**Reporte Sprint #3**

Implementen todas las características que permiten a un jugador humano jugar un juego SOS simple o general contra un oponente humano y **refactoricen su código existente si es necesario**. Las características mínimas incluyen elegir el modo de juego (simple o general), elegir el tamaño del tablero, configurar un nuevo juego, hacer un movimiento (en un juego simple o general) y determinar si un juego simple o general ha terminado. El siguiente es un diseño de GUI de muestra.

Se requiere el uso de una jerarquía de clases para hacer frente a los requisitos comunes del juego simple y general. Si tu código para Sprint 2 no ha considerado la jerarquía de clases, es hora de refactorizar su código.

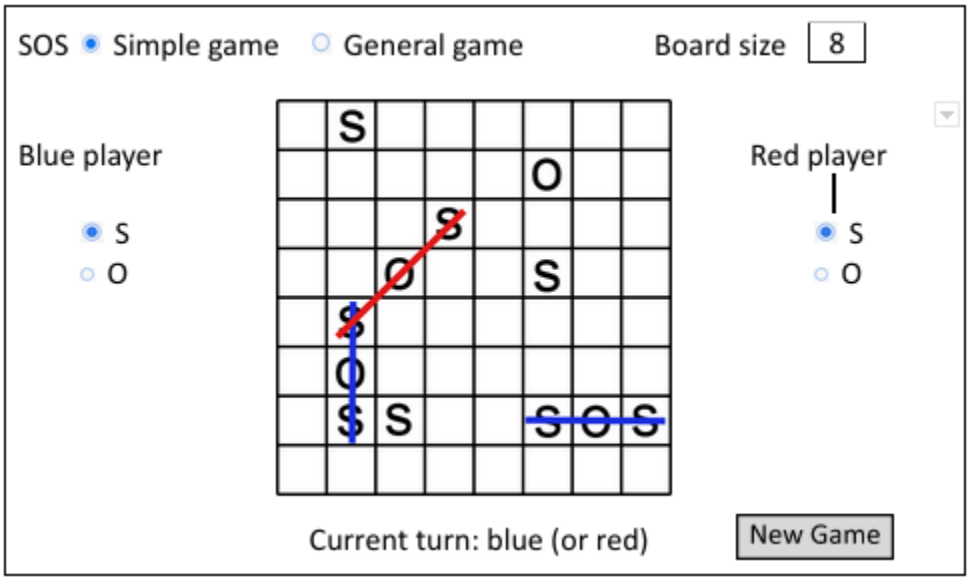


Figura 1. Diseño de GUI de muestra del programa en el Sprint 3

**Entregables: expande y mejora tu entrega para el sprint 2.**

1. **Demostración (6 puntos)**

Envíen un video de no más de cinco minutos, que demuestre claramente las siguientes características.

a) Un juego simple en el que el jugador azul es el ganador.
b) Un juego simple empatado con el mismo tamaño de tablero que es 3
c) Un juego general en el que el jugador rojo es el ganador y el tamaño del tablero es diferente de 3
d) Un juego general empatado con el mismo tamaño de tablero que es 3
e) Algunas pruebas unitarias automatizadas para el modo de juego simple
f) Algunas pruebas unitarias automatizadas para el modo de juego general

En el video, debes explicar lo que se está demostrando.

2. **Resumen del código fuente (2 puntos)**

| Nombre del archivo de código fuente | ¿Código de producción o de prueba? | # lineas de código |
|---|---|---|
| SOSGameBoard | Producción | 142 |
| Console | Producción | 83 |
| SOSGameGUI | Producción | 457 |
| TestSOSGameBoard | Prueba | 63 |
| TestConsole | Prueba | 84 |
| | Total | 829 |

**Deben enviar todo el código fuente para obtener más puntos por esta tarea.**

## 3. Código de producción vs Historias de usuario/Criterio de aceptación (4 puntos)

Resuman cómo se implementa cada uno de los siguientes criterios de aceptación/historia de usuario en tu código de producción (nombre de clase y nombre de método, etc.)

| ID de historia de usuario | Nombre de historia de usuario |
|---|---|
| 1 | Escoge el tamaño del tablero |
| 2 | Escoge el modo de juego de un tablero escogido |
| 3 | Comienza un nuevo juego del tamaño de tablero y del modo de juego elegidos |
| 4 | Hacer un movimiento en un juego simple |
| 6 | Hacer un movimiento en un juego general |

| Nombre y ID de la historia usuario | AC ID | Nombre clase(s) | Nombre Método(s) | Estatus (completo o no) | Notas (opcional) |
|---|---|---|---|---|---|
| 1. Escoge el tamaño de tablero | **AC 1.1<Escoger tamaño del tablero en consola>**<br>**Dado** el usuario antes de iniciar en juego<br>**Cuando** inicie el programa de juego<br>**Entonces** se le debe pedir que ingrese el tamaño deseado del tablero<br>**Y** el tablero debe ser inicializado con ese tamaño | Board | Board(**int** squaresPerSide) | Sí | El tamaño del tablero es pasado al constructor de la clase Board como argumento. |
| | **AC 1.2<Escoger tamaño del tablero en la GUI >**<br>**Dado** visualizado el contenido del juego<br>**Cuando** el jugador quiera cambiar el tamaño del tablero por defecto<br>**Entonces** debe ingresar el tamaño | GameContent<br>GameBoard | addContentTop()<br>drawBoard(int rows, int columns)<br>replace() | Si | El tamaño del tablero por defecto es 3 pero al ingresar un nuevo tamaño y dar enter ente se crea un nuevo tablero vacío del tamaño ingresado |

| | | | | | |
|---|---|---|---|---|---|
| | **Y** el tablero debe ser visualizarce con ese tamaño. | | | | |
| 2.Escoge el modo de juego de un tablero escogido | **AC 2.1<Elegir modo de juego GENERAL/SIMPLE>**<br>**Dado** el usuario antes de iniciar en juego<br>**Cuando** inicie el programa de juego<br>**Entonces** se le debe pedir que ingrese el modo deseado de juego<br>**Y** la condición de victoria será ajustada acordemente. | Console | Console.play() | Sí | El modo de juego es determinado por la variable **isGameSimple** en el método **Console.play().** El modo de juego solamente afecta la condición de victoria. |
| | **AC 2.2<Elegir modo de juego GENERAL/SIMPLE>**<br>**Dado** el usuario al querer realizar un movimiento<br>**Cuando** el juego está iniciado con un tablero por defecto<br>**Entonces** se le pedirá que elija el modo de juego. | GameContent | addContentTop() | Sí | Al elegir el modo de juego ya no se podrá cambiar hasta que inicie otro juego nuevo |
| 3.Comienza un nuevo juego con el tamaño y modo de juego elegidos | **AC 3.1<Iniciar juego en la consola>**<br>**Dado** un usuario que acaba de ingresar el tamaño de tablero deseado<br>**Cuando** ingrese el modo de juego que desea<br>**Entonces** la consola debe imprimir el tablero con el tamaño deseado e iniciar el juego con el turno del jugador 1. | Console | Console.play() | Sí | En la función play() el programa inicializa las variables necesarias para luego entrar al loop del juego en sí. |
| | **AC 3.2<Iniciar juego en la GUI>**<br>**Dado** un usuario que acaba de ingresar el tamaño de tablero y modo de juego deseado<br>**Cuando** presione el botón New Game | GUI | GUI.main() | Sí | El botón NewGame toma inicializa los parámetros del juego. |

| | | | | | |
|---|---|---|---|---|---|
| | **Entonces** GUI debe redimensionar el tablero de juego a lo indicado e iniciar el juego en el modo indicado. | | | | |
| 4.Hacer un movimiento | **AC 4.1<El jugador 1 / 2 hace un movimiento>**<br>**Dado** un juego en curso<br>**Cuando** jugador 1 / 2 ingrese la fila, columna y símbolo deseado<br>**Entonces** dibujará el símbolo elegido en la posición indicada<br>**Y** se cambiará el turno si es que no ha formado un SOS | Board | makePlay(int row, int column, char chosen) | Sí | El método makePlay() conoce el turno actual, pues el turno es un parámetro de la clase Board, de la cual makePlay() es un método. |
| 5.Formar SOS | **AC 5.1<Un jugador forma un SOS en modo GENERAL>**<br>**Dado** un juego curso con la posiblidad de formar un SOS<br>**Cuando** un jugador completa el(los) SOS con el ingreso de una ‘S’ u ‘O’<br>**Entonces** el juego contabiliza la cantidad de SOSs que acaba de formar<br>**Y** se agregan al puntaje del jugador. | Board | hoyManySOS() | Sí | El método **howManySOS()** recibe como argumentos los parámetros de la última jugada e inspecciona cuadrículas adyacentes para contar los SOS formados. El método devuelve un variable tipo INT **points**, que se usará para contabilizar el score de cada jugador. |
| | **AC 5.2<Un jugador forma un SOS en modo SIMPLE>**<br>**Dado** un juego curso con la posiblidad de formar un SOS<br>**Cuando** un jugador completa el(los) SOS con el ingreso de una ‘S’ u ‘O’<br>**Entonces** el juego termina y se designa como ganador al jugador activo. | **Board** | howManySOS() | Sí | Si el modo de juego está en SIMPLE, y además **howManySOS()** devuelve un número mayor que 0, entonces el juego termina. |
| 6. Juego terminado | **AC 6.1<Un juego general termina>** | Board | isGameFinished() | Sí | **isGameFinished()** revisa si alguna celda |

| | | | | | |
|---|---|---|---|---|---|
| | **Dado** un juego curso con solo una casilla libre.<br>**Cuando** el jugador activo llena la última casilla con 'S' u 'O'.<br>**Entonces** el juego termina y se designa como ganador al que haya formado más SOSs | | | | del tablero aún está vacía, si no hay ninguna entonces devuelve **true**; caso contrario, devuelve **false.** |
| | **AC 6.2<Un juego simple termina>**<br>**Dado** un juego curso con la posiblidad de formar un SOS<br>**Cuando** un jugador completa el(los) SOS con el ingreso de una 'S' u 'O'<br>**Entonces** el juego termina y se designa como ganador al jugador activo. | | | | Si el modo de juego está en SIMPLE, y además **howManySOS()** devuelve un número mayor que 0, entonces el juego termina. |

## 4. Pruebas vs Historias de usuario/Criterio de aceptación (4 puntos)

Resuman cómo cada uno de los criterios de aceptación/historia de usuario es probado por su código de prueba (nombre de clase y nombre de método) o pruebas realizadas manualmente.

| User Story ID | User Story Name |
|---|---|
| 1 | Escoge el tamaño del tablero |
| 2 | Escoge el modo de juego de un tablero escogido |
| 3 | Comienza un nuevo juego del tamaño de tablero y del modo de juego elegidos |
| 4 | Hacer un movimiento en un juego simple |
| 6 | Hacer un movimiento en un juego general |

4.2 Pruebas manuales que corresponden directamente a los criterios de aceptación de las historias de usuario anteriores

| **Nombre y ID de la historia usuario** | **AC ID** | **Entrada de caso de prueba** | **Salida esperada** | **Notas** |
|---|---|---|---|---|
| 1. Escoge el tamaño de tablero | **AC 1.1<Tablero de tamaño NxN>** | Una fila y columna válidos | contentBoxes.EMPTY | Esta prueba verifica que el tablero de haya inicializado correctamente |
| 4.Hacer un movimiento | **AC 4.1<El jugador 1 / 2 hace un movimiento>** | testMakePlay()<br>Una fila y columna válida, además un carácter 'S' u 'O' | contentBoxes.Letter_S | Esta prueba verifica que **makePlay()** ponga la letra correcta en la posición correcta. |

| | | | | |
|---|---|---|---|---|
| 5.Formar SOS | **AC 5.1<Un jugador forma un SOS en modo GENERAL>** | testHowManySOS() Una fila y columna válida, además un carácter 'S' u 'O' que completen un SOS | 1 | Verifica que **howManySOS()** cuenta la cantidad de SOSs formados en un turno correctamente. |
| 6. Juego terminado | **AC 6.1<Un juego general termina>** | testIsGameFinished True() La entrada es un tablero lleno. | true | Verifica que **isGameFinished()** verifique si el tablero está lleno de manera esperada. |

5. Describe cómo la jerarquía de clases en tu diseño trata con los requisitos comunes y diferentes del juego simple y el juego general. (4 puntos)

La clase Console y GUI dependen directamente de la clase SOSGameBoard pues usan los métodos de esta numerosas veces. Esto ayuda a independizar la **lógica de los métodos que necesita el juego(Clase SOSGameBoard)** de la **lógica del bucle del juego en sí(clase Console y clase GUI),** además de la construcción de la **interfaz gráfica(clase GUI)**.