**Reporte Sprint #4**

Implementa todas las características que permiten a un jugador (humano o computadora) jugar un juego SOS simple o general contra otro jugador (humano o computadora). Las características mínimas incluyen elegir humano o computadora para jugadores rojos y/o azules, 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 componente informático debe ser capaz de jugar juegos simples y generales completos. Se te anima a considerar estrategias básicas para ganar juegos simples o generales. No se requiere un juego óptimo. El siguiente es un diseño de GUI de muestra. Debes usar una jerarquía de clases para lidiar con los requisitos del oponente de la computadora. Si tu código actual aún no ha considerado la jerarquía de clases, es hora de refactorizar tu código.

|  |  |  |
| --- | --- | --- |
| 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 |

Figura 1. Diseño de GUI de muestra del programa de trabajo para Sprint 3

**Puntos totales: 20**

1. **Demostración (10 puntos)**

Envía un video de no más de cinco minutos, que demuestre claramente que implementaste las funciones del oponente de la computadora y escribiste algunas pruebas de unidad automatizadas.

1. Un juego completamente simple donde el jugador azul es un humano, el jugador rojo es la computadora y hay un ganador.
2. Un juego general completo donde el jugador azul es la computadora, el jugador rojo es un humano y hay un ganador.
3. Un juego simple completo donde ambos lados son jugados por la computadora
4. Un juego general completo donde ambos lados son jugados por la computadora.
5. Algunas pruebas unitarias automatizadas para el oponente de la computadora.

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

1. **Historias de usuario para los requisitos del oponente de la computadora (1 punto)**

**Plantilla de historia de usuario**: Como <rol>, quiero <objetivo> [tal que <beneficio>]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Nombre de historia de usuario** | **Descripción de historia de usuario** | **Prioridad** | **Esfuerzo estimado (horas)** |
| 8 | Escoge el tipo de Jugador | Como Jugador  Quiero escoger un tipo de Oponente  Tal que puedan adaptarse a mis preferencia y habilidades. | BAJA | 0.5 |
| 9 | Jugar contra la máquina | Como Jugador  Quiero tener la opción de jugar contra la maquina  Tal que pueda disfrutar un juego solitario cuando no tenga accesos a internet | ALTA | 3 |

1. **Criterio de aceptación (AC) para los requisitos del oponente de la computadora (1 punto)**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID y nombre de la historia de usuario** | **AC**  **ID** | **Descripción del criterio de aceptación** | **Estado (completado, por hacer, en progreso)** |
| 8 | 8.1 | AC 8.1 <Jugador Azul es una maquina>  Dado que estoy en la pantalla de selección de oponentes  Cuando Selecciono Como jugador Azul como mi oponente y lo configuro como una maquina  Luego el primer jugador es de objeto tipo Maquina | HECHO\* |
| 8.2 | AC 8.2 <Jugador Rojo es una maquina>  Dado que estoy en la pantalla de selección de oponentes  Cuando Selecciono Como jugador Rojo como mi oponente y lo configuro como una maquina  Luego el jugador Rojo es de objeto tipo Maquina | HECHO\* |
| 9 | 9.1 | AC 9.1 < La máquina hace un primer movimiento >  Dado que somos el jugador Rojo y el Jugador Azul es una maquina  Cuando inicie el juego  Luego la maquina hace el primer movimiento | HECHO\* |
| 9.2 | AC 9.2 < La máquina hace el segundo movimiento>  Dado que somos el jugador Azul y el Jugador Rojo es una maquina  Cuando inicie el juego y hago el primer movimiento  Luego la maquina hace el segundo movimiento | HECHO\* |
| 9.3 | AC 9.3 < Dos máquinas se enfrentan y acaban un juego>  Dado un juego simple en curso y dos máquinas se enfrentan  Cuando una de las maquina completa una secuencia ganadora del Juego (SOS) o si quedan empatadas  Entonces el juego termina | HECHO\* |

\*Los métodos funcionan correctamente pero no pudieron ser implementados de forma funcional con la GUI.

1. **Resumen de todo el código fuente (1 punto)**

|  |  |  |
| --- | --- | --- |
| 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 | 355 |
| Consola.cs | Producción | 43 |
| Form1.cs | Produccion | 200 |
| Player | Produccion | 106 |
| Controller.cs | Producción | 36 |
| Total | | 1001 |

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

1. **Código de producción vs Nuevas historias de usuario/Criterios de aceptación (2 puntos)**

Resume cómo se implementa cada una de las nuevas historias de usuario/criterios de aceptación en tu código de producción (nombre de clase y nombre de método, etc.)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID y nombre de la historia de usuario** | **AC ID** | **Nombre(s) clases** | **Nombres(s) métodos** | **Estado (completo o no)** | **Notas (opcional)** |
| 8 | 8.1 | TestSelectTypePlayer | selectBlueComputerPlayer | completo | - |
|  | 8.2 | TestSelectTypePlayer | selectoRedComputerPlayer | completo | - |
| 9 | 9.1 | TestPlayAgainstComputer | ComputerMakesFirstMove | completo | - |
|  | 9.2 | TestPlayAgainstComputer | ComputerMakesSecondMove | completo | - |
|  | 9.3 | TestPlayAgainstComputer | ComputerWinsGame | completo | - |

1. **Pruebas vs Nuevas historias de usuario/Criterio de aceptación (2 puntos)**

Resume cómo tu código de prueba (nombre de clase y nombre de método) o las pruebas realizadas manualmente prueban cada una de las nuevas historias de usuario/criterios de aceptación.

6.1 Pruebas automatizadas que corresponden directamente a algunos criterios de aceptación.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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)** |
| 8 | 8.1 | TestSelectTypePlayer | selectBlueComputerPlayer | Entrada: jugador azul seleccionado como una máquina  Salida esperada: el objeto del jugador azul es una instancia de la clase Computer |
|  | 8.2 | TestSelectTypePlayer | selectoRedComputerPlayer | Entrada: jugador rojo seleccionado como una máquina  Salida esperada: el objeto del jugador rojo es una instancia de la clase Computer |
| 9 | 9.1 | TestPlayAgainstComputer | ComputerMakesFirstMove | Entrada: inicio del juego con la máquina como jugador azul  Salida esperada: la máquina realiza el primer movimiento |
|  | 9.2 | TestPlayAgainstComputer | ComputerMakesSecondMove | Entrada: inicio del juego con la máquina como jugador rojo y el jugador humano haciendo el primer movimiento  Salida esperada: la máquina realiza el segundo movimiento después del movimiento del jugador humano |
|  | 9.3 | TestPlayAgainstComputer | ComputerVsComputer | Entrada: juego simple en curso, ambos jugadores son máquinas y realizan movimientos hasta que llenan el tablero o hay un ganador.  Salida esperada: el juego termina con un estado diferente a Jugando. |

6.2 Pruebas manuales que corresponden directamente a algunos criterios de aceptación.

NOTA: No hay pruebas manuales pues no se ha logrado conectar la GUI con los métodos que ejecutan los jugadores cuando son diferentes del caso Humano-Humano. Por este motivo solo se han escrito pruebas automáticas.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre y ID de la historia usuario** | **AC**  **ID** | **Entrada de caso de prueba** | **Salida esperada** | **Notas** |
| 8 | 8.1 |  |  |  |
|  | 8.2 |  |  |  |
| 9 | 9.1 |  |  |  |
|  | 9.2 |  |  |  |
|  | 9.3 |  |  |  |

6.3 Otras pruebas automatizadas o manuales que no correspondan a los criterios de aceptación

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Número** | **Entrada de prueba** | **Salida esperada** | **Nombre de clase del código de prueba** | **Nombre del método del código de prueba** |
|  |  |  |  |  |
|  |  |  |  |  |

1. Presenta el diagrama de clases de tu código de producción y describe cómo la jerarquía de clases en su diseño trata con los requisitos del oponente de la computadora (3 puntos)?

