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.

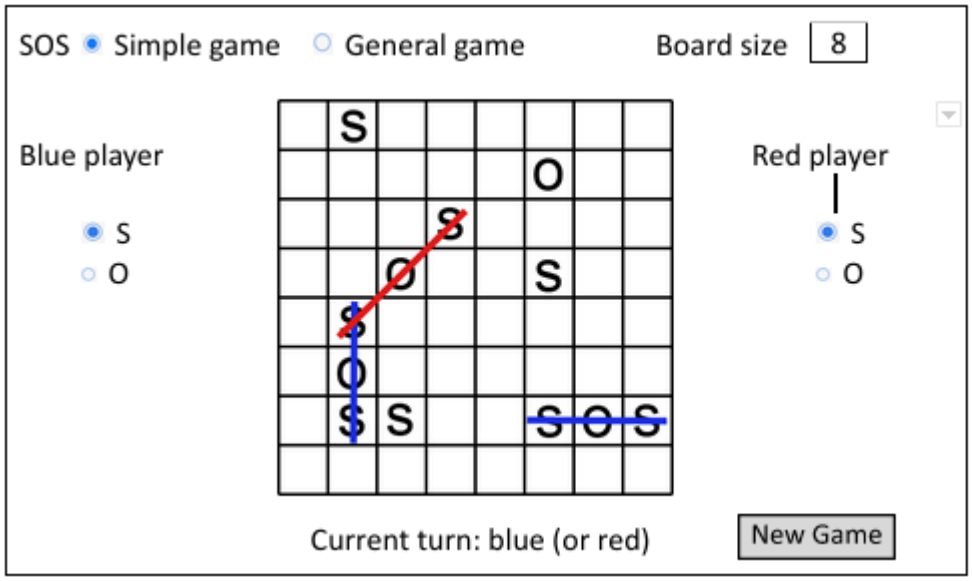

Figura 1. Diseño de GUI de muestra del programa en el Sprint 3

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 1
c) Un juego general en el que el jugador rojo es el ganador y el tamaño del tablero es diferente de 1
d) Un juego general empatado con el mismo tamaño de tablero que es 1
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
Juego.cs	Producción	140
Tablero	Producción	60
ScoreValidator	Producción	61
UnitTest1.cs	Prueba	272
Consola.cs	Producción	43
Form1.cs	Produccion	142
	Total	718

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	1.1	Form1	ReIniciarJuego	Completo	
2	2.1	Form1	GameSelector	Completo	
	2.2	Form1	GameSelector	Completo	
3	3.1	Form1	ShowGameStatus	Completo	
4	4.1	Juego	MakeMove	Completo	
	4.2	Juego	MakeMove	Completo	
5	5.1	ScoreValidator	GameOver	Completo	
	5.2	ScoreValidator	GameOver	Completo	
7	7.1	ScoreValidator	FulBoard	Completo	
	7.2	ScoreValidator	HasWon	Completo	
	7.3	ScoreValidator	HasWon	Completo	

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

4.1 Pruebas automatizadas que corresponden directamente a los criterios de aceptación de las historias de usuario anteriores

Nombre y ID de la historia usuario	**AC ID**	**Nombre Clase (s) del código de prueba**	**Nombre método(s) del código Prueba**	**Descripción de los casos de prueba (entrada & salida esperada)**
1	1.1	TestEmptyBoard	NewTablero	Debe verificar que el tablero esté vacío y sea de tamaño n
2	2.1	TestSelectorModeGame	selectSimpleGameMode	Debe verificar que la variable "juego" se inicialice como un objeto de tipo :JuegoSimple cuando se selecciona esa opción
	2.2	TestSelectorModeGame	selectGeneralGameMode	Debe verificar que la variable "juego" se inicialice como un objeto de tipo :JuegoGeneral cuando se selecciona esa opción.
3	3.1	TestShowGameState	ShowGameState	Debe verificar que el estado del juego se muestre después de que se haya seleccionado el tamaño y el modo de juego
4	4.1	TestMakeMoveSimple	MakeBlueMoveS_SimpleGame	Debe verificar que se marque el movimiento S en una casilla válida para el jugador Azul y que el turno se ceda al siguiente jugador. (juego simple)
	4.2	TestMakeMoveSimple	MakeRedMoveO_SimpleGame	Debe verificar que se marque el movimiento O en una casilla válida para el jugador Rojo y que el turno se ceda al siguiente jugador (juego simple)
5	5.1	TestSimpleGameVictory	VictoryBluePlayerWithS	Debe verificar que el juego termine cuando el jugador Azul forme SOS con un movimiento válido de S y se declare como ganador. (juego simple)
	5.2	TestSimpleGameVictory	VictoryRedPlayerWithO	Debe verificar que el juego termine cuando el jugador Rojo forme SOS con un movimiento válido de O y se declare como ganador. (juego simple)
6	6.1	TestMakeMoveGeneral	MakeBlueMoveO_GeneralGame()	Debe verificar que se hiso un movimiento O del jugador Azul en un juego general
	6.2	TestMakeMoveGeneral	MakeRedMoveS_GeneralGame()	Debe verificar que se hiso un movimiento S del jugador Rojo en un juego general
7	7.1	TestGeneralGameVictory	Fullboard()	Debe verificar que el juego termine cuando se llene el tablero con movimientos válidos .
	7.2	TestGeneralGameVictory	VictoryFullBoardBlue()	Debe verificar que se declare ganador al jugador Azul cuando

				tenga más SOS formados que el jugador Rojo
	7.3	TestGeneralGameVictory	VictoryFullBoardRed()	Debe verificar que se declare ganador al jugador Rojo cuando tenga más SOS formados que el jugador Azul.
	7.4	TestGeneralGameVictory	DrawFullBoards()	Debe verificar que se declare un empate en el juego al haberse llenado el tablero. Cada jugador obtiene la misma cantidad de puntos

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	1.1	Tablero.GetCell(row,colum) para cada celda	0 para cada celda	Tamaño del tablero: 7x7 Un tablero vacío de 7x7 se crea correctamente. Verificando que cada celda este Vacía
2	2.1	juegoSimple.tipoDeJuego	JuegoSimple	El objeto juego es de tipo JuegoSimple
	2.2	juegoGeneral.tipoDeJuego	JuegoGeneral	El objeto juego es de tipo JuegoGeneral
3	3.1	Juego.estadoDeJuego	“JUGANDO”	El juego inicia y muestra correctamente el estado de juego.
4	4.1	Tablero.GetCell(1,1) Tablero.jugador	“S” “Rojo”	Juego en curso, jugador Azul, movimiento S en (1,1) El movimiento S se marca correctamente en la casilla (1,1) y se cede el turno.
	4.2	Tablero.GetCell(2,2) Tablero.jugador	“O” “Azul”	Juego en curso, jugador Rojo, movimiento O en (2,2) El movimiento O se marca correctamente en la casilla (2,2) y se cede el turno.
5	5.1	tablero.Turno tablero.Ficha tablero.EstadoDeJuego	“Azul” “S” “GANOAZUL”	Juego en curso sin SOS, turno de Azul, movimiento S en (1,1) El juego termina y Azul gana al hacer un SOS con S en (1,1).
	5.2	tablero.Turno tablero.Ficha tablero.EstadoDeJuego	“Rojo” “O” “GANOROJO”	Juego en curso sin SOS, turno de Rojo, movimiento O en (2,2) El juego termina y Rojo gana al hacer un SOS con O en (2,2).
6	6.1	tablero.GetCell(0, 2) tablero.Turno	“O” “ROJO”	Debe verificar que se hiso un movimiento O del jugador Azul en un juego general
	6.2	tablero.GetCell(0, 2) tablero.Turno	“S” “AZUL”	Debe verificar que se hiso un movimiento S del jugador Rojo en un juego general
7	7.1	juego.validator.FullBoard()	true	Juego en curso, tablero lleno por movimientos válidos El juego termina correctamente al llenar todo el tablero con movimientos válidos.
	7.2	VictoryFullBoardBlue()	“GANOROJO”	Juego terminado, Azul tiene más SOS hechos que Rojo Azul gana correctamente al tener más SOS hechos que Rojo.
	7.3	VictoryFullBoardRed()	“GANOAZUL”	Juego terminado, Rojo tiene más SOS hechos que Azul Rojo gana correctamente al tener más SOS hechos que Azul.
	7.4	DrawFullBoards()	“EMPATE”	Juego terminado, Rojo tiene más SOS hechos que Azul Rojo gana correctamente al tener más SOS hechos que Azul.

4.3 Otras pruebas automatizadas o manuales que no corresponden a los criterios de aceptación de las historias de usuario anteriores

Número	Entrada prueba	Resultado esperado	Nombre de clase del código de prueba	Nombre del método del código de prueba
7.4	DrawFullBoards()	"EMPATE"	TestGeneralGameVictory	DrawFullBoards()

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)

Según nuestra jerarquía de clases. Usamos la idea de Polimorfismo para poder instanciar un juego que puede tomar dos comportamientos diferentes: Juego Simple y Juego General, cada una con reglas diferentes. Es decir, va tener una cantidad de métodos en cada una de las clases que van a actuar de diferente forma. Entre estas tenemos los métodos: FinalGameState y MakeMove. Para esto con la ayuda de la refactorización Añadimos una clase ScoreValidator que actuaba como operador de las instancias de las demás clases.