**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 |
| Board | Production | 156 |
| GameOptionPanel  Game  MyLabel  Main  Player | Production  Production  Production  Production  Production | 96  155  24  25  79 |
| Total | | 536 |

**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 | 1.1 | Board | Board(int size, Game game) | Complete | The Board class constructor allows choosing a board size |
|  | 1.2 |  |  |  |  |
|  | … |  |  |  |  |
| 2 | 2.1 | GameOptionPanel | apply() | complete | The GameOptionPanel allows choosing between "General" and "Simple" game modes and updates the game mode when applied. |
|  | … | Game | reset(GameMode mode, int boardSize) | complete | The reset method of the Game class allows starting a new game with the chosen board size and game mode. |
|  |  | Board | checkSOS(MyLabel label) | complete | The checkSOS method in the Board class handles making a move in a simple game. |
|  |  | Board | endGame() | complete | The endGame method in the Game class determines when a simple game is over. |
|  |  | Board | checkSOS(MyLabel label) | complete | The checkSOS method in the Board class handles making a move in a general game. |
|  |  | Board | endGame() | complete | The endGame method in the Game class determines when a general game is over. |

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 |  | |  | | |
|  | 1.2 |  | |  |  | |
|  | … |  | |  |  | |
| 2 | 2.1 |  | |  |  | |
|  | … |  | |  |  | |

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 Start a New Simple Game | Select "Simple" mode.  Choose a board size.  Click the "New Game" button. | The game board should reset, and the status message should indicate the starting player's turn (e.g., "RED Player's Turn"). | Verify that a new game in Simple mode starts as expected. |
|  | 1.2 Start a New General Game | Select "General" mode.  Choose a board size, e.g., 8x8.  Click the "New Game" button. | The game board should reset, and the status message should indicate the starting player's turn (e.g., "RED Player's Turn"). | Confirm that a new game in General mode begins correctly |
|  | … |  |  |  |
| 2 | 2.1 Switch between Game Modes | Start a new game in one mode (e.g., "Simple").  While in-game, open the options panel.  Select a different mode (e.g., "General"). | he game mode should change, and the board should reset to the new mode. | Ensure that switching game modes mid-game works as expected |
|  | … Make Moves on the Board | tart a new game (in any mode).  Select a cell on the board.  Click on a cell to place an "S" or "O." | he chosen cell should display the selected move type ("S" or "O"). | Verify that players can make moves on the board |
|  | **Track Player Scores** | Start a new game (in any mode).  Make a move (e.g., "S") by one player.  Confirm that the respective player's score increases. | The player's score should increment by 2 | Ensure that player scores are tracked correctly. |
|  | **Determine Game Winner** | Play a complete game with a clear winner | An alert dialog should appear declaring the winner (e.g., "Blue Player Wins!"). |  |

4.3 Other automated or manual tests not corresponding to the acceptance criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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)**

Handling Common Requirements:

Board Class: The Board class is in charge of the game board, which is shared by both Simple and General games. It is in charge of the label grid and its interactions. This class is in charge of the basic gameplay logic and scoring that is shared by both game types. The Player class manages player-specific attributes such as the move type (S or O) and their scores. Both game modes have these characteristics.GameOptionPanel Class: The GameOptionPanel class gives game options such as game mode (Simple or General) and board size. These settings apply to both game modes, making it a requirement in both.

Handling Different Requirements:

Game Class: The Game class is in charge of handling the overall operation of the game. It is in charge of the game mode's specialized elements. It detects which player's turn it is and modifies the status message accordingly. The Game class uses the GameMode enumeration to distinguish between Simple and General game modes. It ensures that game mode-specific logic is executed in accordance with the current game mode.

While the Board class maintains the game board for both modes, it also contains logic specific to each mode. The logic for recognizing SOS sequences in the checkSOS method . The addLine function additionally modifies the line color dependent on the current player's turn and the game mode.

Different Logic GameOptionPanel Class: The GameOptionPanel class allows the player to choose the game mode (Simple or General). Based on the current mode, the logic in this class enables or disables the checkboxes. When a different game mode is chosen, the apply method modifies the game parameters.