DESIGN DOCUMENT TEMPLATE

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
Dependency Diagrams

PART II : Common Design/Choices and Conventions/Assumptions and Detailed Descriptions etc.
Used inheritance wherever possible, and used camelCase for variable declarations.

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

GameDesign v2.0 - Requirement I
<COMPLETED> - 1. Tic-Tac-Toe consists of 3x3 Square Cells
Implemented using TicTacToeBoard Class which implements boardInterface class.
<COMPLETED> - 2. Game Between Two Humans
Handled by creating ticTacToeGame that implements gameInterface. Can handle multiple players.
<COMPLETED> - 3. Game Between Human and Machine
Created a special player with the name “machine” which doesn’t take input from the stdin and insteads plays random moves on the board.
<COMPLETED> - 4. Winning Criteria - 3 Cells in Row/Column/Diagonal are in Same State.
Made a function analyse_board to implement the logic for winning/losing
<COMPLETED> - 5. Announce Winning Player

The ticTacToeGame class has object of players and hence can announcing a winning player easily.

GameDesign v2.0 - Requirement II

<COMPLETED> - 6. Enhanced Tic-Tac-Toe Game Consist of 9x9 Squares...

Done by making a new class that helps you play the tic tac toe game recursively

<COMPLETED> - 7. Enhanced Tic-Tac-Toe will continue to expand in depth levels...

Done by making a new class that helps you play the tic tac toe game recursively

<COMPLETED> - 8. Extend Game to 4x4 Board

Done by taking user specific inputs for board size and then setting it the constructor of the board.

<COMPLETED> - 9. Human Player is Biased...

Added feature for undoing any number of moves.

<COMPLETED> - 10. Storing and Retrieving Game State

Did this by storing the current board config, as well as the just previous move. This is stored in the instance of TicTacToeBoard which implements BoardInterface.

<COMPLETED> - 11. Store Players Game Statistics: Leaderboard

Player stores its individual results after a match and the results are announced & updated after a match is completed.

GameDesign v3.0 - Requirement III

<COMPLETED> - 12. Super Tic-Tac-Toe Game Extends Enhanced Tic-Tac-Toe Game...

Made a new board Class hexBoard which extends boardInterface to play this game.

<COMPLETED> - 13. Design Winning and Losing Criterias On All Edges...

The analyse_board of the hexBoard class handles the winning/losing criteria. It checks who has the longest diagonal board to announce the winner.

<NOT COMPLETED> - 14. Incorporate Irregular shaped Hexagonal Boards

Feature Specific Design Decision?

GameDesign v4.0 - Requirement IV

<NOT COMPLETED> - 15. Incorporate Biased Game Board

Feature Specific Design Decision?

<COMPLETED> - 16.Incorporate Connect Four Game In Design

Created an additional board class for connectFour that implements boardInterface which has the same functions as TicTacToeBoard and HexBoard

<NOT COMPLETED> - 17. Discover Newer Abstract Types

Feature Specific Design Decision?

<PARTIALLY COMPLETED> - 18. Refactor and Reuse Code In Both Games

Currently using the interface for ConnectFour and TicTacToe, which implement the same functions.

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
Dependency Diagrams

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

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

GameDesign v2.0 - Requirement I
<COMPLETED> - 1. Tic-Tac-Toe consists of 3x3 Square Cells
Implemented using TicTacToeBoard Class which implements boardInterface class.
<COMPLETED> - 2. Game Between Two Humans
Handled by creating ticTacToeGame that implements gameInterface. Can handle multiple players.
<COMPLETED> - 3. Game Between Human and Machine
Created a special player with the name “machine” which doesn’t take input from the stdin and insteads plays random moves on the board.
<COMPLETED> - 4. Winning Criteria - 3 Cells in Row/Column/Diagonal are in Same State.
Made a function analyse_board to implement the logic for winning/losing
<COMPLETED> - 5. Announce Winning Player
The ticTacToeGame class has object of players and hence can announcing a winning player easily.

GameDesign v2.0 - Requirement II
<COMPLETED> - 6. Enhanced Tic-Tac-Toe Game Consist of 9x9 Squares...
Done by making a new class that helps you play the tic tac toe game recursively

<COMPLETED> - 7. Enhanced Tic-Tac-Toe will continue to expand in depth levels...
Done by making a new class that helps you play the tic tac toe game recursively
<COMPLETED> - 8. Extend Game to 4x4 Board
Done by taking user specific inputs for board size and then setting it the constructor of the board.
<COMPLETED> - 9. Human Player is Biased...
Added feature for undoing any number of moves.
<COMPLETED> - 10. Storing and Retrieving Game State

Did this by storing the current board config, as well as the just previous move. This is stored in the instance of TicTacToeBoard which implements BoardInterface.

<COMPLETED> - 11. Store Players Game Statistics: Leaderboard

Player stores its individual results after a match and the results are announced & updated after a match is completed.

GameDesign v3.0 - Requirement III

<COMPLETED> - 12. Super Tic-Tac-Toe Game Extends Enhanced Tic-Tac-Toe Game...

Made a new board Class hexBoard which extends boardInterface to play this game.

<COMPLETED> - 13. Design Winning and Losing Criterias On All Edges...

The analyse_board of the hexBoard class handles the winning/losing criteria. It checks who has the longest diagonal board to announce the winner.

<NOT COMPLETED> - 14. Incorporate Irregular shaped Hexagonal Boards

Feature Specific Design Decision?

GameDesign v4.0 - Requirement IV

<NOT COMPLETED> - 15. Incorporate Biased Game Board

Feature Specific Design Decision?

<COMPLETED> - 16.Incorporate Connect Four Game In Design

Created an additional board class for connectFour that implements boardInterface which has the same functions as TicTacToeBoard and HexBoard

<NOT COMPLETED> - 17. Discover Newer Abstract Types

Feature Specific Design Decision?

<PARTIALLY COMPLETED> - 18. Refactor and Reuse Code In Both Games

Currently using the interface for ConnectFour and TicTacToe, which implement the same functions.

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

Mention Class Name and Function Names?

Test2 : Test Initialisation of 3x3 Board

Mention Class Name and Function Names?

Test3 : Test Cell Generation

Mention Class Name and Function Names?

Completed - 2. Game Between Two Humans

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title
Mention Class Name and Function Names?

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?

<STATUS> - 4. Winning Criteria - 3 Cells in Row/Column/Diagonal are in Same State.
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

<STATUS> - 5. Announce Winning Player

GameDesign v2.0 - Requirement II

<STATUS> - 6. Enhanced Tic-Tac-Toe Game Consist of 9x9 Squares...
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

<STATUS> - 7. Enhanced Tic-Tac-Toe will continue to expand in depth levels...
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

<STATUS> - 8. Extend Game to 4x4 Board
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

<STATUS> - 9. Human Player is Biased...
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title

Mention Class Name and Function Names?

<STATUS> - 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?

<STATUS> - 11. Store Players Game Statistics: Leaderboard
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

GameDesign v3.0 - Requirement III

<STATUS> - 12. Super Tic-Tac-Toe Game Extends Enhanced Tic-Tac-Toe Game...
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

<STATUS> - 13. Design Winning and Losing Criterias On All Edges...
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title
Mention Class Name and Function Names?

<STATUS> - 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

<STATUS> - 15. Incorporate Biased Game Board
TestCases:
Test1 : Test Title
Mention Class Name and Function Names?
Test2 : Test Title

Mention Class Name and Function Names?

<STATUS> - 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?

<STATUS> - 17. Discover Newer Abstract Types

TestCases:

Test1 : Test Title

Mention Class Name and Function Names?

Test2 : Test Title

Mention Class Name and Function Names?

<STATUS> - 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

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

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

I have all you followed following guidelines given for writing test cases, if not then describe it what is working and what is not.

Instructions for Test Cases:

PHASE1 onwards code must have Unit test cases for every class and function as a unit. You can use Junit Library to write your Unit Test cases or otherwise in your own way. Your test cases should be organised as follows.

Suppose your code contains classes Game and Board in Game.java and Board.java.

Then you must have GameTest.java and BoardTest.java consisting of all test cases for Game and Board class object and member functions. Test case's given input and expected output should be stored in file and compared.

There will be one file Test.java which will invoke all tests of project covered in there respective files e.g. GameTest.java, BoardiTest.java etc..