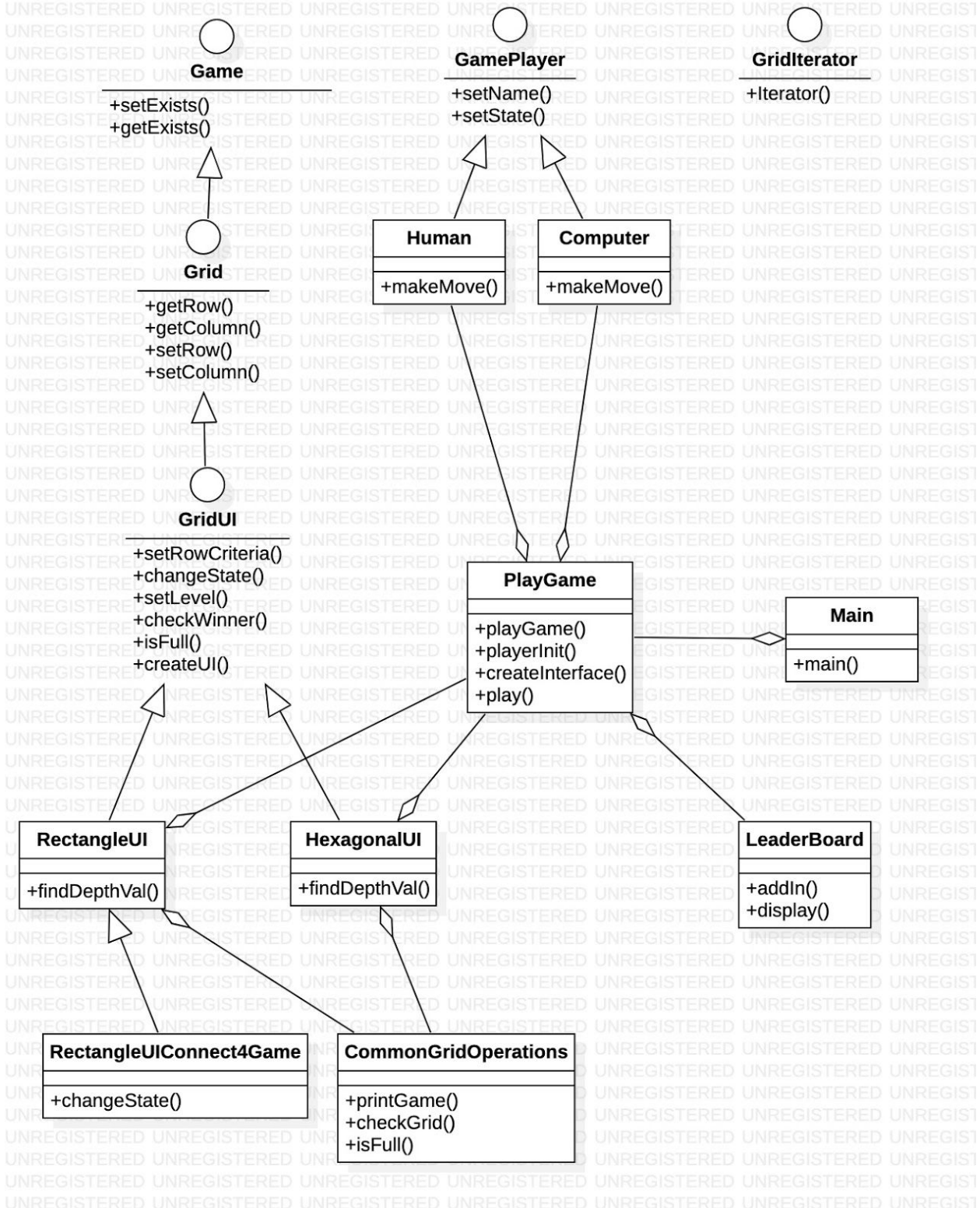
SECTION II: Second Iteration[Refactoring/Redesign] Game Project Design
[Till Phase III or Phase IV Submitted Upto 03rd February, 2020]

PART I :

Sequence Diagrams



Class Diagrams



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

Interfaces:-

1> Design starts with Game as an abstract interface of TicTacToe game with infinite row and column.

2> This gets extended by Grid Interface and transforms the game into finite row and column by defining row and column of grid.

3> Now GridUI Interface extends GameWithInterface and defines criteria of winning of row ,column , and diagonal.

4> GamePlayer Interface is an abstract type of player which plays games.

5> GridIterator Interface is for lambda.

CLasses:-

1> RectangleUI class implements GridUI and defines UI structure of grid. Also defines behaviour of displaying grid, change in grid, checking if anyone win, then finding if the grid is full or not.

2> PlayGame class is an actual class where gameUI interacts with users, it has behaviour PlayWithHuman , PlayWithComputer with common behaviour play where users play games.

3> Human Class implements GamePlayer and defines Human Player.

4> Computer Class implements GamePlayer and defines Computer Player.

5> Main Class starts PlayGame class which starts the game.

6> HexagonalUI implements GridUI and defines UI for grid with Hexagonal cells.

7> Leaderboard class.

8> RectangleUIConnect4game extends RectangleUI to perform connect 4 games.

9> CommonGridOperations class implements all grid basic operations.

Why this design?

1> Well for one we can manipulate the size of grid.

2> Number of players can be dynamic.

3> Winning criteria is also dynamic.

4> Anyone can also define their own implementation of grid.

5> Make a move function in Human and Computer is purely dynamic and can be designed in any way independently.

6> Also, grid operations are defined in separate classes which are shared among any type of grid.

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

GameDesign v2.0 - Requirement I

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

1> Abstraction in GridUI.

2> This feature is defined in RectangleUI.

Completed - 2. Game Between Two Humans

1> Now Player Can define his state with which he want to play.

Completed- 3. Game Between Human and Machine

1> Both are defined in separate class and hence have separate Move logic.

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

1> Winner is checked by common function checkWinner which Take lambda as an argument to define search in different direction.

Completed - 5. Announce Winning Player

GameDesign v2.0 - Requirement II

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

1> Specialised from feature 7.

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

1> Improved logic of depth traversal.

Completed - 8. Extend Game to 4x4 Board

1> Dimensions are dynamic.

Completed - 9. Human Players are Biased…

1> Only one time back when that person has a turn only.

NotCompleted- 10. Storing and Retrieving Game State

Completed - 11. Store Players Game Statistics: Leaderboard

1> Leaderboard has score based upon levels of a particular level.

2> Leaderboard is valid in current game session only.

GameDesign v3.0 - Requirement III

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

1> CommonGridOperations used to increase code reuse.

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

1> Logic in HexagonalUI , implementation in CommonGridOperations.

NotCompleted - 14. Incorporate Irregular shaped Hexagonal Boards

GameDesign v4.0 - Requirement IV

NotCompleted - 15. Incorporate Biased Game Board

Completed - 16.Incorporate Connect Four Game In Design

1> Inherited from RectangleUI.

Completed - 17. Discover Newer Abstract Types

1> Abstraction done in grid operations.

2> Abstraction in methods to enhance use of SOLID principles.

Completed - 18. Refactor and Reuse Code In Both Games

1> Major focus on Reuse by creating more abstract types.

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

RectangleUITest,testAdd

Completed - 2. Game Between Two Humans

TestCases:

Test1 : Checking if the winner is identified or not.

RectangleUITest,testAdd

NotCompleted - 3. Game Between Human and Machine

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

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

TestCases:

Test1 : Checking Winner

RectangleUITest,testAdd

Completed - 5. Announce Winning Player

Test1:Checking Winner

RectangleUITest,testAdd

GameDesign v2.0 - Requirement II

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

TestCases:

Test1 : change test dimension

RectangleUITest,testAdd

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

TestCases:

Test1 : change test level

RectangleUITest,testAdd

Completed - 8. Extend Game to 4x4 Board

TestCases:

Test1 : changel test dimension

RectangleUITest,testAdd

NotCompleted - 9. Human Players are Biased...

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

NotCompleted- 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 : Leaderboard test

LeaderboardTest,testAdd

GameDesign v3.0 - Requirement III

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

TestCases:

Test1 :

HexagonalUITest,testAdd

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

TestCases:

Test1 :

HexagonalUITest,testAdd

NotCompleted- 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

NotCompleted - 15. Incorporate Biased Game Board

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

Completed - 16. Incorporate Connect Four Game In Design

TestCases:

Test1 :

RectangleUIConnect4GameTest.java,testAdd

NotCompleted - 17. Discover Newer Abstract Types

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

NotCompleted - 18. Refactor and Reuse Code In Both Games

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

SECTION III: How to Run/Test Your Code?

Describe How To Run Your Code

1> Compile and run the Main class file.
2> For every decision it will ask for input.
3> Each individual test file should be used to test only a particular component.

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

1> No

Are you providing all Input/Output files to run Test Code using Test.java?

1> No