Final Design Report (SOS Game)

Andhika Putratama

N11434457

Requirements that have been implemented.

1. **Human vs Human**
   The program has allowed the game to be played by two players, Player 1 and player 2. Both players are inputting their desired coordinate of where to put the letter ‘S’ or ‘O’ in the grid.

2. **Human vs Computer**
   The program has allowed the game to be played against computer. Player 1 will input the coordinate desired and then the computer is trying to prevent the player 1 to win.

3. **3x3 Grid**
   The game operates under 3x3 grid dimension. This is considered an appropriate amount since this is a learning project. More than 3x3 will increase the complexity and potential of the program being crashed.

4. **Validity of movement**
   The program is able to validate whether the prompted coordinate is validated or not. Invalid input will bring the player to the previous state until the coordinate is valid.

Requirements that haven’t been implemented.

1. **Save File**
   Unfortunately, as someone who never studied IT before my coding skill is very limited. Pushing to add features will crash the whole game that is already running really well.

2. **Undo and redo**
   Similar obstacle is the reason undo and redo feature could not be performed in this program.

3. **To.Upper() function**
   The player needs to manually input the value ‘S’ or ‘O’ in capital letter instead of using To.Upper() function to make all input become a capital letter. This is because putting a simple function can give so much bug that makes the program doesn’t work properly. In this case it’s better to sacrifice one feature instead of crashing the whole game

Contribution of Each Team Members

To avoid complexity of workflow and target, I decided to work by myself. This Allow me work as soon as possible without needing to wait or consult anything to anyone. Working alone also allows me to only make one game project instead of two. The decision is made after long consideration and proven to be effective in this situation.

Overall Design and Changes from Preliminary Design

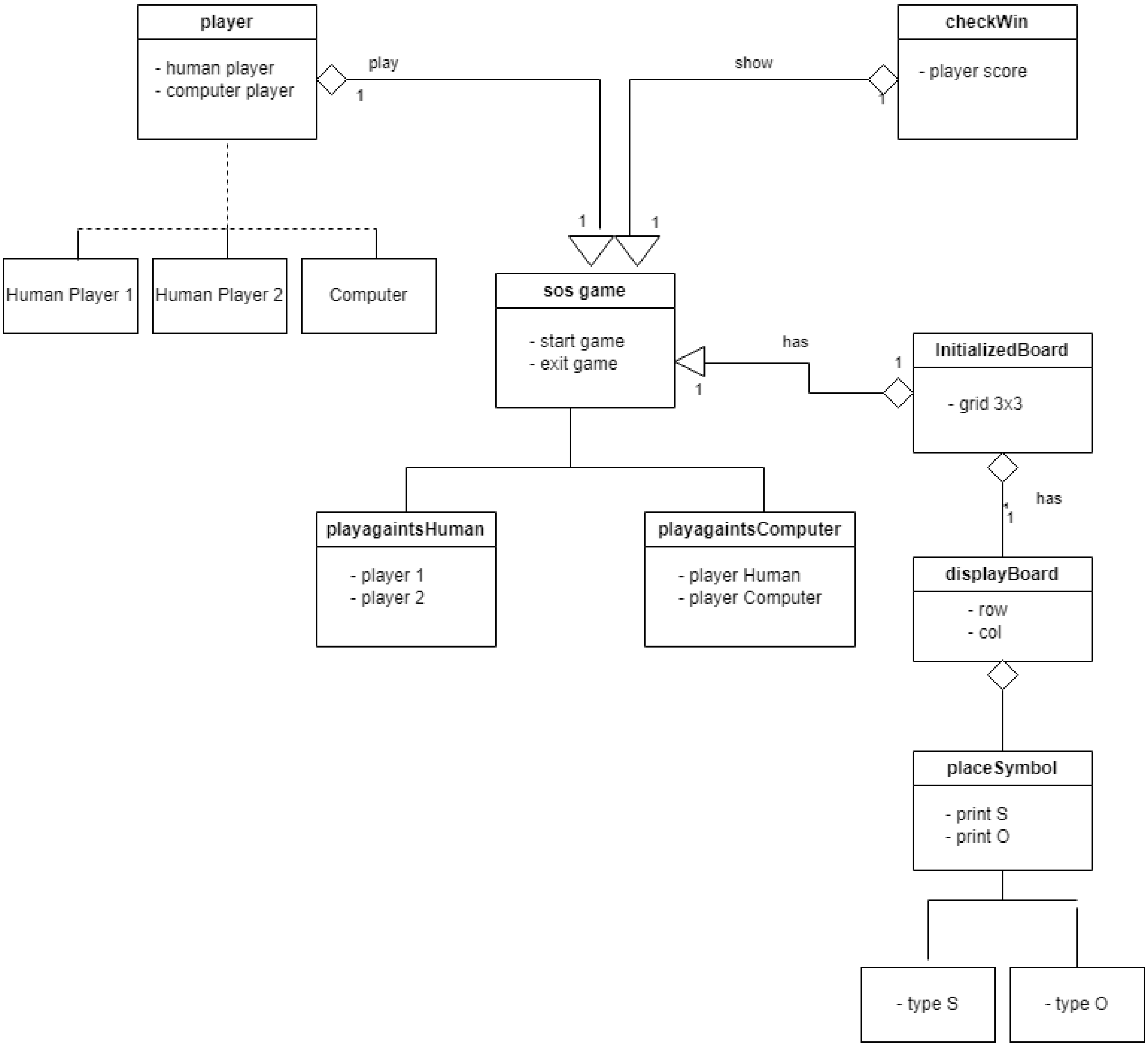
There's not much difference between the preliminary design and the final one. In Class diagram, the class 'menu' is branched out to 2 class "playagainstHuman" and "playagainstComputer", the class "board" has it's name changed into "initializeBoard" for better understanding purposes and has additional class "diplayBoard" that work on the row and column program. Lastly "printBoard" is changed into "placeSymbol" for better explanation.

Object diagram however has more things changed from its previous state. "game" is now become the main "SOS game" that indicates to start or exit the game. "printboard" has become "displayBoard" that prints row, coloum and the players turn in the game."printboard" has it's name changed to "placeSymbol'. On the right side "score" is changed to "checkWin" to notify whether the player or computer who wins.

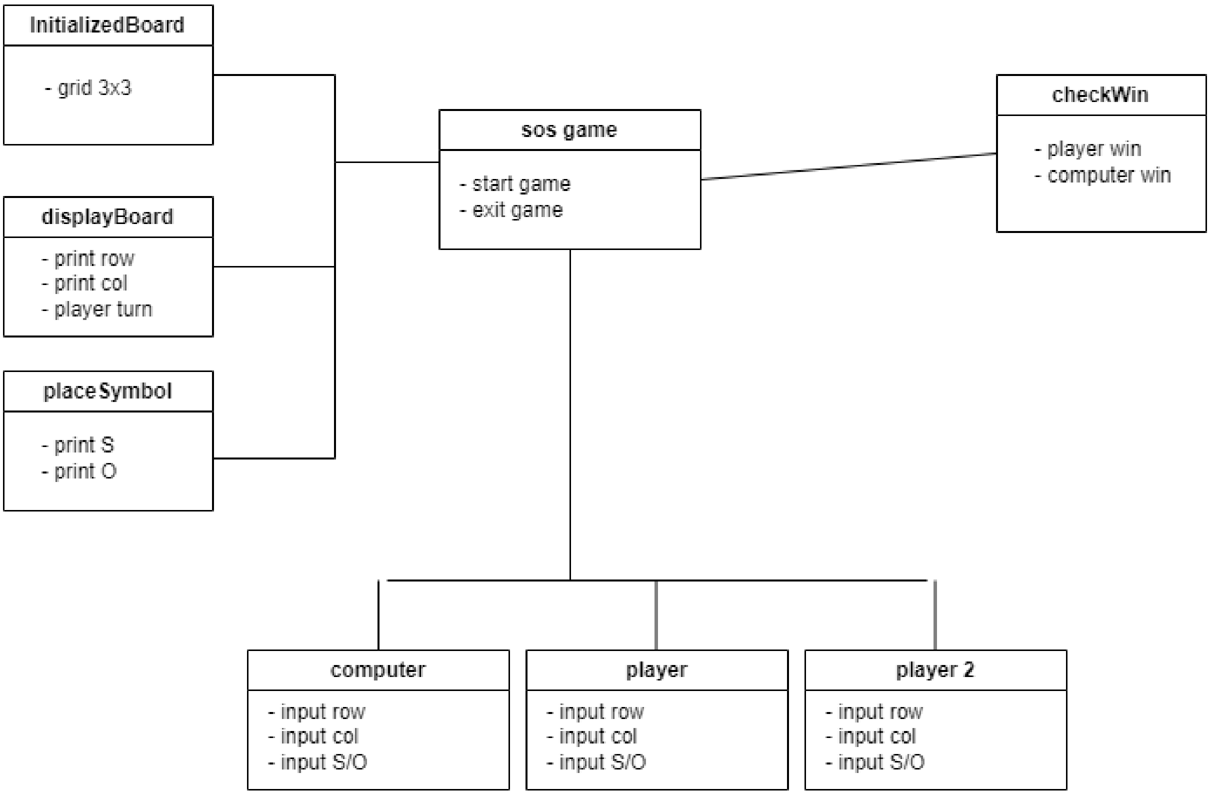
Nothing has changed in the sequence diagram except the name of the class since the sequence of the program is relatively the same from what it's intended to. "Game " is changed to "SOSGame" as in the code. The name is chosen to represent the game that has been coded is an SOS game and not any other game. "PRINTBOARD" has been changed to "DisplayBoard" to match what is written in the code, as well as "PlaceSymbol" and "CheckForWin" replacing "Board" and "Score".

Final Class, object and sequence diagram

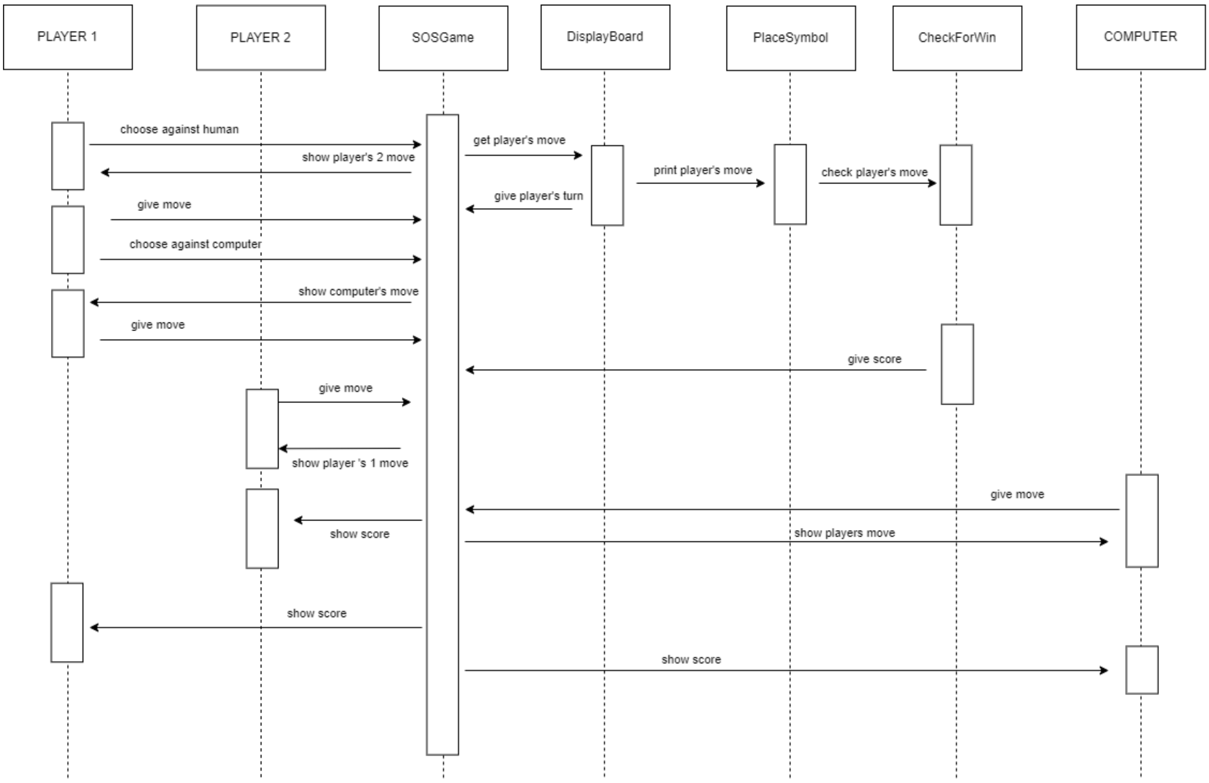
Class Diagram



Object Diagram



Sequence Diagram



How the classes work and important operations

The “Program” class is handling the whole game function. “Main()” function shows the welcome message and showing the menu options that players can choose either playing with another player with computer or just exit the game right away. If the player chooses exit then the program will show thank you for playing message right away.

The “GetMenuChoice” funtion is picking up the input from the player and direct the player to the desired option.” InitializeBoard”, “DisplayBoard” and “PlaceSymbol” are functions that build the grid from empty and then place the coordinate from the player accordingly.

The “CheckForWin” function is used to make sure that one of the player has already reached the state of winning when playing against another player while “CheckForWin2” has similar functionality but works when the player is playing against computer.

The “PlayGameagainstPlayer” function lets two players play the game on the board. It asks the player for the coordinate they desire to put the letter ‘S’ or ‘O’ to be place. It will keep going until one of the player wins and the board is full and it’s a draw. “PlayGameAgaintsComputter” has similar functionality only it works when the player against computer.

How the program can be executed

The program can be executed in a number of ways. First user can open the file “SOS Classic game.csproj” or “SOS Classic game.sln” file that’s located in the “SOS game’” folder. This will direct the user to Microsoft Visual Studio and run the program. Alternatively, the code can be copied to online compiler and to be played there. The game will appear in the console and the player can start the game in there. The player is meant to input the value using according to the instructions which is numerical number and capital letter “S” or “O”. Each player is playing according to their turn while computer player is moving automatically after the player input the instruction in the game. The first player that make the word “SOS” either horizontally, vertically, or diagonally will win. After this process the game is repeated to be back to the main menu.

Classes/interfaces to be reused from existing libraries and frameworks

This Program does not use any existing libraries as it is not necessary. Although it may be used, it will be beyond my novice skill and have potential to crash the program. All the code written is using basic function, logic and syntax.