**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.

|  |  |  |
| --- | --- | --- |
| SOS Icon  Description automatically generated Simple game Icon  Description automatically generated General game Board size  8 | | |
| Blue player  Icon  Description automatically generated Human  Icon  Description automatically generated S  Icon  Description automatically generated O  Icon  Description automatically generated Computer | Chart, line chart  Description automatically generated | Red player  Icon  Description automatically generated Human  Icon  Description automatically generated S  Icon  Description automatically generated O  Icon  Description automatically generated Computer |
|  | Current turn: blue (or red) | New Game |

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

**Total points: 24**

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.

1. **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 | Play against computer | As a player, I want to be able to play against a computer so I can practice my skills. | High | 2 |
|  | Computer vs Computer | As a player. I want to be able to watch computers play against eachother so that I can learn from them. | High | 2 |
| .. | Challenge vs computer | As a player, I want the computer opponent to make logical moves so that the game is challenging | Low  (Optional) | 3 – trying to implement |
|  |  |  |  |  |

1. **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, inProgress)** |
| 8 Make a valid Computer Move | 8.1 | AC 8.1 < Play against computer >  Given the game is being played and it is blue computer vs red human  When red player makes a turn valid  Then blue computer should make a valid turn | Completed |
| 8.2 | AC 8.2 < Play against computer >  Given the game is being played and it is blue human vs red computer  When blue human makes a valid turn  Then it should be red computers turn to make a valid turn | Completed |
| 8.3 | AC 8.3 < Computer vs Computer >  Given the game is being played and it is blue computer vs red computer  When blue computer makes a valid turn  Then it should be red computers turn to make a valid turn | Completed |
| 8.4 | AC 8.4 Challenge vs computer  Given the game is being played and it is the computers turn,  When they go to make a move and there are two or more “S” and “O” in a row of my color  Then I should complete that SOS to gain a point. | In progress |
| … |  |  |

1. **Summary of All Source Code (1 points)**

|  |  |  |
| --- | --- | --- |
| Source code file name | Production code or test code? | # lines of code |
| Main.java | Prod | 11 |
| Checker.java | Prod | 504 |
| GUI.java | Prod | 586 |
| GUITesting.java | Test | 593 |
| Total | |  |

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

1. **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 Play against computer | 8.1 | GUI.java | public void actionPerformed(ActionEvent e) {  else if (redHuman.isSelected() && blueComputer.isSelected()) { | Complete | Checks to see when someone else goes with valid input (pressed a non blank button) and will send it on down the code |
| 8 Play against computer | 8.2 | GUI.java | public void actionPerformed(ActionEvent e) {  else if (redComputer.isSelected() && blueHuman.isSelected()) { | Complete |  |
| Computer vs Computer | 8.3 | Gui.java | public void actionPerformed(ActionEvent e) { | Complete |  |
| Challenge vs computer | 8.4 | GUI.java |  | In progress | I know this was optional but it is really good practice. I am trying to implement but it is taking a lot longer than I expected. |

1. **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)** |
| 8.1 | 8.1 | GUITesting.java | public void testRedComputerVsBlueHuman() { | Red computer is playing,  Blue human is playing,  Verifying that turns are working properly and that a winner is selected when there should be a winner. |
|  | 8.2 | GUITesting.java | public void testRedHumanVsBlueComputer() {  setUp(); | Blue computer is playing vs red human. Verifying that turns are working and that a winner is selected when there should be |
|  | 8.3 | GUITesting.java | public void testRedComputerVsBlueComputer() { | Computer vs computer.  Verifying that they take turns and that there is a wnner when their should be |
|  | 8.4 | GUITesting.java | TBD | Still working on this |
|  |  |  |  |  |

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 | 8.1 | Red human gets SOS in simple mode | Red wins |  |
|  | 8.1 | Blue computer gets SOS in simple mode | Blue wins |  |
|  | 8.1 | Red humans gets more SOS in general mode | Red wins |  |
|  | 8.1 | Blue computer gets more SOS in general mode | Blue wins |  |
|  | 8.2 | Red computer gets SOS in simple mode | Red wins |  |
|  | 8.2 | Blue human gets SOS in simple mode | Blue wins |  |
|  | 8.2 | Red computer gets more SOS in general mode | Red wins |  |
|  | 8.2 | Blue human gets more SOS’s in general game mode | Blue wins |  |
|  | 8.3 | Blue computer gets sos in simple mode | blue wins |  |
|  | 8.3 | Red computer gets sos in simple mode | Red wins |  |
|  | 8.3 | Blue computer gets more sos in general mode | Blue wins |  |
|  | 8.3 | Red computer gets more sos in general mode | Red wins |  |

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** |
|  |  | NONE |  |  |
|  |  |  |  |  |

1. **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)**?

My class hierarchy needs work. I am not a strong coder, as this project was very difficult for me. In the real world, I should have a lot of separate classes instead of one big one, let alone a ComputerOpponent class that handles the computer. This would probably inherit? From a player class but implement slightly different logic. You could probably have an interface for them as well which would let the ComputerOpponenct class limited privledges to get required methods. As far as my code goes, the computer opponent requirement hierarchy is inside the GUI class. This makes almost no sense as to why I did it this way but dealing with classes is not my strong suit. You aren’t able to call a class ComputerComponent with mine without going through the GUI class…

A picture containing graphical user interface

Description automatically generated