**Project Sprint #3**

Implement all the features that support a human player to play a simple or general SOS game against a human opponent and refactor your existing code if necessary. The minimum features include **choosing the game mode (simple or general), choosing the board size, setting up a new game, making a move (in a simple or general game),** and **determining if a simple or general game is over**. The following is a sample GUI layout. It is required to use a class hierarchy to deal with the common requirements of the Simple Game and the General Game. **If your code for Sprint 2 has not considered class hierarchy, it is time to refactor your code**.

|  |  |  |
| --- | --- | --- |
| SOS Icon  Description automatically generated Simple game Icon  Description automatically generated General game Board size  8 | | |
| Blue player  Icon  Description automatically generated S  Icon  Description automatically generated O | Chart, line chart  Description automatically generated | Red player  Icon  Description automatically generated S  Icon  Description automatically generated O |
|  | Current turn: blue (or red) | New Game |

Figure 1. Sample GUI layout of the working program for Sprint 3

**Deliverables: expand and improve your submission for sprint 2.**

1. **Demonstration (9 points)**

Submit a video of no more than five minutes, clearly demonstrating the following features.

1. A simple game that the blue player is the winner
2. A simple draw game with the same board size as (a)
3. A general game that the red player is the winner, and the board size is different from (a)
4. A general draw game with the same board size as (c)
5. Some automated unit tests for the simple game mode
6. Some automated unit tests for the general game mode

In the video, you must explain what is being demonstrated.

1. **Summary of Source Code (1 points)**

|  |  |  |
| --- | --- | --- |
| Source code file name | Production code or test code? | # lines of code |
| Main.java | Production | 5 |
| MyFrame.java | Production | 207 |
| boardPerfromed.java | Production | 205 |
| checkMovePerFromed.java | Production | 84 |
| newGame.java | Production | 29 |
| Pair.java | Production | 26 |
| radioPerfromedClass.java | Production | 33 |
| Total | | 589 |

**You must submit all source code to get any credit for this assignment.**

1. **Production Code vs User stories/Acceptance Criteria (3 points)**

Summarize how each of the user story/acceptance criteria is implemented in your production code (class name and method name etc.)

|  |  |
| --- | --- |
| **User Story ID** | **User Story Name** |
| 1 | Choose a board size |
| 2 | Choose the game mode of a chosen board |
| 3 | Start a new game of the chosen board size and game mode |
| 4 | Make a move in a simple game |
| 5 | A simple game is over |
| 6 | Make a move in a general game |
| 7 | A general game is over |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Story ID** | **AC ID** | **Class Name(s)** | **Method Name(s)** | **Status (complete or not)** | **Notes (optional)** |
| 1. Choose a board size | 1. choose board size from combo box (3 to 9) | MyFrame.java  Location: Main/Head frame | Jcombobox gameBoardSize;  Integer[] boardSizeSelection={0, 3, 4, 5, 6, 7, 8, 9}; | Complete | Default will be 0. Player need to choose board size. |
| 2. choose the game mode | 2.1 simple game | MyFrame.java  Location: Main/Head frame | JradioButton simpleRadio; | Completed | The default selection |
|  | 2.2 general game | MyFrame.java  Location: Main/Head frame | JradioButton generalRadio; | Completed |  |
| 3 Start new game | Select New game | MyFrame.java  Location: bottom panel  newGame.java | actionPerformed of  Jbutton newGmButton; | Completed |  |
| 4.Make a move | Make a move in any mode | boardPefromed.java | actionPerfroemed(Actionevent e) | Completed |  |
| 5 Simple game over | Simple game end | boardPerFromed.java | onSOSButtonPressed | Complete |  |
| 6 Make a move on General game | Play general game | boardPerFromed.java | actionPerfroemed(Actionevent e) | Complete |  |
| 7 general game over | General game end | boardPerFromed.java | onSOSButtonPressed | Not | Can still play the game but there are still some minor bug need to fix to make the game accurate |

1. **Tests vs User stories/Acceptance Criteria (3 points)**

Summarize how each of the user story/acceptance criteria is tested by your test code (class name and method name) or manually performed tests.

|  |  |
| --- | --- |
| **User Story ID** | **User Story Name** |
| 1 | Choose a board size |
| 2 | Choose the game mode of a chosen board |
| 3 | Start a new game of the chosen board size and game mode |
| 4 | Make a move in a simple game |
| 5 | A simple game is over |
| 6 | Make a move in a general game |
| 7 | A general game is over |

4.1 Automated tests directly corresponding to some acceptance criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Story ID** | **Acceptance Criterion ID** | **Class Name (s) of the Test Code** | **Method Name(s) of the Test Code** | **Description of the Test Case (input & expected output)** |
| 1 player Win for Simple Game | 1.1 player 1 Win | MainTest.java | simpleGame | Ensure the player 1 win by checking the SOS on the board and the message to announce player 1 win is correct |
|  | 1.2Player 2 Win | MainTest.Java | simpleGame | Ensure the player 2 win by checking the SOS on the board and the message to announce player 2 win is correct |
|  | … |  |  |  |
| 2 Player Win For general game | 2.1Player 1 Win | MainTest.Java | generalGame | player 2 has Ensure to check the number of SOS which player 1 has have to be > than the number of SOS |
|  | 2.1Player 2 Win | MainTest.Java | generalGame | player 1 has Ensure to check the number of SOS which player 2 has have to be > than the number of SOS |

4.2 Manual tests directly corresponding to some acceptance criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Story ID** | **Acceptance Criterion ID** | **Test Case Input** | **Test Oracle (Expected Output)** | **Notes** |
| 1 | 1.1 |  |  |  |
|  | 1.2 |  |  |  |
|  | … |  |  |  |
| 2 | 2.1 |  |  |  |
|  | … |  |  |  |

For this sprint I haven’t finished my Junit test because I have some bug need to fix for my General mode and I took me a lot of time and also I try to figure out how to draw a line when there is an SOS appear. I did manage to draw the line but when I moved my mouse to there the line is disappear. It took me a lot of time to try to fix those bug.

Some of the Junit tests I did to test the acceptance criteria 1 to 4 and 6 I did from my last sprint only 5 and 7 I need to do it for this sprint.

1. **Describe how the class hierarchy in your design deals with the common and different requirements of the Simple Game and the General Game**? **(4 points)**

\_The SOS game contain a main class call the GUI class name MyFrame to display the GUI to the user. On MyFrame class I call radioPermedClass when the user select the O or S in other to make a move, the class make sure the text associated with O and S selection is correct. On MyFrame I called boardPerfromed class in other to create a board in the user according to the board size selection which user selected. On the boardPerfromed when the user make a move, the board class will called checkMovePerfromed which inheritance a Pair of the move coordinate from the class Pair, the checkMovePerfromed will begin to check the surround of that coordinate to see if he can find SOS or not. If the SOS is found or not it will send the result back to boardPerfdromed to do check win condition or continue the game.

\_For simple game if there is a SOS send back the game will end and a popup will appear for the user to start a new game.

\_For General mode if there is a SOS the game will continue and increasing a count for that player by 1.