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 computer component must be able to play complete simple and general games. You are encouraged to consider basic strategies for winning simple or general games (e.g., against a poor human player). Optimal play is not required.

The following is a sample GUI layout. You should 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.

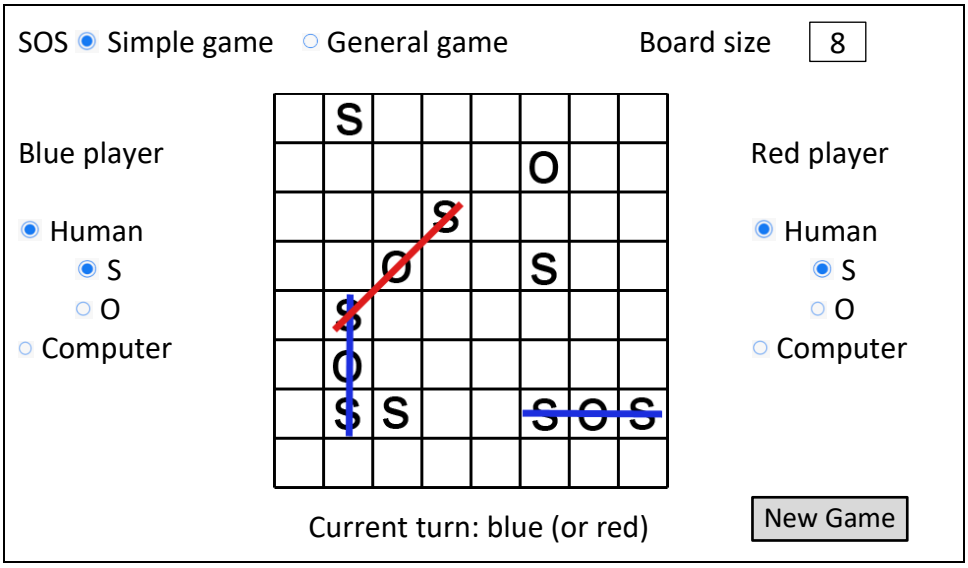


Figure 1. Sample GUI layout of the working program for Sprint 3

Total points: 24

GITHUB REPO: https://github.com/Tim-DeGraffenreid/CS449-Software-Engineering.git

1. **Demonstration (8 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)
8	Simple Computer SOS Game	As a player, I want the computer play a simple game, so I can play a game	1	5
9	General Computer SOS Game	As a player, I want the computer play a general game, so I can play a game	1	5

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

Add or delete rows as needed.

User Story ID and Name	AC ID	Description of Acceptance Criterion	Status (completed, toDo, inPprogress)
8. Simple Computer SOS Game	8.1	AC 8.1 A new simple game against computer Given A new SOS screen When the computer player option is chosen Then a game with the computer begins	complete
	8.2	AC 8.2 A new simple game computer vs computer Given A new SOS screen When both computer player options are chosen Then the computer plays a simple game against itself	
9. General Computer SOS Game	9.1	AC 9.1 A new general game against computer Given A new SOS screen When the computer player option is chosen Then a general game with the computer begins	
	9.2	AC 8.2 A new general game computer vs computer Given A new SOS screen When both computer player options are chosen Then the computer plays a general game against itself	

4. Summary of All Source Code (1 points)

Source code file name	Production code or test code?	# lines of code
Board.java	Production	265
GameFrame.java	Production	620
GeneralBoard.java	Production	73
GeneralComputerGame.java	Production	95
SimpleComputerGame.java	Production	89
BoardTest	Test	379
	Total	1521

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 of the new user story/acceptance criteria 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)
8 Simple Computer SOS Game	8.1	SimpleComputerGame	makeMove, getComputerMove, getComputerLetter,	Complete	
	8.2	SimpleComputerGame	makeMove, getComputerMove, getComputerLetter,	Complete	
9. General Computer SOS Game	9.1	GeneralComputerGame	makeMove, getComputerMove, getComputerLetter,	Complete	
	9.2	GenerealComputerGame	makeMove, getComputerMove, getComputerLetter,	Complete	

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

Summarize how each of the new user story/acceptance criteria 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)
8 Simple Computer SOS Game	8.1	BoardTest	testSimpleGameRedIsComputer	Input: a simple computer game Output: One of the players is the winner
	8.2	BoardTest	testSimpleGameBothComputer	Input: a simple computer game with both players computer Output: A win, loss or draw…sometimes an error
8 Simple Computer SOS Game	p.1	BoardTest	testGeneralGameBlueIsComputer	Input: a general computer game Output: One of the players is the winner
	p.2	BoardTest	testGeneralGameBothComputer	Input: a general computer game with both players computer Output: A win, loss or draw…sometimes an error
	…			

6.2 Manual tests directly corresponding to some acceptance criteria

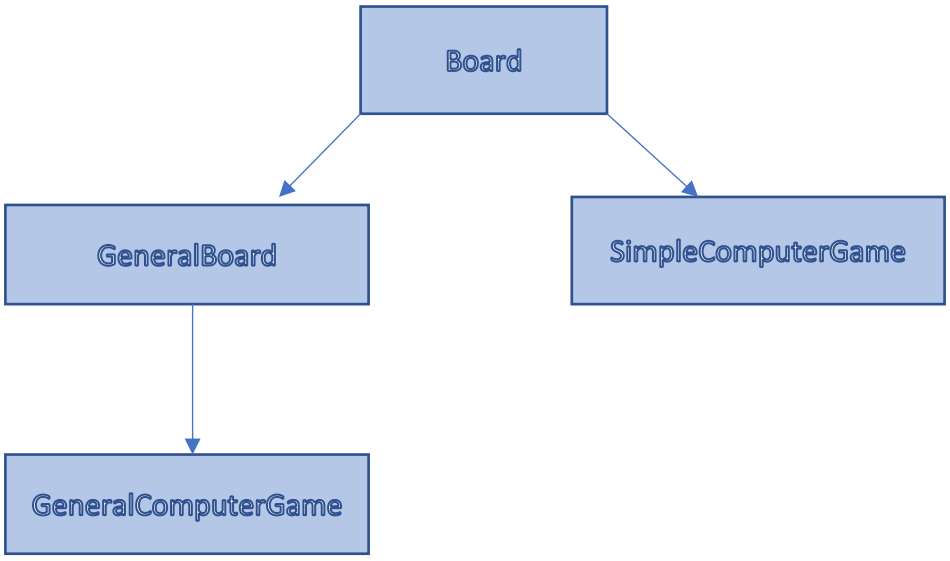
User Story ID and Name	Acceptance Criterion ID	Test Case Input	Test Oracle (Expected Output)	Notes
8 Simple Computer SOS Game	8.1	A simple game (red computer vs blue human)	A human win	
	8.2	A simple game (computer vs computer)	A wind by red or blue	
9 General Computer SOS Game	9.1	A genreal game (red computer vs blue human)	A human win	Sometimes I get an out of range error and can't pinpoint where it is coming from.

	9.2	A general game (computer vs computer)	A wind by red or blue	Sometimes I get an out of range error and can't pinpoint where it is coming from.

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

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)?



The SimpleComputerGame class inherits from the Board class and overrides the makeMove method and I also added two new methods to the SimpleComputerGame class: getComputerMove and getComputerLetter. The getComputerMove uses the Random class to randomly choose a square and then checks to see if it is already occupied. If it is then it picks another random square until it finds one that it can use. The getComputerLetter uses the Random class as well. makeMove then calls on both of the to determine what the computer's move is going to be. Once an SOS is created the game is over.
The GeneralComputerGame is created in a similar fashion with the exception that it waits until all of the squares are filled before declaring a winner. I also added getComputerPlayer and setComputerPlayer to the parent Board class to determine and set which player or both players with be the computer. Everything needs to be refactored, particularly the new subclasses. They have smells on top of smells on top of smells, but I'm twenty minutes from the assignment deadline, and this is the best I have at the moment.