**Github link:**

**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 |
| Sprint2GameLogic |  | 138 |
| Sprint2UI |  | 449 |
| BoardGlassPane |  | 64 |
| BoardGlassPaneTest |  | 27 |
| Sprint2GameLogicTest |  | 28 |
| Total | | 706 |

**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 and Name** | **AC ID** | **Class Name(s)** | **Method Name(s)** | **Status (complete or not)** | **Notes (optional)** |
| 1. Choose a board size | 1.1 | Sprint2UI | startGame(): this method will retrieve the selected board size.  displayBoard(int): this method creates the game board with the chosen size. | complete |  |
|  | 1.2 |  |  | complete | I went with predetermined board sizes that the user chooses from, so there is no possibility of an invalid choice anymore. |
|  | … |  |  |  |  |
| 2. Choose the game mode of a chosen board | 2.1 | Sprint2UI | startGame(): this method will retrieve the selected game mode.  displayBoard(int): this method creates the game board with the chosen game mode. | Complete |  |
|  | 2.2 | Sprint2UI | startGame(): this method will retrieve the selected game mode.  displayBoard(int): this method creates the game board with the chosen game mode. | Complete |  |
|  | 2.3 |  |  | complete | I removed the possibility of no game mode being selected by having an option pre-selected so there will always be a game mode selected. The info regarding 2.3 is now null. |
| 3. Start a new game of the chosen board size and game mode | 3.1 | Sprint2UI | startGame(): this method will start the game with the chosen board size and game mode. | complete |  |
|  | 3.2 |  |  |  | Similarly to previous items on this sheet, I removed the possibility of this happening by having a preset board size and preset game mode selection so no matter what, the game can’t be created without a board size and game mode. Therefore, the info regarding 3.2 is now null. |
| 4. Make a move in a simple game | 4.1 | Sprint2UI | takeTurn(JButton) allows a user to make a move. | complete |  |
|  | 4.2 | Sprint2UI | takeTurn(JButton): This technically comes into play here, but also nothing will happen if no move is made. | complete |  |
|  | 4.3 | Sprint2UI | TakeTurn(JButton): There is some logic to show a pop up if a move is invalid and lets the user make a new move. | complete |  |
| 5. A simple game is over | 5.1 | Sprint2UI  Sprint2GameLogic | takeTurn(Jbutton button): This works with checkForSOS(int,int,string,Boolean) to make a final move and detect a draw. | complete |  |
|  | 5.2 | Sprint2UI  Sprint2GameLogic | takeTurn(Jbutton button): This works with checkForSOS(int,int,string,Boolean) to make a final move and detect a win. | complete |  |
| 6. Make a move in a general game | 6.1 | Sprint2UI | takeTurn(JButton) allows a user to make a move. | complete |  |
|  | 6.2 | Sprint2UI | takeTurn(JButton): This technically comes into play here, but also nothing will happen if no move is made. | complete |  |
|  | 6.3 | Sprint2UI | TakeTurn(JButton): There is some logic to show a pop up if a move is invalid and lets the user make a new move. | complete |  |
| 7. A general game is over | 7.1 | Sprint2UI | takeTurn(JButton): This works with determineWinnerOrDraw() and checkforSOS() to detect a draw. | complete |  |
|  | 7.2 | Sprint2UI | takeTurn(JButton): This works with determineWinnerOrDraw() and checkforSOS() to detect a win. | complete |  |

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 |
| 6 | Make a move in a general game |

4.1 Automated tests directly corresponding to the acceptance criteria of the above user stories

This time around I am testing out some of the new features

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User Story ID and Name** | **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)** | **Notes:** |
| 6. Make a move in a general game | 6.1 | Sprint2GameLogicTest | testSOSCountUpdate | Input an SOS on a 3by 3 board and expected the SOS counter to be updated to 1. Passed. |  |
| 6 and 4 |  | BoardGlassPaneTest | testAddLine, | I added a line to glassPane. The count should be 1 line. Passed. | This one kinda applies to both 6 and 4. I forgot to require it in the original stories, but I am testing the ability of creating lines for SOS. |
| … |  | BoardClassPaneTest | testClearLines | I added a line to glassPane. I expected the line to be cleared after using clearLines. Passed. | This one kinda applies to both 6 and 4. I forgot to require it in the original stories, but I am testing the ability of creating lines for SOS. |

4.2 Manual tests directly corresponding to the acceptance criteria of the above user stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Story ID and Name** | **Acceptance Criterion ID** | **Test Case Input** | **Test Oracle (Expected Output)** | **Notes** |
| 1. Choose a board size | 1.1 |  |  |  |
|  | 1.2 |  |  |  |
|  | … |  |  |  |
| 2. Choose the game mode of a chosen board | 2.1 |  |  |  |
|  | 2.2 |  |  |  |
|  | 2.3 |  |  |  |
| 3. Start a new game of the chosen board size and game mode | 3.1 |  |  |  |
|  | 3.2 |  |  |  |
| 4. Make a move in a simple game | 4.1 |  |  |  |
|  | 4.2 |  |  |  |
|  | 4.3 |  |  |  |
| 5. A simple game is over | 5.1, 5.2 | I made moves appropriate for a win and a draw. | A popup was shown in both cases and correctly determined a win/draw. |  |
| 6. Make a move in a general game | 6.1 |  |  |  |
|  | 6.2 |  |  |  |
|  | 6.3 |  |  |  |
| 7. A general game is over | 7.1  7.2 | I made moves appropriate for a win and a draw. | A popup was shown in both cases and correctly determined a win/draw. |  |
| … |  |  |  |  |

4.3 Other automated or manual tests not corresponding to the acceptance criteria of the above user stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Test Input** | **Expected Result** | **Class Name of the Test Code** | **Method Name of the Test Code** |
|  |  |  |  |  |
|  |  |  |  |  |

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 Sprint2GameLogic class holds the logic which is based around the main game functions. Sprint2UI on the other hand holds the logic that the UI is based around. In other words, Sprint2UI is the outer shell, while Sprint2GameLogic is the brain.This separation helps to create a boundary between where the game logic and UI logic is stored.   
  
E.G. In Sprint2GameLogic, the hierarchy allows for method calls to deal with specific conditions such as a general game being played and how to win/lose/draw in that game using different methods within the class. The same can be said about Sprint2GameLogic and simple games.  
  
Sprint2GameLogic can work with Sprint2UI when necessary to show the user what is happening on screen and also calculate moves and win/loss/draws in the background. Sprint2GameLogic can take what the user inputs in Sprint2UI and calculate what it needs to with the information.

It may also be worth mentioning that BoardGlassPane also has a slight role in the hierarchy, but it is there more or less to be called by the other too when necessary to show lines on the board.