**Project Sprint #4**

The SOS game is described in CS449HomeworkOverview.docx. You should read the description very carefully.

Your submission must include the GitHub link to your project and you must ensure that the instructor has the proper access to your project. You will receive no points otherwise.

**GitHub link: https://github.com/kimbrow-slice/SOSGame/tree/dev**

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 must 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 4

**Total points: 24**

1. **Demonstration (8 points)**

Submit a link to a video of no more than five minutes, clearly demonstrating that you have implemented the computer opponent features and written some automated unit tests. No points will be given without a video link.

**YouTube/Panopto link:**

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. https://umsystem.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=dae5709d-eb82-44d0-8bd0-b2c300398700

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 | Simple Game - Human vs Computer | As a player, I want to play a complete Simple game where the blue player is human and the red player is the computer, ensuring a winner is clearly declared. | High | 4 |
| 9 | General Game - Computer vs Human | As a player, I want to play a complete General game where the blue player is computer and the red player is human, ensuring a winner is clearly declared. | High | 4 |
| 10 | Simple Game - Computer vs Computer | As a user, I want to watch a complete Simple game fully played by the computer for both blue and red players. | Medium | 3 |
| 11 | General Game - Computer vs Computer | As a user, I want to watch a complete General game fully played by the computer for both blue and red players. | High | 5 |
| 12 | Automated Tests for Computer Opponent | As a developer, I want automated unit tests for the computer opponent logic to ensure reliable AI decisions. | Medium | 3 |

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, inPprogress)** |
| 8 | 8.1 | AC 8.1  Given a Simple game, when blue is human and red is computer, then the game plays to completion with a clear winner. | Completed |
| 9 | 9.1 | AC 9.1  Given a General game, when blue is computer and red is human, then the game plays to completion with a clear winner. | Completed |
| 10 | 10.1 | AC 10.1  Given a Simple game, when both sides are computer-controlled, then the game plays automatically to a clear completion. | Completed |
| 11 | 11.1 | AC 11.1  Given a General game, when both sides are computer-controlled, then the game plays automatically to a clear completion. | Completed |
| 12 | 12.1 | AC 12.1  Given the computer opponent logic, when automated tests are executed, then all decision-making logic passes successfully. | Completed |

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

|  |  |  |
| --- | --- | --- |
| Source code file name | Production code or test code? | # lines of code |
| MainWindow.axaml.cs | Prod | 288 |
| BaseGame.cs | Prod | 188 |
| SimpleGame.cs | Prod | 68 |
| GeneralGame.cs | Prod | 86 |
| GameController.cs | Prod | 150 |
| GridSystem.cs | Prod | 206 |
| HumanPlayer.cs | Prod | 23 |
| ComputerPlayer.cs | Prod | 85 |
| IGamePlayer.cs | Prod | 13 |
| ComputerPlayerTests.cs | Test | 56 |
| HumanPlayerTests.cs | Test | 28 |
| IPlayerTests.cs | Test | 31 |
| GameContollerTests.cs | Test | 69 |
| Total | | 1,291 |

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 | 8.1 | SimpleGame, ComputerPlayer, GameController | MakeMove(), GetNextMoveWithLetter(), TryAutoPlayNext() | Complete | Handles mixed Human/CPU interactions. |
| 9 | 9.1 | GeneralGame, ComputerPlayer, GameController | MakeMove(), GetNextMoveWithLetter(), TryAutoPlayNext() | Complete | Handles mixed CPU/Human interactions. |
| 10 | 10.1 | SimpleGame, ComputerPlayer, GameController | MakeMove(), GetNextMoveWithLetter(), TryAutoPlayNext() | complete | Fully automated Simple Game mode implemented. |
| 11 | 11.1 | GeneralGame, ComputerPlayer, GameController | MakeMove(), GetNextMoveWithLetter(), TryAutoPlayNext() | Complete | Fully automated General Game mode implemented. |
| 12 | 12.1 | ComputerPlayerTests | MakeMove(), GetNextMoveWithLetter(), TryAutoPlayNext() | Complete | Unit tests for AI logic validation. |

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)** |
| 12 Automated Tests for Computer Opponent | 12.1 | ComputerPlayerTests | ChoosesWinningMove\_WhenAvailable(), BlocksOpponent\_WhenTheyCouldWinNext(), PicksRandom\_WhenNoWinningOrBlocking() | AI chooses best move, blocks opponent, or selects randomly as appropriate |
|  | 12.1 | HumanPlayerTests | Constructor\_InvalidLetter\_ThrowsArgumentException(), GetNextMove\_ThrowsException(), GetNextMoveWithLetter\_ThrowsException() | Human player throws exceptions for invalid usage or input |
|  | 12.1 | IPlayerTests | Player\_ImplementsInterface\_AndStoresLetter() | Confirms both HumanPlayer and ComputerPlayer implement IGamePlayer properly |

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 | Human vs CPU (Simple game) scenario played manually | Game finishes clearly declaring winner |  |
| 9 | 9.1 | CPU vs Human (General game) scenario played manually | Game finishes clearly declaring winner |  |
| 10 | 10.1 | CPU vs CPU (Simple game) scenario executed manually | Game completes fully automated, declares winner |  |
| 11 | 11.1 | CPU vs CPU (General game) scenario executed manually | Game completes fully automated, declares winner |  |

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** |
|  |  |  |  |  |
|  |  |  |  |  |

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)**?