Project Sprint #2

Implement the following features of the SOS game: (1) the basic components for the game options (board size and game mode) and initial game, and (2) S/O placement for human players ***without*** checking for the formation of SOS or determining the winner. The following is a sample interface. The implementation of a GUI is strongly encouraged. You should practice object-oriented programming, making your code easy to extend. It is important to separate the user interface code and the game logic code into different classes (refer to the TicTacToe example). xUnit tests are required.

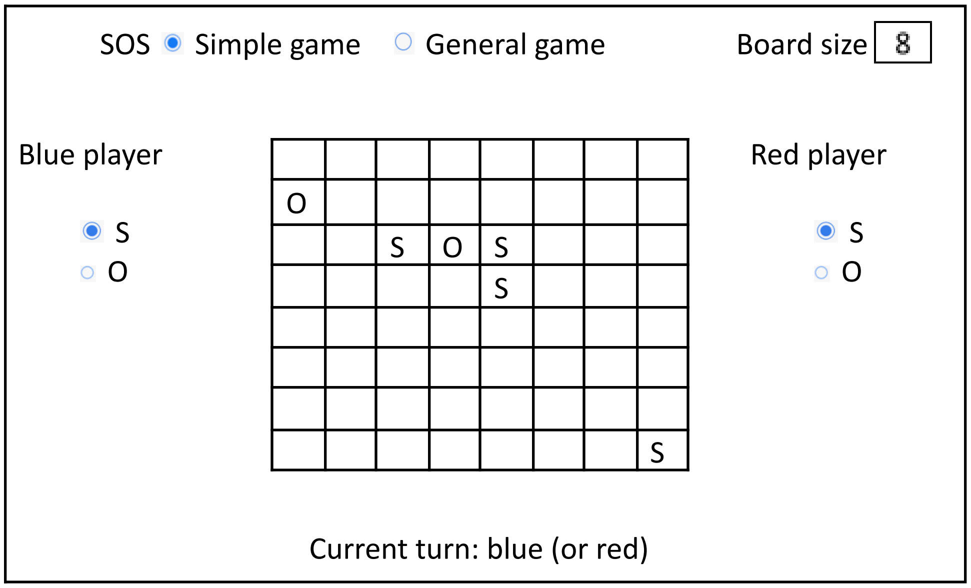

Figure 1. Sample GUI layout of the Sprint 2 program

Deliverables:

1. Demonstration (8 points)

Submit a video of no more than three minutes, clearly demonstrating that you have implemented the required features and written some automated unit tests. In the video, you must explain what is being demonstrated.

	Feature	
1	Choose board size	Yes
2	Choose game mode	No
3	Initial game of the chosen board size and game mode	Yes
4	"S" moves	Yes
5	"O" moves	Yes
6	Automated unit tests	Tes
…		

2. Summary of Source Code (1 points)

Source code file name	Production code or test code?	# lines of code
SOSBoard.java	Production	165
SOSGame.java	Production	33
	Total	198

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

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

Update your user stories and acceptance criteria from the previous assignment and ensure they adequately capture the requirements. Summarize how each of the following 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
6	Make a move in a general game

User Story ID and Name	AC ID	Class Name(s)	Method Name(s)	Status (complete or not)	Notes (optional)
1	1.1	SosGameSwing	SosGameSwing (method name is same as class name)	complete	N/A
2	1.2	SosGameSwing	SosGameSwing (method name is same as class name)	complete	N/A
3	2.1	SosGameSwing	SosGameSwing (method name is same as class name)	complete	N/A
4	2.2	SosGameSwing	SosGameSwing (method name is same as class name)	complete	N/A

4. 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

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)**
1	1.1	SOSGameSwingTest	testSelectBoardSize	● Simulate a user selecting a board size from available options. ● Verify that the selected board size is stored correctly.
2	2.1	SOSGameSwingTest	testSelectGameMode	● Simulate a user choosing a game mode (e.g., single player). ● Verify that the selected game mode is stored correctly.
3	3.1	SOSGameSwingTest	testStartNewGame	Simulate the user initiating a new game with the selected board size and game mode. Verify that a new game is created with the chosen parameters.
4	4.1	SOSGameSwingTest	testMakeMoveSimpleGame	● Simulate the user making a valid move in a simple game. ● Verify that the game state is updated correctly in response to the move.
5	5.1	SOSGameSwingTest	testMakeMoveGeneralGame	● Simulate the user making a valid move in a general game. ● Verify that the game state is updated correctly in response to the move.

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	1.1	Manually select a board size from available options (e.g., 8x8).	The selected board size should be reflected in the UI.	Check for visual changes in the selected board size.

1	1.2	Manually select a different board size from available options (e.g., 10x10).	The new board size should be reflected in the UI.	Check for visual changes in the selected board size.
2	2.1	Manually choose a game mode (e.g., single player) for the selected board.	The selected game mode should be displayed in the UI.	Check for the display of the selected game mode.
2	2.2	Manually choose a different game mode (e.g., multiplayer) for the selected board.	The new game mode should be displayed in the UI.	Check for the display of the selected game mode.
3	3.1	Manually initiate a new game with the selected board size and game mode.	A new game with the chosen parameters should be created.	Check for the initialization of a new game.
3	3.2	Manually initiate a new game with a different board size and game mode.	A new game with the new parameters should be created.	Check for the initialization of a new game with the updated parameters.
4	4.1	Manually make a valid move in a simple game.	The game state should be updated to reflect the move.	Check for the correct update of the game state.
4	4.2	Manually make an invalid move in a simple game and observe the game's response.	The game should respond appropriately to the invalid move (e.g., show an error message).	Check for the game's response to an invalid move.
5	5.1	Manually make a valid move in a general game.	The game state should be updated to reflect the move.	Check for the correct update of the game state.
5	5.2	Manually make an invalid move in a general game and observe the game's response.	The game should respond appropriately to the invalid move (e.g., show an error message).	The game should respond appropriately to the invalid move (e.g., show an error message).

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**
6	Automated: N/A	Observe that UI components behave as expected.	SOSGameSwingTest	testUIInteractions
7	Automated: N/A	UI components should adjust appropriately to changes in window size.	SOSGameSwingTest	testUIResizability
8	Automated: N/A	User should be able to navigate through the application without issues.	SOSGameSwingTest	testUserNavigation