Project Sprint #4

Implement all the features that support a player (**human or computer**) to play a simple or general SOS game against another player (**human or computer**). The minimum features include **choosing human or computer for red and/or blue players**, **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 computer opponent requirements. If your current code has not yet considered class hierarchy, it is time to refactor your code.

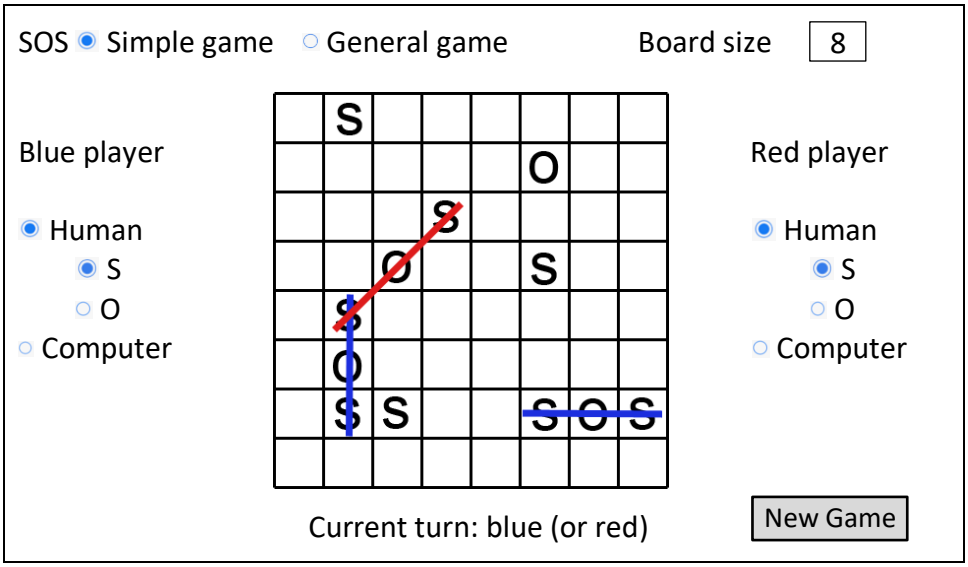


Figure 1. Sample GUI layout of the working program for Sprint 3

1. Demonstration (4 points)

Submit a video of no more than five minutes, clearly demonstrating that you have implemented the computer opponent features and written some automated unit tests.

1) A complete simple game where the blue player is a human, the red player is the computer, and there is a winner
2) A complete general game where the blue player is the computer, the red player is a human, and there is a winner
3) A complete simple game where both sides are played by the computer
4) A complete general game where both sides are played by the computer
5) Some automated unit tests for the computer opponent.

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

2. User Stories for the Computer Opponent Requirements (1 points)

- **User Story Template**: As a <role>, I want <goal> [so that <benefit>]

ID	User Story Name	User Story Description	Priority	Estimated effort (hours)
1	Empty board	As a player, I want an empty board	High	15
2	Game mode	As a player, I want to select between Simple and General	High	13
3	Player option	As a player, I want to choose between Red or Blue	Medium	10
4	Human or AI	As a player, I want to choose who to play with	High	14
5	Sign Options	As a blue or red player, I want to place S or O	High	10
6	Won	As a player, I want to know when the game is ending	High	16
7	Draw	As a player, I want to know if the game is a draw	High	20
8	Start new	As a player, I want to start new game	Low	12

3. Acceptance Criteria (AC) for the Computer Opponent Requirements (4 points)

User Story ID and Name	AC ID	Description of Acceptance Criterion	Status (completed, toDo, inPprogress)
1 story one	1.1	AC 1.1 <scenario description> Given An input for rows, columns When selects the number of rows and columns Then opens an empty board	
	1.2	AC 1.2 <scenario description> Given An empty board When chooses a simple or general game Then the game mode selected opens with conditions of that mode	
	…		
2 story two	2.1	AC 2.1 <scenario description> Given An empty board with selected game mode When chooses between blue or red Then able to make move with same color as the player blue or red	
	2.2	AC 2.2 <scenario description> Given Chosen player Blue When choose to place S or O Then able to make move with same sign chosen – S or O	
	2.3	AC 2.3 <scenario description> Given Chosen player Red When choose to place S or O Then able to make move with same sign chosen – S or O	
	…		
3 story three	3.1	AC 3.1 <scenario description> Given When the move is made When the SOS is made Then checks if the player has won	
	3.2	AC 3.2 <scenario description> Given When the SOS is made When the game mode is general Then shows incrementation of count	
4 story four	4.1	AC 4.1 <scenario description> Given the game is on last move When choose to place S or O in the last empty cell Then shows if the game is won by blue player or red player or if it's a daw	
	4.2	AC 4.2 <scenario description> Given The game is a draw / won	

		When choose to click new game Then gives a new board with empty cells	
5 story five	5.1	AC 5.1 <scenario description> Given A working game When Blue choose to play with Red computer Then Blue able to play with computer	
	5.2	AC 5.2 <scenario description> Given A working game When Red choose to play with Blue computer Then Red able to play with computer	

4. Summary of All Source Code (1 points)

Source code file name	Production code or test code?	# lines of code
Auto.java	production	135
Game.java	production	200
GUI.java	Production	490
TestBlueMoves	test	63
TestGameGUI	Test	52
TestEmptyBoard	Test	43
TestRandomMoves	Test	92
TestCompleteGame	Test	76
Total		

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

5. Production Code vs New User stories/Acceptance Criteria (2 points)

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

User Story ID and Name	**AC ID**	**Class Name(s)**	**Method Name(s)**	**Status (complete or not)**	**Notes (optional)**
1	1.1	Game	initGame()	Complete	Gives a new block of game
	1.2				
2	2.1	GUI	Ssgame.isSelected() or ggame.isSelected()	Complete	Choses between game modes
	…				
3	3.1	Game	BlueMoves() or redMoves()	Complete	Chooses between game player
4	4.1	GUI	blueS() or blueO() or redS() or redO()	Complete	Choses between signs – s or o
5	5.1	Auto	makeRandomMove()	Complete	Chooses to play with computer
6	6,1	Game	hasWon()	Complete	If the game is won
7	7.1	Game	isDraw()	Complete	Checks if the game is a draw
8	8.1	GUI / Game	New GUI()	Complete	Starts a new game

6. Tests vs New User stories/Acceptance Criteria (2 points)

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

6.1 Automated tests directly corresponding to some acceptance criteria

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	TestEmptyBoard	testNewBoard()	asks for new board & presents a new board
2	2.1	TestEmptyBoard	testGame()	checks if the correct win condition is delivered & expected game mode
3	3.1	TestGameGUI	Test()	….
4	4.1	TestGameGUI	Test()	Tests if the a proper board interface is produced for game
5	5.1	TestCompleteGame	Testhaswon()	Tests if the game is done, and someone has won & return true
6	6.1	TestCompleteGame	Testisdraw()	Tests is the game is a draw and return the answer
7	7.1	TestRandomMoves	testRandomMoves()	Checks if the random moves are being made when the computer is sleected
8	8.1	TestGameGUI	Testemptyboard()	Brings a new board

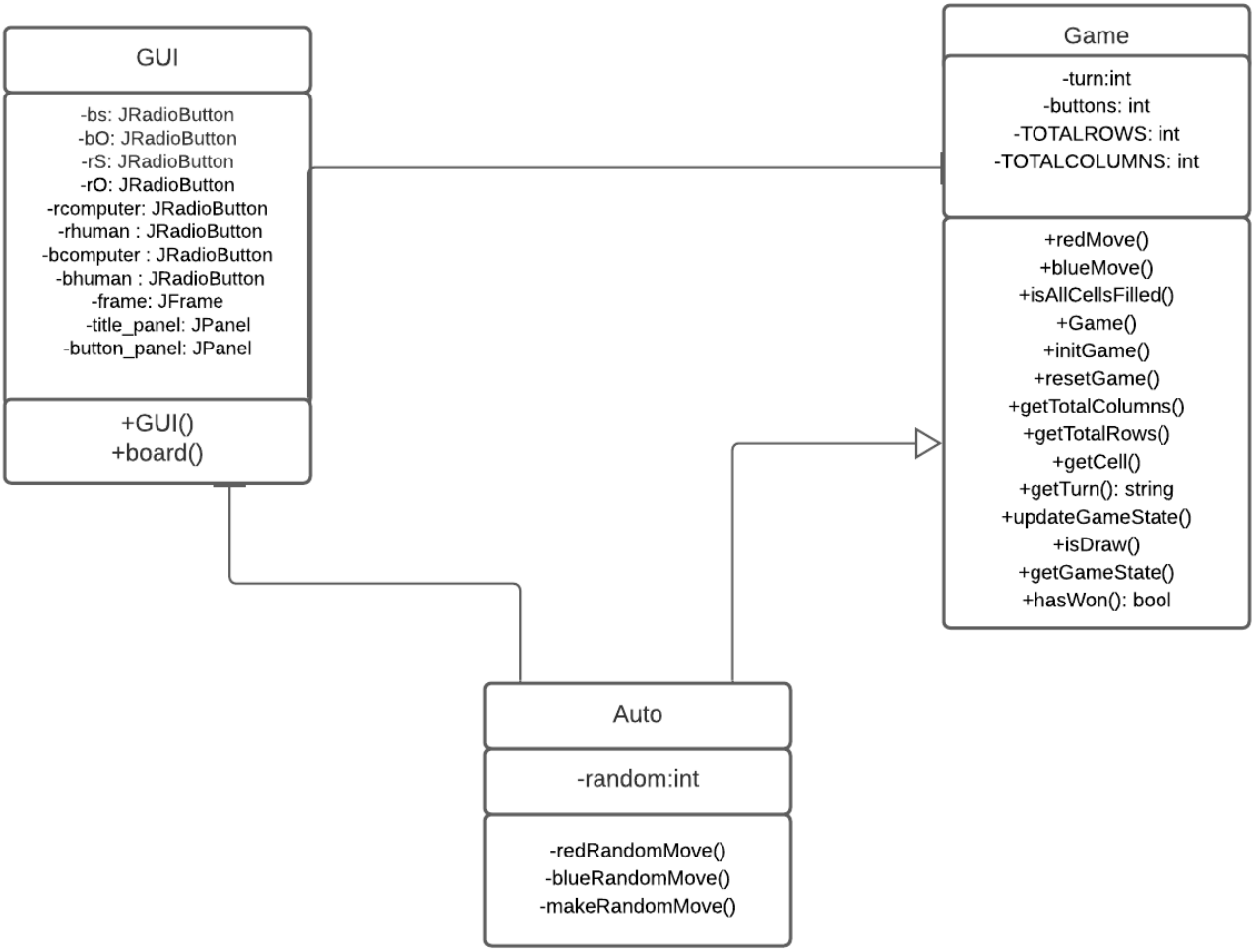
6.2 Manual tests directly corresponding to some acceptance criteria

User Story ID and Name	Acceptance Criterion ID	Test Case Input	Test Oracle (Expected Output)	Notes
1	1.1	board	An empty board	
	1.2		Appropriate GUI	
	…			
2	2.1	actionPerformed	Puts S or O	
	…			
3	3.1	playerRandom	Make random moves	

6.3 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
n/a	n/a	n/a	n/a	n/a

7. **Present the class diagram of your production code (3 points) and describe how the class hierarchy in your design deals with the computer opponent requirements (3 points)**?



Computer opponent requires all blue player options, and also all red player options which include making a sign S or an O on a board. Also to find out if an SOS is made – hasWon(), and make redMoves() and blueMoves(). Most of the requirements that computer opponent has are inherited from the Game class. The auto class (computer opponent class) required some features like making random moves which are totally unique to the Auto class, other than that everything is being inherited from Game class hierarchy. Features like setting up a new game which are shared among both Game and Auto -- provided by GUI, or features like setting up a new game resetGame() , initGame() is provided by Game.