CS3343 Analysis and Design Report

Reversi

Group 3

Content

* Introuduction
* User Stories
* Design Constraints
* Analysis Classes
* Sequence Diagram
* Test Report
* Bug Report

**Introduction**

Reversi is a classical games that played by lots of people before.

The reason behind the development of this game is simple, to bring joy to the users and for us, to challenge ourselves. We would like to explore how to develop this game and also, the techniques for project management.

There are Single-player mode and Multi-player mode. User can play with another human user or challenge the AI.

**User Story**

|  |
| --- |
| START THE Multi-player Mode |
| When multiplayer’s mode is selected, a chessboard will be created. Two white pieces and two black pieces are placed in the center of the grid. The black user can put chess first. |

|  |
| --- |
| START THE Single-player Mode |
| User can choose color and difficulty level. There are three difficulty levels, easy, normal and hard. After that, two white pieces and two black pieces are placed in the center of the grid. The game can start. |

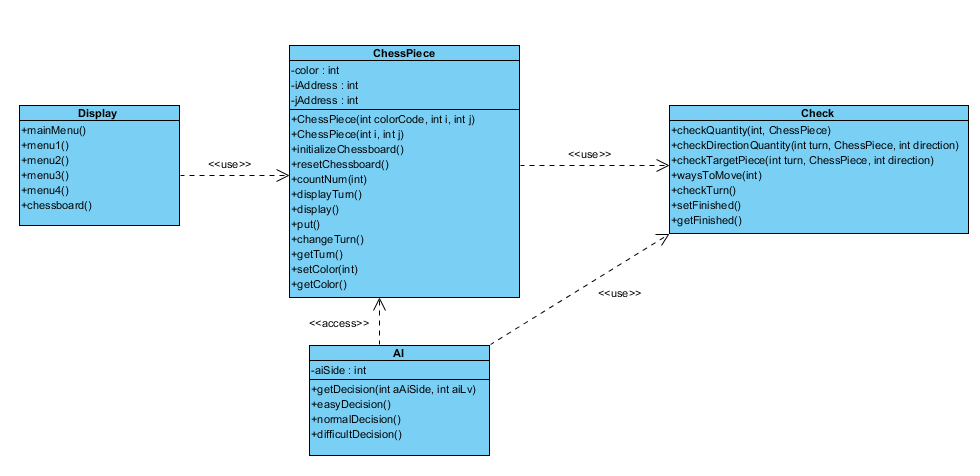
|  |
| --- |
| PUT CHESS |
| Player can enter “A-H” which represents columns and “1-8” which represents rows to put chess. Then the system will check whether the input is valid. If an invalid input is entered, player can enter the input again. For any valid input, the system will place the chess on the square and turn the color of other chess. At the end, the system will check whether the game is already finished and decide it’s black user’s turn or white user’s turn. |

**Design Constraints**

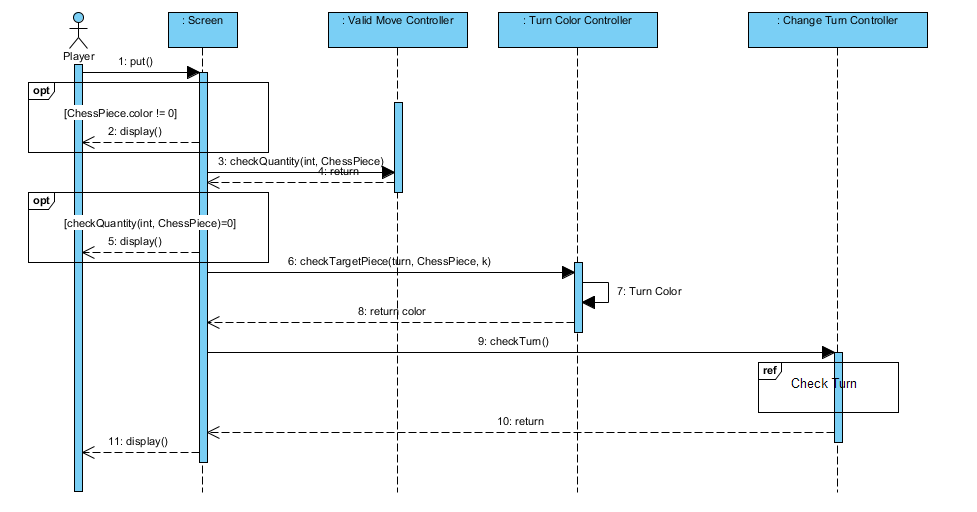
When we are developing this game, we found that a powerful AI is not easy to implement. For example, we have to develop three kinds of algorithm in order to differentiate the difference between easy, medium and hard level.

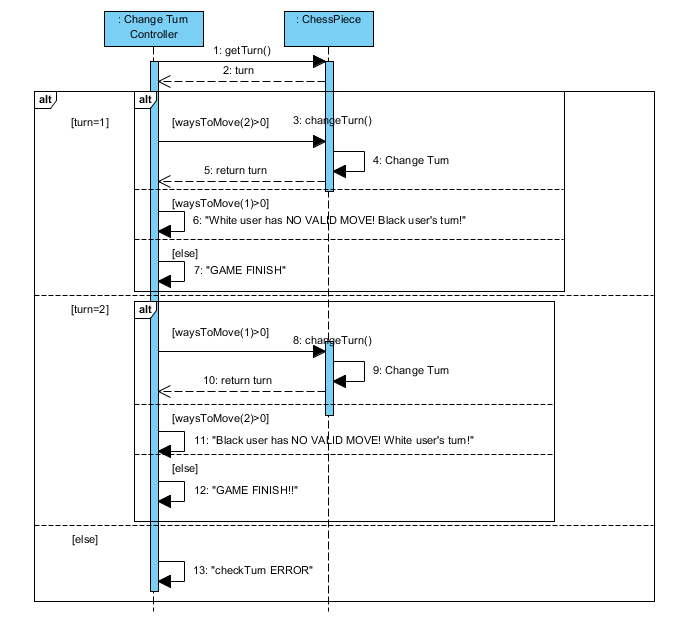
Besides, it is not easy to design test cases to check for different scenarios.

**Class Diagram**



**Sequence Diagram**





**Test Report**

**Version 1: Testing**

Unit Testing

1. InitializeChessboard

Purpose: Check whether a blank chessboard can be initialized or not.

1. ResetChessboard

Purpose: Check whether the four chess can be placed at the correct location.

1. Valid Input

Purpose: Check whether the game can detect it is a valid input and then proceed the computation

1. Invalid Input

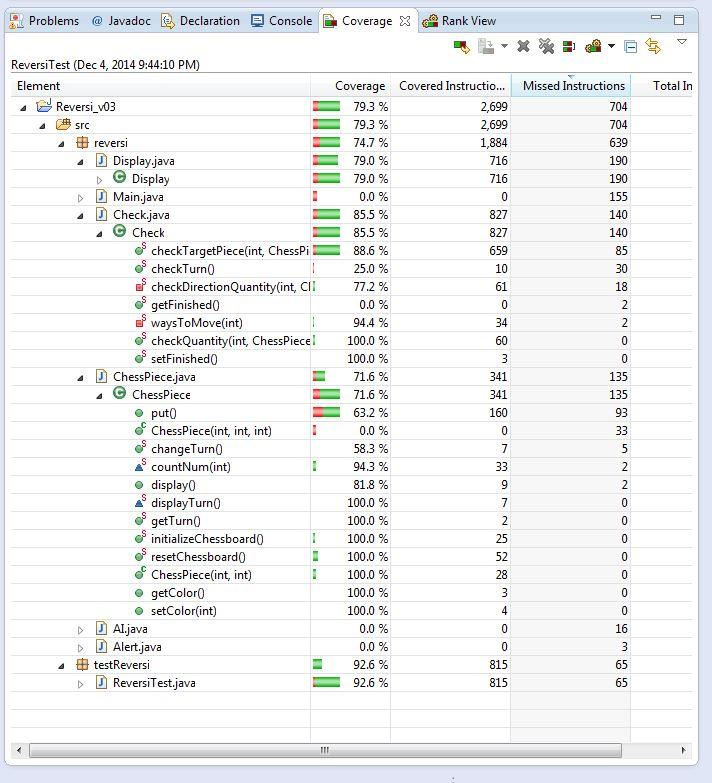
Purpose: Check whether the game can detect it is an invalid input and prevent the wrong input were being sent for computation. After receiving an invalid input, the game should detect it an inform user to inform a new input until the received input in valid.

Integration Testing

1. ValidPut

Purpose: Check whether the game can detect a valid input and then proceed to run chessPiece.put in order to put the chess on the chessboard.

Testing the input for D3,E6,F5,C4



**Version 2 of Test Cases**

Unit Testing

1. CheckQuantity

Purpose: To check whether the game will detect how many chess have to change color when user successfully put a chess on the chessboard.

1. CheckTurn

Purpose: To check when black chess was being put, will the game informed user to put a white chess

1. CheckFinish

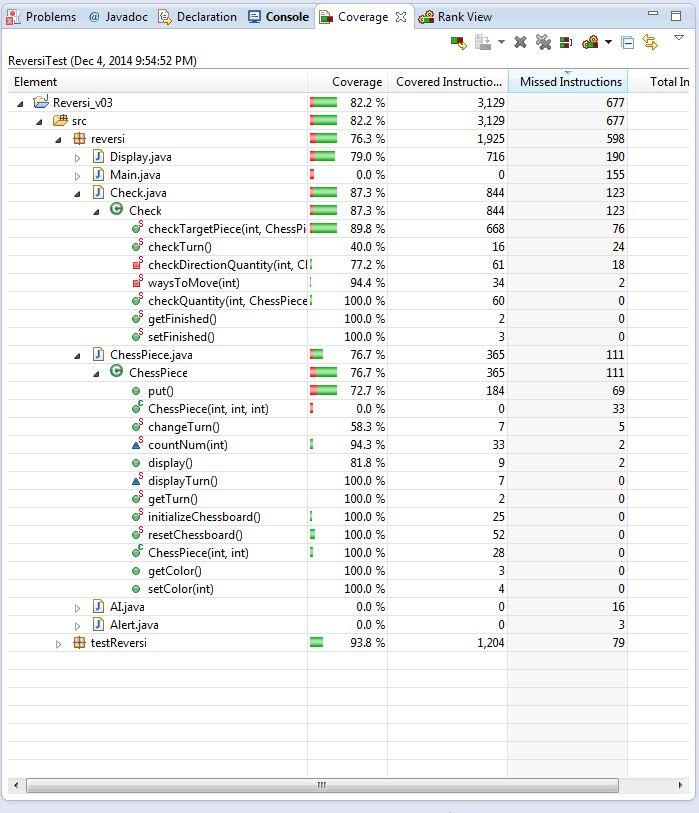
Purpose: To check whether the game can detect when the game finish.

1. CheckFinishBoundary

Purpose: To check whether the game is finished at boundary

1. CheckNotFinishBounday

Purpose: To check whether the game is not finished at boundary



As the AI function was being implement at week 13, so, we are planning the run the test cases and check whether the AI function can meet the design standards in the future.

**Version 3 of Test Cases**

Unit Testing

1. checkEasyAiValid\_AiMoveFirst()

Purpose: To check whether the AI could start the first move of game (Easy)

1. checkNormalAiValid\_AiMoveFirst()

Purpose: To check whether the AI could start the first move of game (Normal)

1. checkDifficultAiValid\_AiMoveFirst()

Purpose: To check whether the AI could start the first move of game (Difficult)

1. checkEasyAiValid\_AiMoveLast()

Purpose: To check whether the AI could start the second move of game (Easy)

1. checkNormalAiValid\_AiMoveLast()

Purpose: To check whether the AI could start the second move of game (Normal)

1. checkDifficultAiValid\_AiMoveLast()

Purpose: To check whether the AI could start the second move of game (Difficult)

