

Informe Sprint #3

Nombre del equipo: Development Team

Información dada por el equipo del Proyecto		A ser usado por el profesor	
NombreEstudiante	Contribuciones específicas para este Sprint	Puntaje Equipo	Puntaje Individual
Aguilar Mario	Resumen de código fuente ,Documentación de diseño		
Pumayalli Kevin	Documentación de diseño, Actualización de las tareas de implementación		
Rosales Julio	Actualización de criterios de aceptación, Documentación de diseño, Resumen de código fuente.		
Unda Carlos	Actualizaciones de Historias de Usuario,Actualización de las tareas de implementación,Documentación de Diseño		

I. Actualización de las historias

ID	Nombre Historia de Usuario	Descripción Historia Usuario	Prioridad	Esfuerzo	Esfuerzo real	Status	Nombre desarrollador
1	Configuración del juego	Como usuario del juego necesito seleccionar si deseo jugar con otro jugador o con la máquina luego colocar el nombre para poder empezar el juego	Prioridad Alta	1h	2h30 min	Hecho	Pumayalli Kevin
2	Comenzar un juego	Como jugador, necesito un tablero vacío para comenzar un juego Nine Men's Morris	Prioridad Media	1h30min	4h	Hecho	Pumayalli Kevin
3	Colocar una ficha (Usuario -Computador)	Como jugador/Computador, necesito colocar una ficha, para que termine mi turno.	Prioridad Media	2h30min	5h	Hecho	Unda Carlos
4	Mover una ficha (Usuario-Computador)	Como jugador/Computador, necesito mover una ficha, para intentar formar una línea de 3 fichas.	Prioridad Alta	2h	6h	Hecho	Aguilar Mario Rosales julio
5.-	Posibles Movimientos Permitidos	Como jugador/Computador,necesito tengo opciones para intentar colocar una ficha en el tablero	Prioridad Medi	2h	3h	Hecho	
6	Remover ficha del oponente (Usuario-Computador)	Como jugador/Computador necesito formar una fila de 3 fichas linealmente ordenada para poder quitarle una ficha al contrincante y así tenga menos fichas.	Prioridad Alta	4h	7h	Hecho	Aguilar Mario Rosales julio
7	Volar una ficha (Usuario-Computador)	Como jugador/Computador , necesito saber si tengo 3 fichas únicamente en el tablero para poder volar.	Prioridad Media	5h	8h	Hecho	Aguilar Mario Rosales Julio
8	Terminar el juego (Usuario-Computador)	Como jugador/Computador necesito saber si yo o el contrincante tiene 2 fichas para que el juego termine.	Prioridad Media	4h	6h	Hecho	Mario Aguilar Rosales julio

II. Actualización de los criterios de aceptación de Player vs Player(AC) o Player vs PC

ID Historia de Usuario y Nombre	AC ID	Descripción de los criterios de aceptación	Status	Nombre desarrollador
1.-Configuración del juego	AC1.1 Preparar el juego Dado mi solicitud de jugar Cuando escojo el adversario (PC o Player) Y digito mi nombre y me asignan el color Entonces puedo empezar a jugar.	Como el software ya me dio acceso ,se solicita escoger oponente ,puede ser otro usuario o la misma pc	Hecho	Unda Carlos Pumayalli Kevin
	AC1.2 Turno de juego Dado que puse mi nombre Cuando me asignan el color de fichas Entonces sabré cual es mi turno para empezar el juego	Como nuevo usuario del juego necesito poner mi nombre y escoger un color de ficha para poder saber si empiezo primero o segundo	Hecho	Unda Carlos Pumayalli Kevin
2.- Comenzar el juego	AC2.1 Tablero vacío Dado una configuración previa del juego Cuando se inicia el juego Entonces el sistema muestra un tablero vacío y fichas a su disposición.	Dado que ya tengo permiso para jugar entonces me dan un tablero vacío y fichas disponibles para jugar iniciando el juego	Hecho	Pumayalli Kevin Unda Carlos Rosales julio Aguilar Mario
3.- Colocar una ficha (Usuario-Computador)	AC3.1 Colocar una ficha en una casilla libre Dado el turno de un jugador/Computador Cuando coloque una ficha en una posición libre del tablero Entonces el sistema debería colocar la ficha.	Como es mi turno, ademásuento con fichas fuera del tablero y fichas en el tablero con algún espacio vacío aledaño a estas,entonces coloco una ficha en un espacio libre	Hecho	Unda Carlos Aguilar Ybarra
	AC3.2 Colocar una ficha en una casilla ocupada Dado el turno de un jugador/Computador Cuando coloque una ficha en una posición ocupada Entonces el sistema no debería colocar la ficha en esa ubicación.	Como es mi turno, ademásuento con fichas fuera del tablero y fichas en el tablero necesito verificar si la casilla está ocupada o no para hacer un movimiento	Hecho	Unda Carlos Aguilar Ybarra
	AC3.3 Colocar una ficha fuera del tablero Dado el turno de un jugador/Computador Cuando coloque una ficha fuera del tablero Entonces el sistema no debería colocar la ficha en esa ubicación.	Como es mi turno, ademásuento con fichas fuera del tablero y fichas en el tablero necesito saber en qué lugar del tablero se puede mover una ficha y no fuera de este.	Hecho	Unda Carlos Aguilar Ybarra
	AC3.4 Colocar una ficha en una posición no permitida Dado el turno de un jugador/Computador Cuando coloque una ficha dentro del tablero que no sea una intersección. Entonces el sistema no debería colocar la ficha en esa ubicación.	Como es mi turno, ademásuento con fichas fuera del tablero y fichas en el tablero necesito verificar que la casilla esté aledaña a la ficha que colocaré y esté definida.	Hecho	Unda Carlos Aguilar ybarra

4.- Mover una ficha(Usuario-Computador)	<p>AC4.1 Movimiento permitido de una ficha</p> <p>Dado que el jugador/Computador mueve la pieza</p> <p>Cuando se haya movido la pieza en intersecciones permitidas.</p> <p>Entonces el sistema debe indicar que es el turno del contrincante.</p>	Como es turno del jugador/Computador y tengo fichas disponibles para hacer un movimiento adyacente a las fichas que se usará.	Hecho	Rosales julio Aguilar Mario
	<p>AC4.2 Movimiento de una ficha no permitida</p> <p>Dado que el jugador/Computador mueve la pieza</p> <p>Cuando se haya movido la pieza en intersecciones no permitidas.</p> <p>Entonces el sistema debe indicar que el movimiento no fue válido.</p> <p>Y le debe indicar que debe realizar un movimiento válido.</p>	Como es turno del jugador/Computador y tengo fichas disponibles para hacer un movimiento adyacente a las fichas que se usará y no en un casillero donde esté ocupado por otra ficha .	Hecho	Rosales julio Aguilar Mario
5.-Posibles Movimientos Permitidos	<p>AC5.2 Movimiento de una ficha en posibles casilleros opcionales</p> <p>Dado que el jugador/Computador mueve la pieza a un casillero</p> <p>Cuando se haya colocado la pieza</p> <p>Entonces el sistema debe indicar las posibles casillasopcionales.</p>	Como es turno del jugador/Computador y se tiene casillerosopcionales posibles se deberá colocar en una de ellas. .	Hecho	Rosales julio Aguilar Ybarra
6.- Remover ficha (Usuario-Computador)	<p>AC6.1 Remover ficha del tablero</p> <p>Dado que el jugador/Computador formó una terna de fichas alineadas.</p> <p>Cuando el jugador seleccione la ficha a robar</p> <p>Entonces el sistema debe remover la ficha al adversario.</p>	En mi turno he logrado colocar una línea de tres fichas de mi color escogido,por lo cual remuevo una ficha contraria pero considerando que esta ficha del adversario no esté formando molino.	Hecho	Rosales julio Aguilar Mario
	<p>AC6.2 Remover una ficha fuera del tablero</p> <p>Dado que el jugador/Computador formó una terna de fichas alineadas.</p> <p>Cuando el jugador/Computador intenta remover la ficha del adversario fuera del tablero</p> <p>Entonces el sistema no le permite remover la ficha</p> <p>Y le debe indicar que debe remover una ficha del tablero.</p>	En mi turno he logrado colocar una línea de tres fichas de mi color escogido,por lo cual tengo que evitar remover una ficha fuera del tablero porque el juego no me dejará hacerlo	Hecho	Rosales julio Aguilar Mario
7.- Volar una ficha(Usuario-Computador)	<p>AC7.1 Volar una ficha</p> <p>Dado que el jugador/Computador pierde fichas</p> <p>Cuando el jugador/Computador dispone de 3 únicas fichas sobre el tablero</p> <p>Entonces el sistema le permite el vuelo al jugador/Computador, en el cual las fichas saltan libremente a cualquier punto vacío del tablero.</p>	Como jugador/Computador , necesito saber si tengo 3 fichas únicamente en el tablero para poder tener opción de volar,osea poder desplazar una ficha a lo largo de una línea si están e incluso por encima de otras al espacio vacío.	Hecho	Rosales julio Aguilar Mario
8.- Terminar el Juego(Usuario-Computador)	<p>AC8.1 Solo tener 2 fichas</p> <p>Dado que un jugador/Computador pierde una ficha</p> <p>Cuando al jugador/Computador le quedan únicamente 2 fichas</p> <p>Entonces el sistema indica que el juego terminó.</p> <p>Y también indica el nombre del ganador.</p>	Como jugador/Computador necesito saber si yo o el contrincante tiene 2 fichas para que el juego termine.	Hecho	Rosales julio Aguilar Mario

	AC8.2 No tener opción a moverse Dado el turno de un jugador/Computador Cuando realice el movimiento Y haya restringido por completo los movimientos del otro jugador. Entonces el sistema indica que el juego terminó. Y también indica el nombre del ganador.	Como jugador/Computador necesito saber si yo o el contrincante, aun tenemos opción a mover alguna ficha que está en el tablero, en caso contrario ,alguno habrá perdido.	Hecho	Rosales julio Aguilar Mario
	AC8.3 Empatar por hacer las mismas jugadas reiteradamente Dado que un jugador/Computador hace un movimiento repetitivamente Cuando haya llegado a los 50 movimientos repetidos Entonces el sistema indica que el juego terminó indicando el empate.	Como jugador/Computador necesito saber que si hago el mismo movimiento reiteradamente el juego terminará en empate.	Hecho	Rosales julio Aguilar Mario

III. Actualización de las tareas de implementación

Resumen del código de producción.

ID Historia de Usuario y Nombre	AC ID	Clases	Nombre método(s)	Nombre desarrollador	Status
1.-Configuración del juego	1.1	Configuracion UsuarioB UsuarioN	getIconImage() initComponents() jButtonCloseActionPerformed() jButtonPlayerPlayerActionPerformed() jButtonPlayerPcActionPerformed() initComponents() jButtonCloseActionPerformed() jButtonContinuarActionPerformed() jTextFieldNombreActionPerformed() initComponents() jButtonCloseActionPerformed() jButtonContinuarActionPerformed() jTextFieldNombreActionPerformed()	Pumayalli Kevin	Hecho
	1.2	UsuarioB UsuarioN BoardGUI	initComponents() jButtonCloseActionPerformed() jButtonContinuarActionPerformed() initComponents() jButtonCloseActionPerformed() jButtonContinuarActionPerformed() setEngine() setNombreBlanco() setNombreNegro() getPositionCoords() paintComponent() mouseClicked()	Pumayalli Kevin	Hecho
2.- Comenzar el juego	2.1	UsuarioB Board Debug BoardGUI	jButtonContinuarActionPerformed() printBoard() fillAll() interactive() remove() place()	Pumayalli Kevin Unda Carlos Rosales julio Aguilar Mario	Hecho

		MainGUI	move() paintComponent() actionPerformed()		
3.- Colocar una ficha	3.1 3.2 3.3 3.4	Cell Player Board Engine EngineAI	inPlaceMode() placeRandom() evalPlace()	Unda Carlos Pumayalli Kevin	Hecho
4.- Mover una ficha	4.1 4.2	Cell Player Board Engine EngineAI	inMoveMode() moveRandom() evalMove() evalMoveInverse()	Rosales julio Aguilar Mario	Hecho
5.- Remover ficha	5.1 5.2	Cell Player Board Engine EngineAI	inRemoveMode() removeRandom() evalRemove()	Rosales julio Aguilar Mario	Hecho
6.- Volar una ficha	6.1	Cell Player Board EngineAI	flyRandom() evalFly() evalFlyInverse()	Rosales julio Aguilar Mario	Hecho
7.- Terminar el Juego	7.1	Cell Player Board Engine	gameOver()	Rosales julio Aguilar Mario	Hecho

Resumen del código de prueba automatizado

ID Historia de Usuario y Nombre	AC ID	Nombre de clase de prueba	Nombre del método de prueba	Descripción del caso de prueba (entrada y salida esperada)	Status	Nombre desarrollador
2.Comenzar el Juego	2.1	BoardTest	testprintBoard()	La prueba pasa satisfactoriamente si se pintan las posiciones válidas del tablero.	testeo valido	Unda Carlos
		BoardTest	testOwnerShip()	La prueba pasa satisfactoriamente si el dueño de la casilla marcada obtenida coincide con el dueño esperado	testeo valido	Unda Carlos
3.-Colocar una ficha	3.1 3.2 3.3 3.4	EngineTest	testinPlaceMode()	La prueba testinPlaceMode() pasa satisfactoriamente cuando hay un lugar vacío en el tablero para colocar ficha.	testeo valido	Unda Carlos
		BoardTest	testColocacionValida()	La prueba pasa satisfactoriamente cuando una celda con una posición específica está ocupada.	testo valido	Unda Carlos
4.- Mover una ficha	4.1	EngineTest	testinMoveMode()	La prueba testinMoveMode() pasa satisfactoriamente cuando un jugador mueve una ficha.	testeo valido	Rosales julio
5.- Remover ficha	5.1	EngineTest	testinRemoveMode()	La prueba testinRemoveMode() pasa satisfactoriamente si aún existen fichas en las celdas para remover.	testeo valido	Rosales julio

		BoardTest	testFilaMillFormation()	La prueba pasa satisfactoriamente si se forma un molino horizontalmente, comparando la cadena de celdas que retorna al agregar otra celda , con la cadena esperada.	testeo valido	Rosales julio
		BoardTest	testColumnaMillFormation()	La prueba pasa satisfactoriamente si se forma un molino verticalmente, comparando la cadena de celdas que retorna al agregar otra celda , con la cadena esperada.	testeo valido	Rosales julio

Resumen de casos de prueba manual

ID Historia de Usuario y Nombre	AC ID	Entrada de casos de prueba	Prueba Oráculo (salida esperada)	Status	Notas	Nombre desarrollador
1.-Configuración del juego	1.1.	java.awt.event.ActionEvent evt	JFrame UsuarioB	Hecha	Ya está implementado una acción al botón Player VS PC para dirigir al otro JFrame UsuarioB.	Unda Carlos
	1.1	java.awt.event.ActionEvent evt	null	Hecha	Se guarda correctamente la lectura del JTextField para guardar un nombre y Mostrarlo en la interfaz de Tablero	Pumayalli Kevin

IV. Resumen del código fuente

Código	Nombre del archivo de código fuente	# lineas código	Nombre desarrolladores y contribuciones (% de código fuente)
produccion	Configuracion.java UsuarioB.java UsuarioN.java	145 185 162	Pumayalli Kevin (Configuracion - 100%, UsuarioB - 100% UsuarioN - 100%)
produccion	Board.java Cell.java Player.java	586 76 182	Pumayalli Kevin(Board - 25%, Cell - 15%, Player - 25%) Aguilar Ybarra(Board - 30%, Cell-25%, Player.20%) Rosales Julio(Board-25%, Cell-30%, Player-25%) Unda Carlos(Board-20%,Cell-30%,Player-30%)
produccion	Debug.java Engine.java EngineAI	57 183 557	Pumayalli Kevin(Debug - 25%, Engine -15%, EngineAI - 15%) Rosales Julio(Debug-25%, Engine-35%, EngineAI-35%,) Aguilar Ybarra (Debug-25%, EngineAI-15%, Engine-15%) Unda Carlos(Debug - 25%,Engine-35%,EngineAI-35%)
produccion	BoardGUI MainGUI	373 135	Pumayalli Kevin (BoardGUI - 20%, MainGUI - 30%) Rosales Julio(BoardGUI-25%, MainGUI- 20%,) Unda Carlos(BoardGUI - 30%, MainGUI - 25%) Mario Aguilar (BoarGUI-25%, MainGui-25%)
Prueba	BoardTest.java Engine.Test	107 36	Rosales julio(BoardTest-60%, EngineTest-50%) Aguilar ybarra Mario(BoardTest-40%, EngineTest-50%)
Total		2784	

V. Documentación de diseño

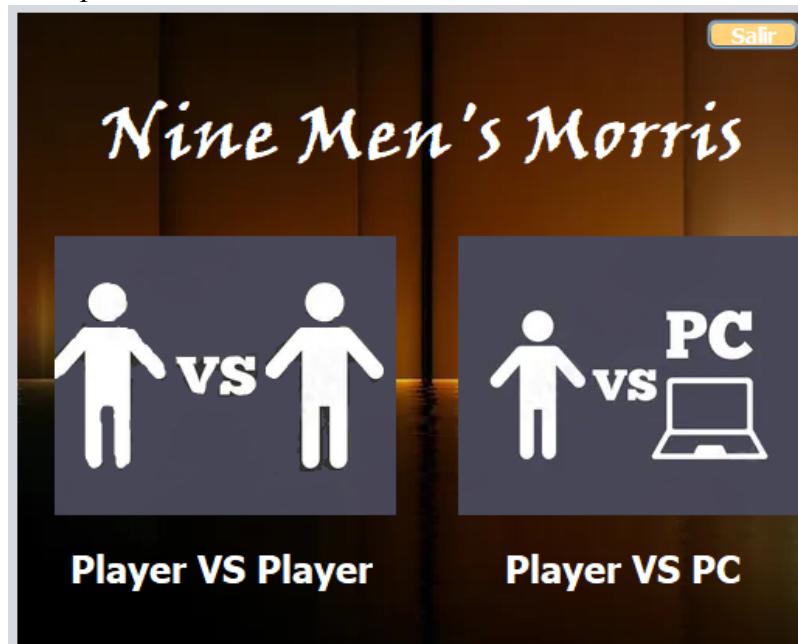
1. Diseño de interfaz de usuario

- A. Carlos Sinai Unda Miguel
- B. Pumayalli Quispe Kevin

Resumiendo los diseños de las interfaces que se muestran cuando se ejecuta el juego:

1. Interfaz de configuración del juego.

En esta interfaz de configuración de juego, nos da la opción de escoger si el usuario desea jugar con otro usuario (Player vs Player) o jugar contra la computadora (Player vs PC), para cada opción que elija el usuario se ejecutará una serie de interfaces posteriores.



2. Interfaz del Jugador

2.1 Jugador Blanco

En ésta interfaz del Jugador Blanco, le pide al usuario ingresar su nombre (en este caso “Pancho”) y le da opciones si quiere salirse del juego o si quiere continuar(en este caso te direcciona a la interfaz del jugador Negro).

Nota; En caso de que el usuario haya seleccionado la opción de Player vs PC en la interfaz de configuración de juego entonces al momento de dar el botón Continuar, te mostrará la interfaz del Tablero de Juego.



2.2 Jugador Negro

En esta interfaz llamada Jugador Negro, se le pide al usuario, en este caso el jugador P2, que ingrese su nombre (en este caso “Juan”), y en esta ventana al igual que la anterior para Jugador Blanco, le damos la opciones de salirse del juego o en caso contrario quiera continuar con este (si es así lo direcciona a la interfaz del Tablero de Juego).

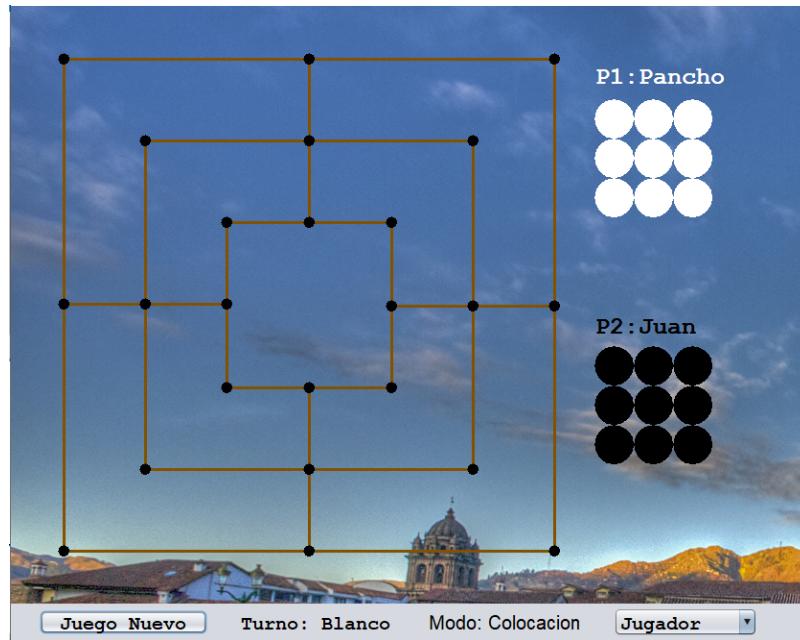
NOTA: Si en caso el usuario haya seleccionado la opción Player vs PC en la interfaz de configuración de juego, entonces esta interfaz de Jugador Negro ya no se mostraría y pasaría de frente a ejecutarse la interfaz del Tablero de Juego.



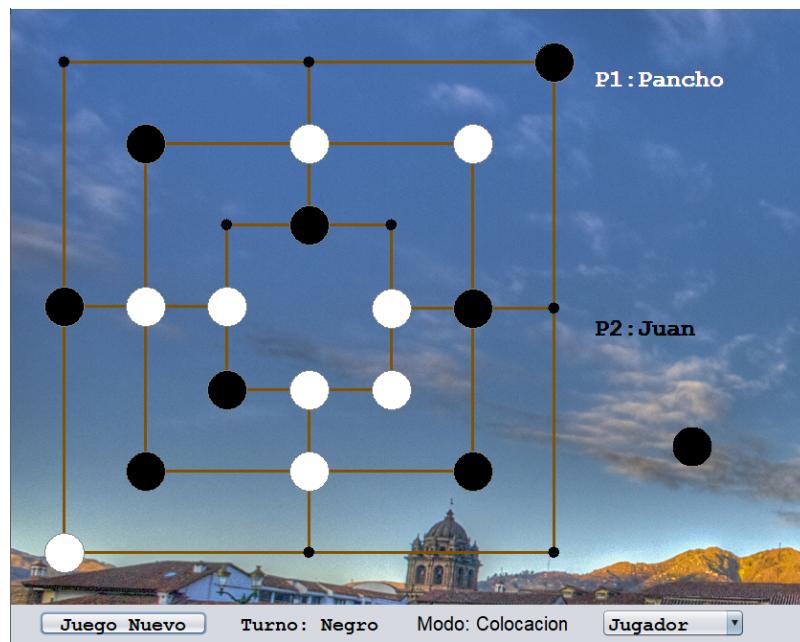
3. Interfaz del Tablero de Juego

3.1 Modo de juego Player vs Player

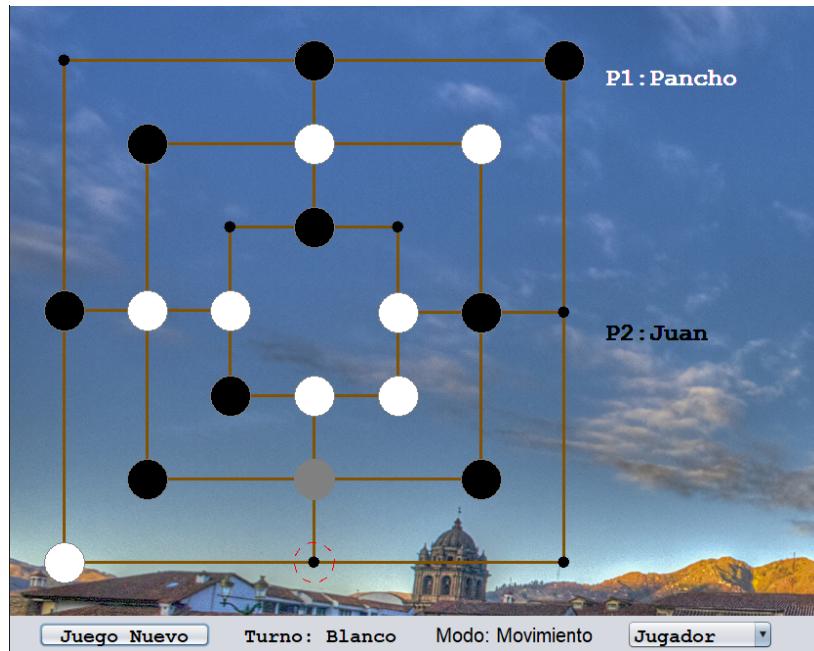
En esta Interfaz de Tablero de Juego, nos muestra los nombres de los jugadores con sus respectivas fichas empezando el turno por el primer jugador P1 (Jugador Blanco, que lo llamamos “Pancho”) que tiene la fichas blancas y luego tambien tenemos al segundo jugadro P2 (Jugador Negro, que lo llamamos “Juan”), que tiene las fichas negras. Esta interfaz tambien nos muestra un botón que sirve como reset, que lo llamamos Juego Nuevo, por si cualquier jugador quisiera volver a empezar nuevamente la partida. Por ultimo vemos que el primer turno es para el P1(Jugador Blanco) y que el modo de juego en que nos encontramos es el de Colocación.



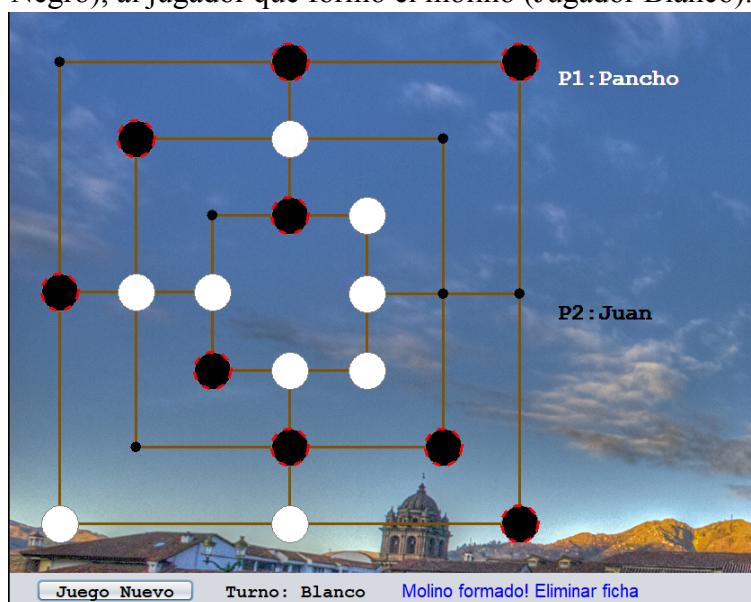
Aqui vemos que, tanto el jugador P1 (Jugador Blanco, que lo llamamos “Pancho”) que tiene la fichas blancas ,asi como el jugador P2 (Jugador Negro, que lo llamamos “Juan”), que tiene las fichas negras, colocaron la mayoria de sus fichas, dentro del tablero, en las posiciones definidas. Señalar que previamente todas las fichas estaban fuera del tablero. Todo esto ocurre ya que nos encontramos en el modo de juego de Colocación.



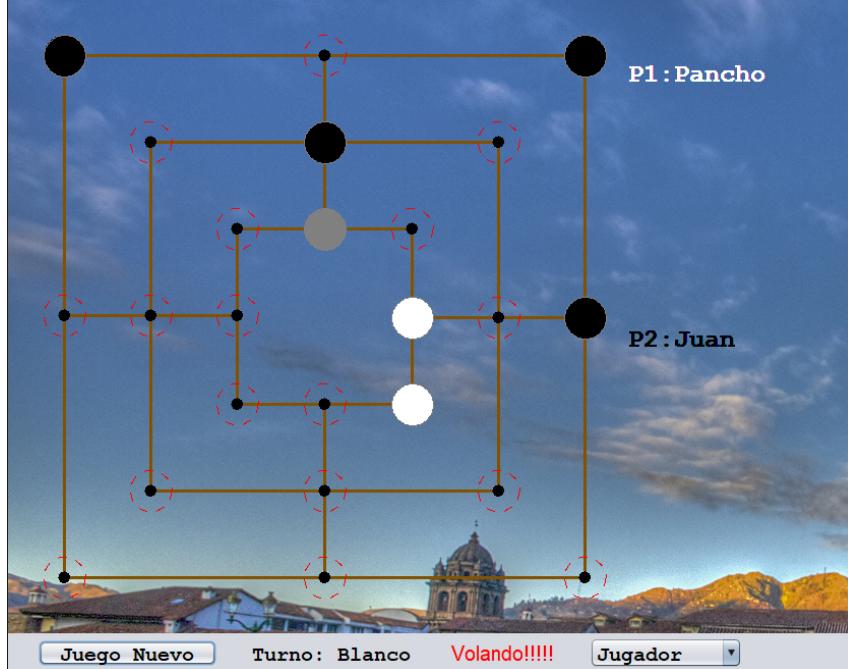
Aquí vemos que, tanto el jugador P1 (Jugador Blanco, que lo llamamos “Pancho”) que tiene la fichas blancas y el jugador P2 (Jugador Negro, que lo llamamos “Juan”), que tiene las fichas negras, terminaron de colocar la totalidad de sus fichas dentro del tablero en las posiciones definidas. Esto da pie a comenzar la siguiente fase del juego, que es el modo de juego Movimiento, en el cual cada jugador puede moverse por el tablero únicamente a sus posiciones adyacentes libres, marcadas por un círculo rojo segmentado. En este caso se muestra el movimiento disponible para la ficha del jugador P1 (Jugador Blanco) marcada en el tablero.



Aquí podemos observar que después de que el jugador P1 (Jugador Blanco, que lo llamamos “Pancho”) que tiene la fichas blancas y el jugador P2 (Jugador Negro, que lo llamamos “Juan”), que tiene las fichas negras, hayan hecho sus respectivos movimientos de ficha válidos dentro del tablero, intercalándose ambos los turnos de juego. Se llegó a un momento en el cual el jugador P1 (Jugador Blanco) logró formar una hilera de 3 fichas consecutivas de su mismo color, formando así lo que se llama un molino. Como consecuencia de esto, entramos al modo de juego Eliminar Ficha, en el cual le damos la opción de eliminar cualquier ficha de su contrincante (Jugador Negro), al jugador que formó el molino (Jugador Blanco).

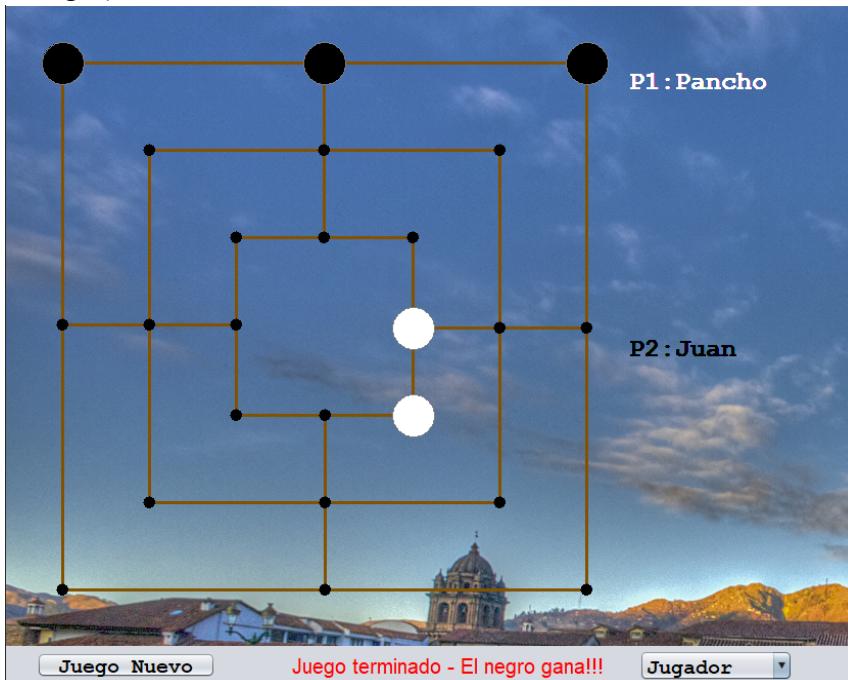


Aquí podemos ver que, tanto el jugador P1 (Jugador Blanco, que lo llamamos "Pancho") como el jugador P2 (Jugador Negro, que lo llamamos "Juan") se eliminaron sus fichas mutuamente, para lo cual cada uno tuvo que formar molinos. Luego de esto se llegó a un punto en el cual el jugador P1 (Jugador Blanco) se quedó únicamente con 3 fichas disponibles, como consecuencia de esto se pasa al modo de Juego Volando. En el cual le otorga la opción de volar al jugador que únicamente tiene 3 fichas sobre el tablero (Jugador Blanco), con la característica de que puede colocar cualquiera de sus fichas en cualquier espacio disponible del tablero.



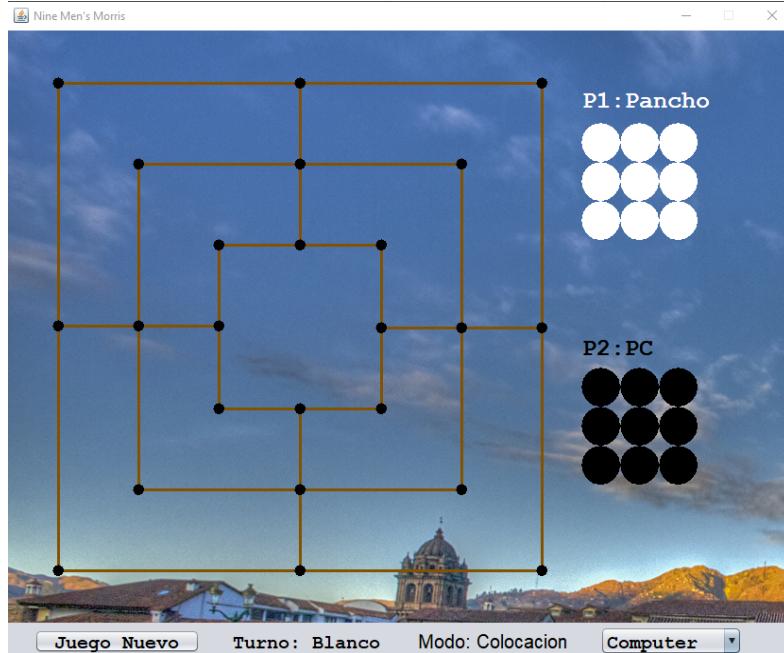
Aquí se observa que el último jugador en formar el molino fue el jugador P2(Jugador Negro), por lo que elimino una ficha de su adversario jugador P1(Jugador Blanco), que únicamente le quedaban 3 fichas y ahora paso a tener solo 2 fichas, dejando así al jugador P1(Jugador Blanco) inhabilitado para seguir jugando, ya que para cualquier movimiento que habría podido realizar el jugador P1 mientras volaba, perdería.

. Esto da como consecuencia que el modo de juego cambie a JuegoTerminado e indicandonos como ganador al jugador P2(Jugador Negro).

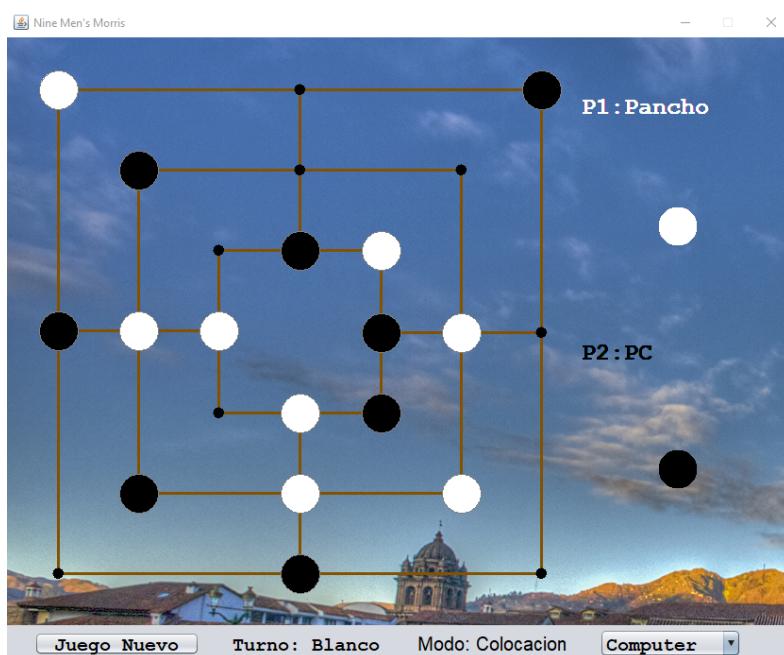


3.2 Modo de juego Player vs PC

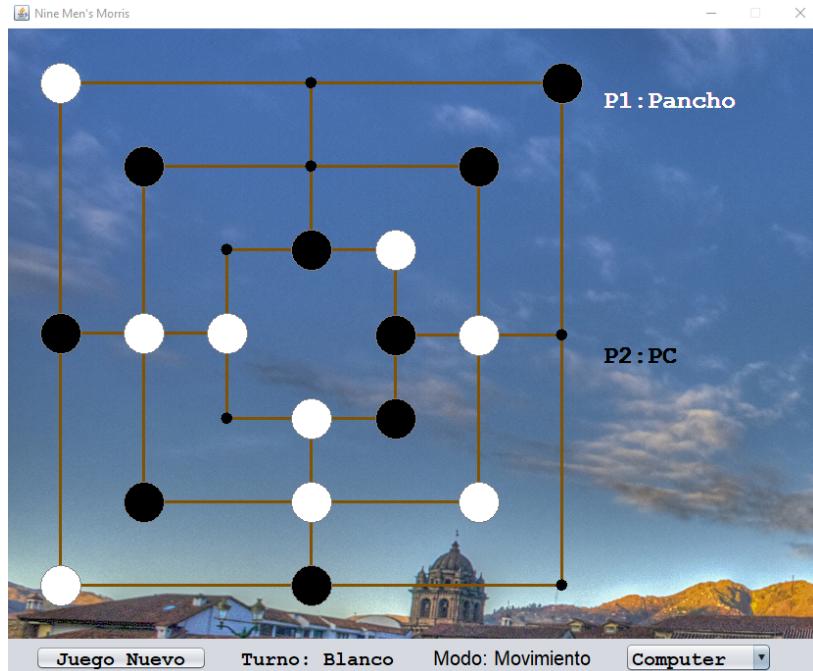
A esta interfaz de Juego se llega siempre y cuando previamente se haya seleccionado el botón Player vs PC en la interfaz de Configuración de Juego. Aquí nos muestra el nombre del jugador P1 que ingresó su nombre en la interfaz Jugador Blanco con su respectivas fichas. Por defecto el turno empieza por el primer jugador P1 (Jugador Blanco, que lo llamamos “Pancho”) que tiene las fichas blancas y luego también tenemos al segundo jugador P2 (que sería por defecto la “PC”), que tiene las fichas negras. Esta interfaz también nos muestra un botón que sirve como reset, que lo llamamos Juego Nuevo, por si el jugador P1 quisiera volver a empezar nuevamente la partida contra la PC. Por último, como ya señalamos, el primer turno es para el jugador P1(Jugador Blanco) y el modo de juego en que nos encontramos es el de Colocación.



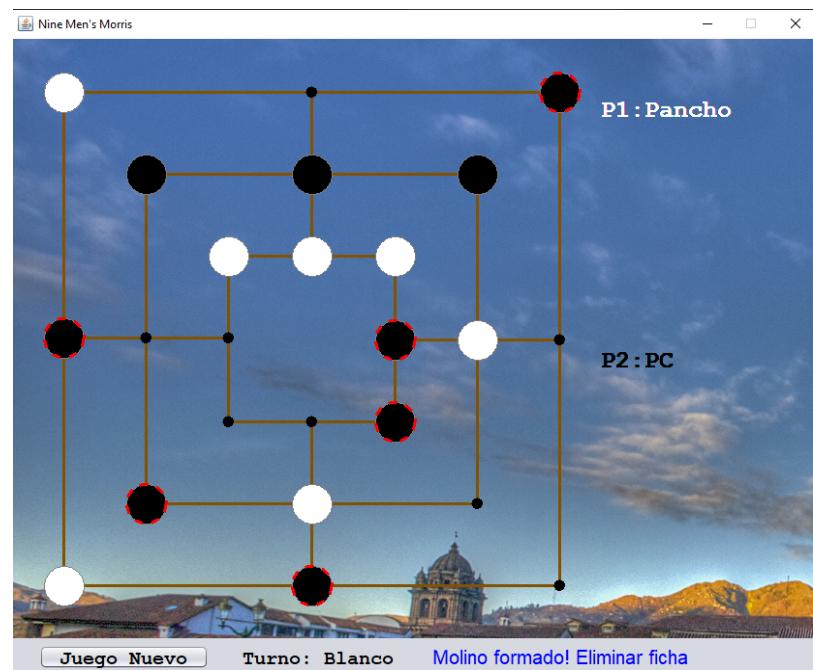
Aquí vemos que, tanto el jugador P1 (Jugador Blanco, que lo llamamos “Pancho”) que tiene las fichas blancas ,así como el jugador P2 (que sería por defecto la “PC”), que tiene las fichas negras, colocaron ya la mayoría de sus fichas, dentro del tablero, en las posiciones definidas. Señalar que previamente todas las fichas estaban fuera del tablero. Todo esto ocurre ya que nos encontramos en el modo de juego de Colocación.



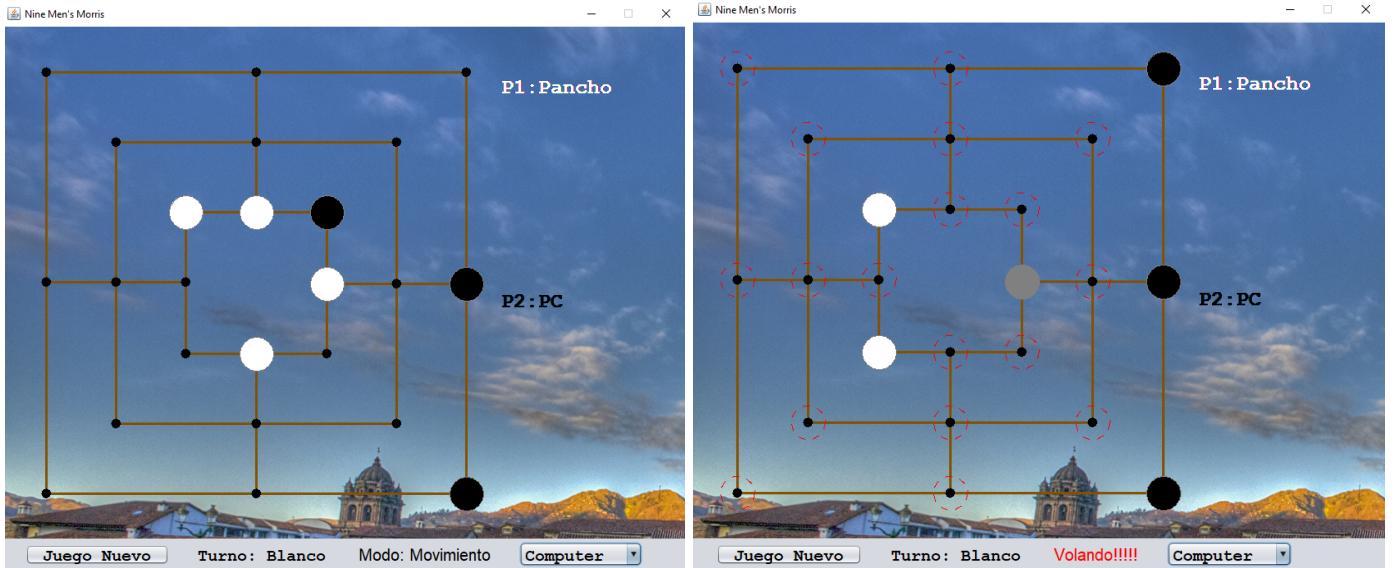
Aquí vemos que, tanto el jugador P1 (Jugador Blanco, que lo llamamos “Pancho”) que tiene la fichas blancas y el jugador P2 (que sería por defecto la “PC”), que tiene las fichas negras, terminaron la colocación de la totalidad de sus fichas dentro del tablero en las posiciones definidas, cabe recalcar que el jugador P2 (PC) coloca sus fichas en el tablero automáticamente. Esto da pie a comenzar la siguiente fase del juego, que es el modo de juego Movimiento, en el cual cada jugador puede moverse por el tablero únicamente a sus posiciones adyacentes libres, marcadas por un círculo rojo segmentado.



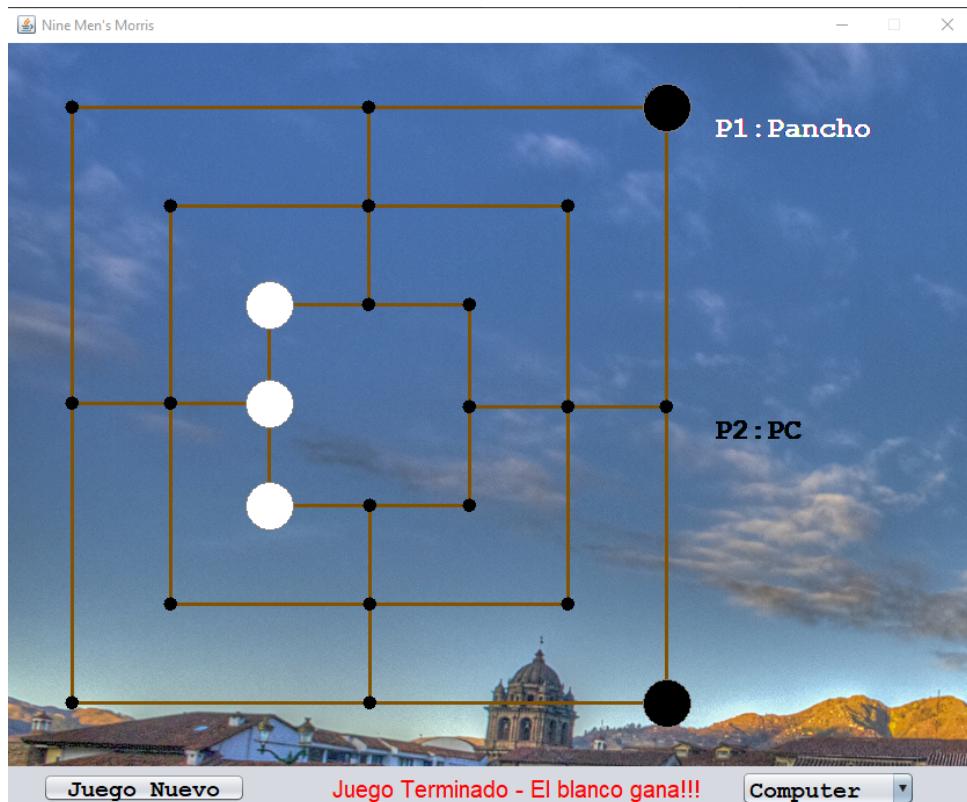
Aquí podemos observar que después de que ambos jugadores, tanto el jugador P1 (Jugador Blanco) y el jugador P2 (“PC”) que tiene las fichas negras, hayan hecho sus respectivos movimientos de ficha válidos dentro del tablero, intercalando ambos los turnos de juego. Se llegó a un momento en el cual el jugador P1 (Jugador Blanco) logró formar una hilera de 3 fichas consecutivas de su mismo color, formando así lo que se llama un molino. Como consecuencia de esto, entramos al modo de juego Eliminar Ficha, en el cual le damos la opción de eliminar cualquier ficha de su contrincante (PC), al jugador que formó el molino (Jugador Blanco).



Aquí podemos ver que, primero el jugador P2 (que sería por defecto la “PC”) está empezando a volar. Una vez que el jugador P2 (la PC) le ha quitado una ficha al jugador P1(Jugador Blanco), quedando únicamente con 3 fichas empezará a volar también. Luego de esto se llegó a un punto en el cual el jugador P1 (Jugador Blanco) tanto como el jugador P2 (que sería por defecto la “PC”) se quedaron únicamente con 3 fichas disponibles, como consecuencia de esto se pasa al modo de Juego Volando. En el cual le otorga la opción de volar a cualquier jugador que tenga 3 fichas sobre el tablero (en este caso el Jugador Blanco y la PC), con la característica de que puede colocar cualquiera de sus fichas en cualquier espacio disponible del tablero.



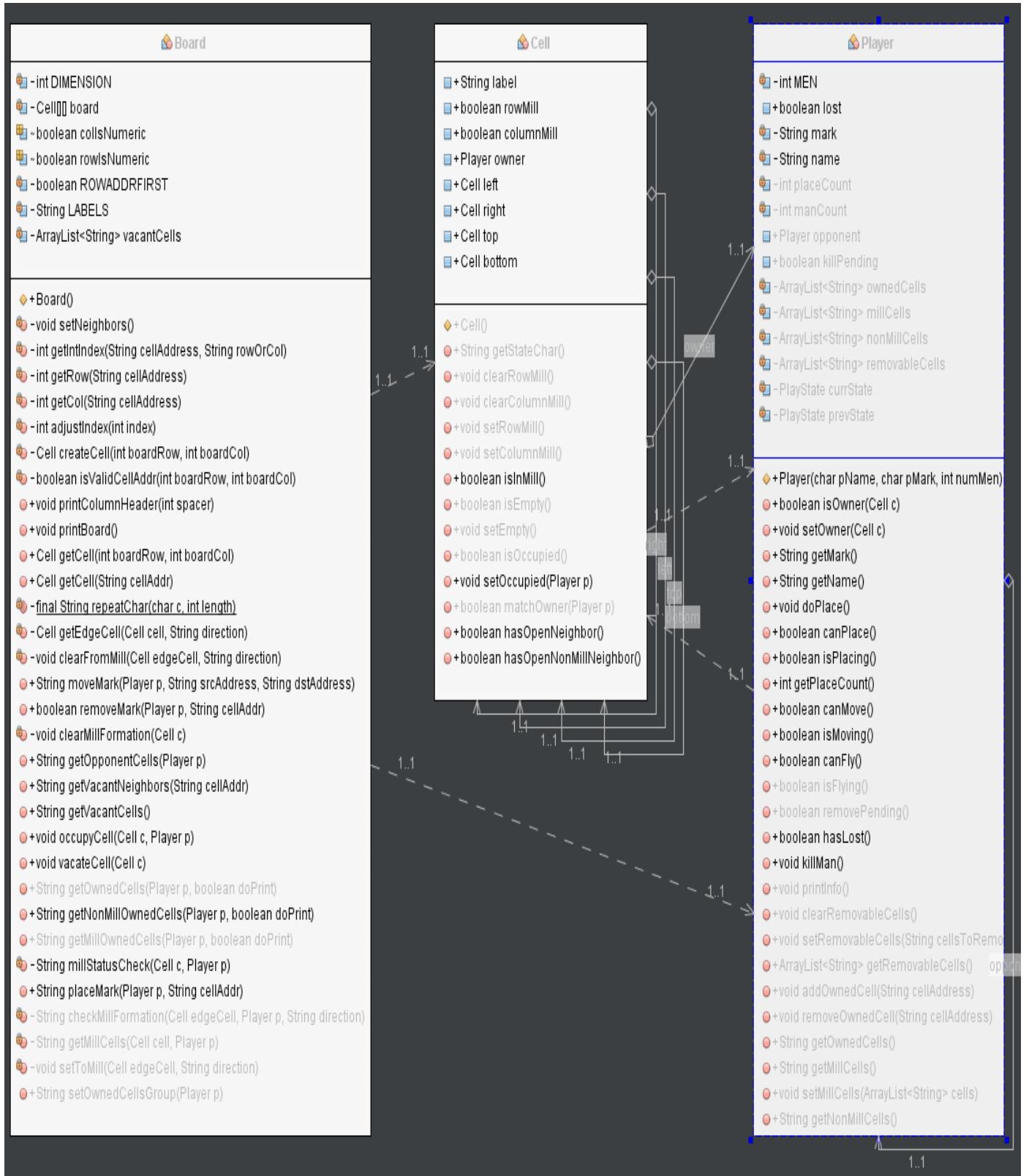
Por último, en esta interfaz nos muestra que el jugador P1(Jugador Blanco, que lo llamamos “Pancho”) tiene 3 fichas y al jugador P2(que sería por defecto la “PC”) le quedan únicamente 2 fichas negras. Así también nos indica que el juego ha terminado dando como ganador al jugador P1(Jugador Blanco) y además tiene la opción de reiniciar la partida(Juego Nuevo) o simplemente cerrar el juego.



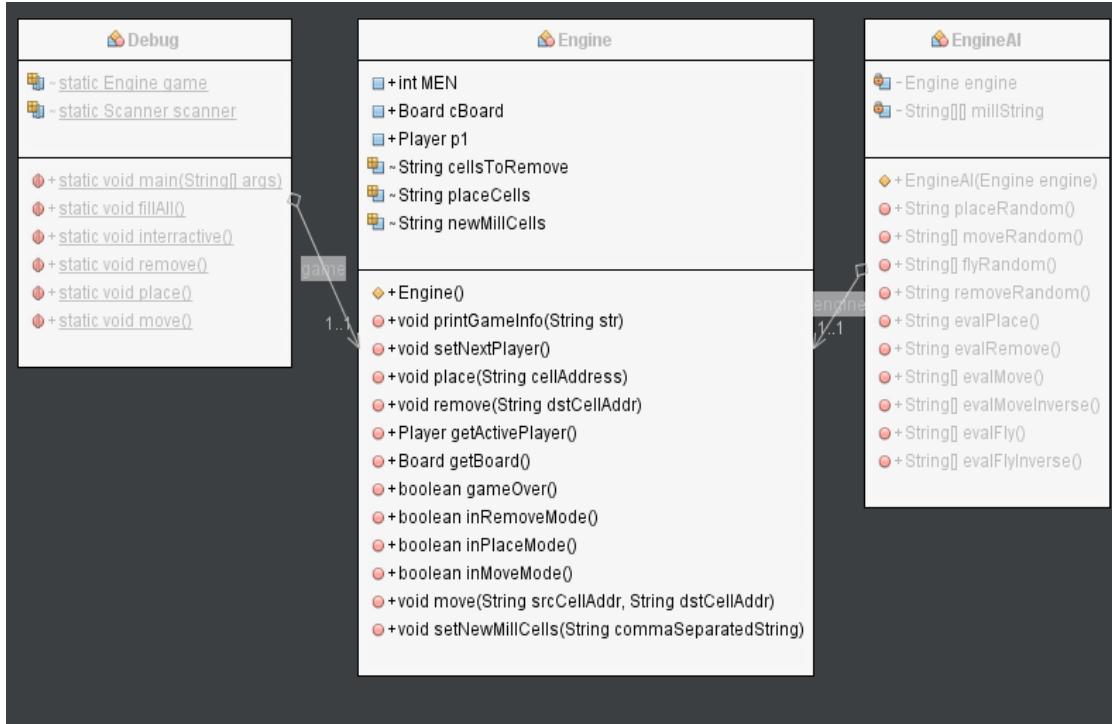
2. Diagrama de clases

Proporciona un diagrama de clases completo de tu código de producción final.

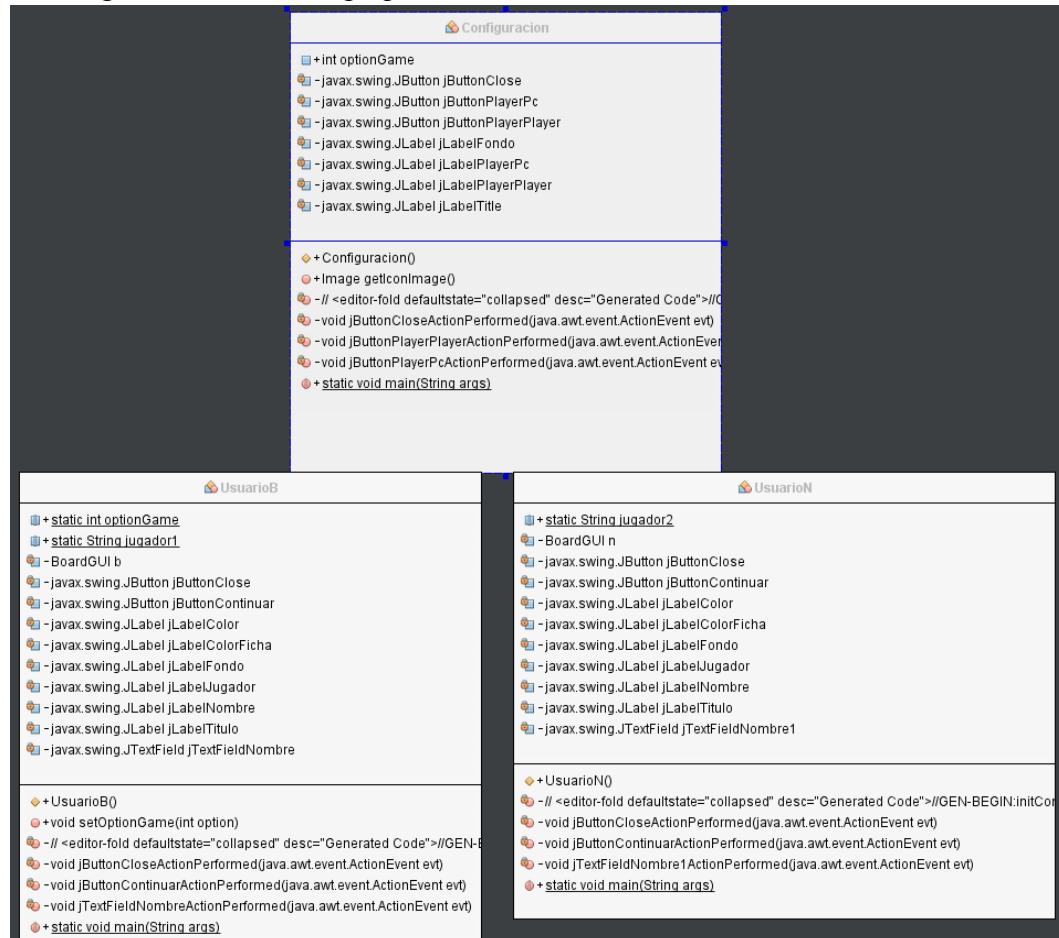
- Diagrama de clase del paquete Board



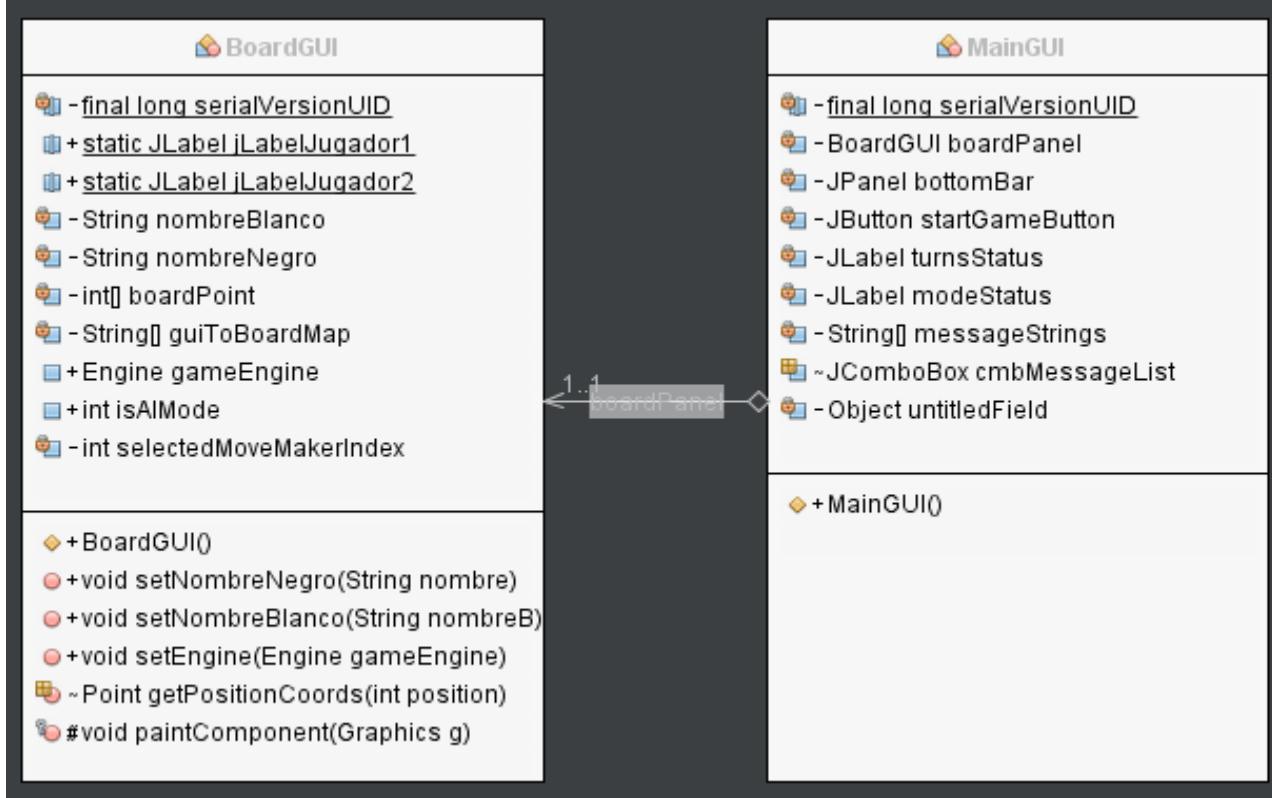
- Diagrama de clase del paquete Engine



- Diagrama de clase del paquete Ventana



- Diagrama de clase del paquete Gui.



Descripción del diagrama de clase de BoardGui:

En el diagrama de clase de la BoardGui mostramos los atributos:

los cuales muestran un signo “negativo” cuando los atributos son privados osea protegen su información para no ser modificados, mientras que muestran un signo “positivo” cuando los atributos son privados osea si permiten el cambio de la información. Estos atributos son herramientas esenciales en el manejo de los métodos que se presentan a continuación en el diagrama..

En el caso de los métodos privados (“-”) nos indican que solo se puede acceder a los elementos de esa misma clase ,muestran ,para ser mas puntual este metodo solo se puede usar en la misma clase , mientras que los demás métodos (“+”) considerados públicos, si se pueden usar desde otra clase. Los métodos que estamos presentando sirven para:

+ void setNombreNegro(String nombre) : Permite almacenar el nombre de un jugador con las fichas negras en una cadena de caracteres llamada nombre ,que luego ingresará al atributo nombre nombreNegro del objeto ,como vemos es un método que realiza un procedimiento ,mas no retorna un valor por el hecho de llevar void.

+ void setNombreBlanco(String nombreB): Permite almacenar el nombre del jugador con las fichas blancas en una cadena de caracteres llamada nombreB ,que luego ingresará al atributo nombreBlanco del objeto ,También es considerado un procedimiento por llevar la palabra void.

+ void setEngine(Engine gameEngine): Permite ingresar gameEngine , en la clase Engine,con lo cual se ingresarán los atributos respectivos del objeto.

- getPositionCoords(int position): Permite recibir la posición de una ficha,para luego con el fin de hacer una revisión si se ha formado tres fichas en una sola hilera.

void paintComponent(Graphics g): En este caso el símbolo # ,que significa protected solo permite el acceso desde subclases y miembros del mismo paquete

Descripción del diagrama de clase de BoardGui y MainGUI:

En este diagrama de clase también se puede rescatar la información del gráfico para los atributos:

En los cuales también se observa el encapsulamiento de la información contenida al ponerle pública o privada.

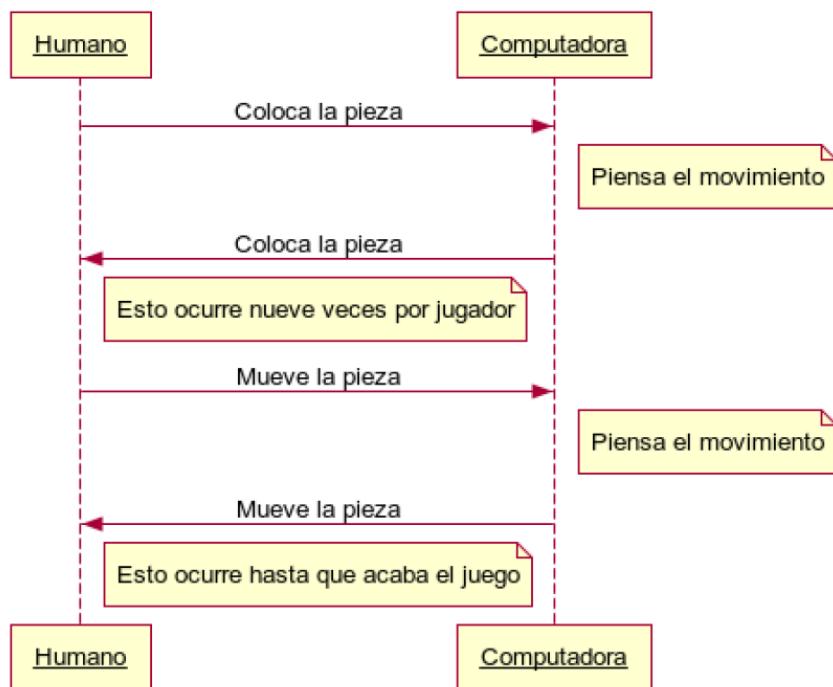
Finalmente la MainGUI esta formado por la composición de la BoardGUI y la barra de configuración donde vemos botones y JLabel que es una caja de texto.

3. Diseño de algoritmos

Enumera los nombres de los miembros del equipo que contribuyeron a esta sección.: Rosales Joaquin, Julio y Aguilar Ybarra, Mario.

Describe el diseño del algoritmo del oponente de la computadora(por ejemplo, usando pseudocódigo). La descripción debe ser comprensible sin hacer referencia al código fuente.

Aqui vemos el caso de uso humano-computador



Tanto el enfrentamiento con una computadora como con otro usuario es el mismo razonamiento que se debe considerar ,es decir :

- 1.- El usuario coloca una pieza en el tablero y espera el juego del computador
- 2.- El usuario coloca otra pieza en el tablero buscando lugares estratégicos y espera su turno
- 3.- El usuario coloca otra pieza en el tablero tratando de tener ventaja para que en la próxima jugada pueda hacer molino y luego espera su turno.
- 4.- El usuario busca formar molino en forma vertical u horizontal si es el caso ,y espera la siguiente jugada
- 5.- El usuario busca hacer molino ,si no puede coloca una pieza en un lugar que le sea favorable formar molino posteriormente. y espera su turno.
- 6.- El usuario hace un recuento de sus fichas y coloca una ficha buscando hacer molino .si hace molino ,le quitara una ficha a la pc ,y espera su turno
- 7.- Y asi hasta que cualquier jugador llegue se quede solamente con dos fichas y se acabara el juego.

4. Extensibilidad

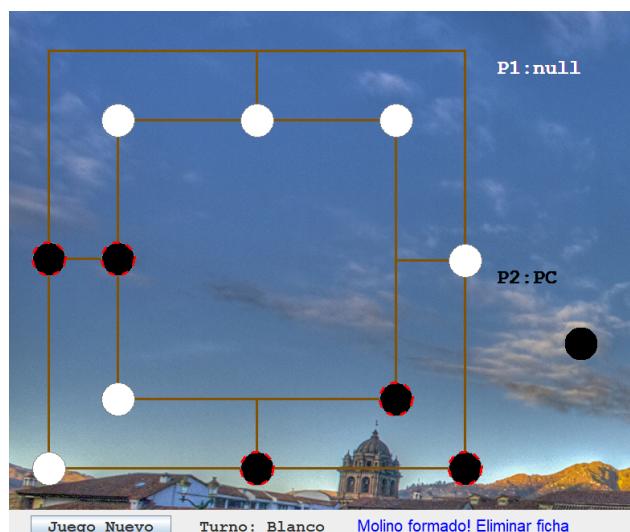
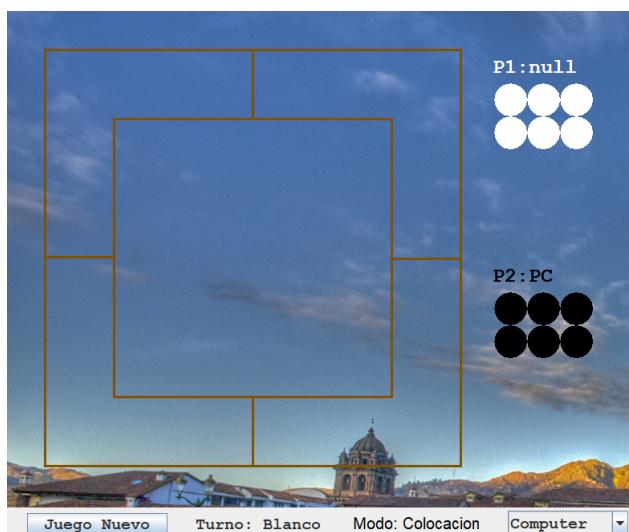
A. Unda Miguel Carlos

B. Pumayalli Quispe Kevin

Discute cómo se puede ampliar el código para las variantes de Nine Men's Morris, incluidos Six Men's Morris y Twelve Men's Morris.

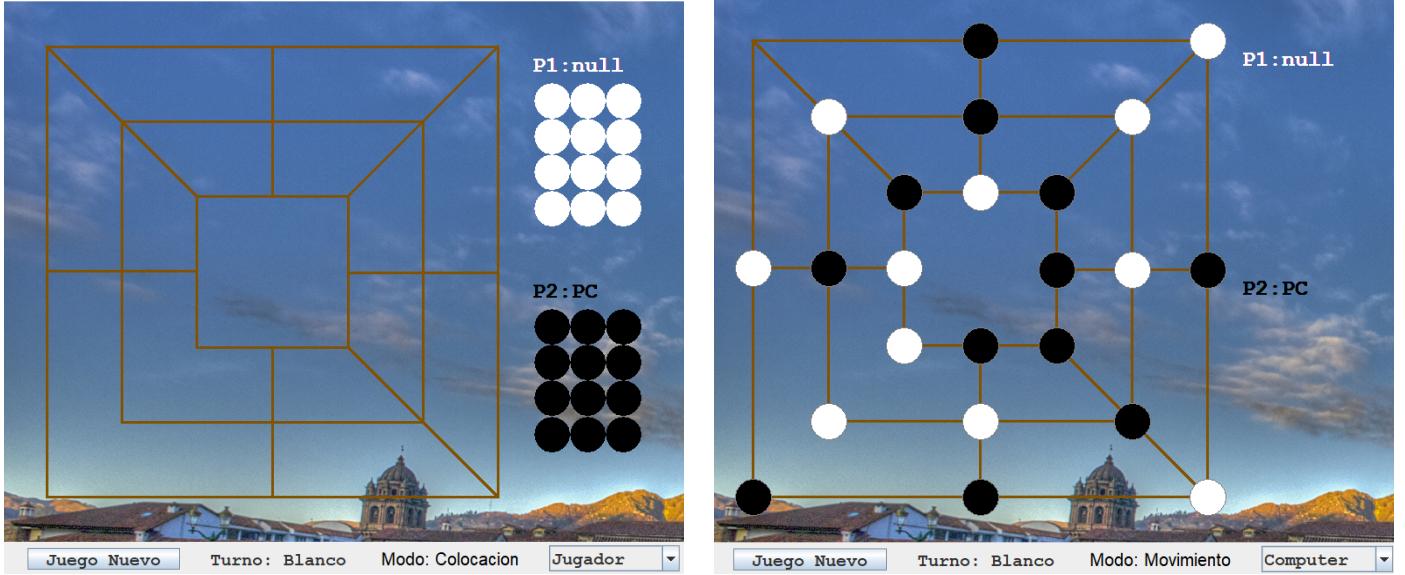
Six Men's Morris:

En primera instancia, tendríamos que cambiar el pintado del tablero para que se aadecue al nuevo número de fichas disponibles para cada jugador (en este caso serían 6 fichas). Otra cosa que tendríamos que cambiar serían las posiciones definidas (en este caso serían 16 posiciones definidas) en donde se podrían colocar las fichas en el tablero. Además también tendríamos que cambiar el número de fichas con que se define que va a empezar cada jugador el juego. También tendríamos que restringir aún más el número de movimientos disponibles para cada ficha del jugador en el tablero. Y por último, lo más importante sería redefinir la lista de las posiciones de las fichas en las cuales es posible formar un molino(osea 3 fichas alineadas consecutivamente) ya que a diferencia del Nine Men's Morris esta lista de posiciones definidas es menor.



Twelve Men's Morris:

En primera instancia, tendríamos que cambiar el pintado del tablero, agregando nuevas diagonales en el tablero, para que se aadecue al nuevo número de fichas disponibles para cada jugador (en este caso serían 12 fichas). Otra cosa que tendríamos que modificar serían los movimientos de forma diagonal en el tablero. Además también tendríamos que cambiar el número de fichas con que se define que va a empezar cada jugador el juego. También tendríamos que asignar aún más el número de movimientos disponibles para cada ficha del jugador en el tablero. Y por último, lo más importante sería redefinir la lista de las posiciones de las fichas en las cuales es posible formar un molino(osea 3 fichas alineadas tanto consecutivamente como de forma diagonal) ya que a diferencia del Nine Men's Morris esta lista de posiciones definidas es mayor.



¿Qué clases y métodos deben cambiarse y cómo?

Primero cambiamos la clase BoardGUI, tendríamos que hacer algunas modificaciones menores al método paintComponent con el fin de reasignar la cantidad de fichas necesarias para cada juego, y únicamente tendremos que redefinir algunos atributos privados de la clase con el fin de que se acople a la variante respectiva del Nine Men's Morris.

Luego en clase Engine, tendríamos que redefinir un atributo público de esta clase y también el método move para así redefinir los movimientos posibles de las fichas en el tablero, dependiendo de la variante del Nine Men's Morris que se trate.

En la clase EngineAI, únicamente tendríamos que redefinir un atributo privado de esta clase, encargado de definir las posiciones relativas de las fichas en las cuales se forma un molino dependiendo del tablero que tenemos ya sea para el Nine Men's Morris o para alguna de sus variantes.

Por último en la clase Board, redefinimos un variable privada que se toma como referencia para el tamaño del tablero a definir, también cambiaremos los puntos definidos dentro del tablero y también las coordenadas de las celdas adyacentes de cada celda definida en el tablero.

¿Cómo se aplicó el principio abierto-cerrado?

Las clases Cell y Player están cerradas para la modificación ya que prácticamente son independientes de las otras clases, ya que solo son llamadas por otras para utilizarlas, los únicos cambios posibles que podría hacer en estas clases sería redefinir un atributo. Mientras que por ejemplo en la clase Board tendríamos que redefinir o agregar algunos atributos privados de esta clase, y tendríamos que añadir nuevas funciones dependiendo de las variantes del juego que quisiéramos implementar, sin modificar las funciones previamente definidas, en conclusión esta clase está abierta para su extensión pero también está cerrada para su modificación.

También está la clase Engine que únicamente estaría cerrada para su modificación ya que también es prácticamente independiente de las otras clases. Mientras que por otro lado está la clase EngineAI que está

abierta para su extensión ya que tendríamos que agregar atributos privados en los cuales se defina las series de posiciones en las cuales se forma un molino para las distintas versiones del juego, pero claro sin modificar las funciones ya hechas.

Por último encontramos a la MainGUI que está abierta para sus extensiones ya que podríamos agregarle más funcionalidades a la interfaz, pero son modificar el funcionamiento de las otras. Por otro lado está la clase BoardGUI que tendríamos que agregar y definir nuevos puntos en el mapa dependiendo del tablero que queramos jugar así como también para poder pintar el tablero, también tendríamos que agregar nuevas funciones para imprimir los componentes de este sin dañar ni modificar los que anteriormente teníamos, es decir cerrada para la modificación pero abierta para la extensión.

¿Cómo se aplica el Principio de Segregación de Interfaces?

-En este caso Para las interfaces de Configuración-UsuarioB-UsuarioN, al momento de seleccionar el botón Player vs Player utiliza la interfaz UsuarioB y UsuarioN y posteriormente se usa la interfaz del tablero, en cambio cuando seleccionar el botón Player vs PC solo manda a llamar al UsuarioB y no al UsuarioN ya que en este caso sería por defecto la PC por ende se aplica el Principio de Segregación de Interfaces donde no usa ni depende de los métodos que usar el Botón (Continuar) ya que en este caso llama directamente al Tablero de Juego.

-También para el momento de cada tipo de Juego que se requiera se usarán los métodos específicos y no se usarán métodos del juego Nine en el tipo de juego de Six o Twelve. Por ejemplos los arreglos de asignación de posiciones,molinos definidos,cantidad de fichas,y dependiendo del tipo de jugabilidad mandará a llamar los métodos de la Clase Player donde se ve que aplica el Principio de Segregación de Interfaces ya que se podría asignar 1 Player o 2 Players.

VI. Conclusiones del ejercicio de revisión del código

Nombre de los participantes: Rosales Joaquin, Juilo; Aguilar Ybarra, Mario, Pumayalli Kevin.

Clase que fue revisada:

Checklist	Items Checklist	Conclusiones
Estándares de codificación	Convenciones de nombres	Se usaron una convención de nombres estratégicos que permitieron el fácil entendimiento del código.
	Comentarios significativos y válidos.	Se comentaron los métodos o aquellas líneas de código que puede ser un poco difícil de entender.
	Estilo consistente de bloques de código	En todos los bloques de código se usa el mismo estilo
	Indentación consistente	Todo el código tiene una correcta indentación tanto a nivel de clases como a nivel de condicionales
Principio de diseño	Buena abstracción de clase e interfaz	La abstracción es una buena alternativa para poder implementar diversas clases con el apoyo de la interfaz, para evitar complicaciones con la herencia múltiple.
	Visibilidad adecuada de cada variable, método y clase.	Las variables escogidas con el adecuado nombre sugerente a su función dentro de una clase son una herramienta que puede ayudar a entender el código de manera que se debería tener cuidado para su uso local y no global.
	Alguna violación del principio de separación comando-consulta ¹	
	Diseño por contrato (pre/postcondiciones)	

¹ Revisa: [Writing professional code with command-query separation](#)

	¿Se viola el Principio Abierto-Cerrado?	se le da la capacidad para modificar el código mientras que el cerrado no se puede modificar	
	¿Se viola el Principio de Responsabilidad Única ² ?		
Smells código	Números mágicos	No hemos considerado números mágicos porque la existencia de estos genera probabilidad de errores.	
	Variable global /clase innecesaria	Trabajar con variables globales hace que el código este muy acoplado ,le da un carácter monolítico se hace más complicado cambiar le algunas partes con facilidad	
	Código duplicado	Aplicando la Refactorización vemos que vamos a tener un clean code(Código limpio) en donde cada vez que tenga que hacer un cambio en un código duplicado, debe recordar hacer el mismo cambio en cada instancia. Esto aumentará la carga cognitiva y relentiza el progreso.	
	Métodos largos	Los métodos largos fueron reemplazados por métodos cortos de fácil comprensión en cuanto a su función.	
	Larga lista de parámetros	Cuando hacemos uso de otras clases en este código mayormente se le pasa hasta un máximo de 3 parámetros.	
	Expresión demasiado compleja	Son muy pocas las expresiones complejas las que utilizamos, más que su complejidad, es la longitud de la expresión.	
	Switch o if-then-else que necesita ser reemplazado con polimorfismo	En el caso cuando queramos extender a otras variantes del juego, es necesario usar polimorfismo ya que se utilizan los mismo métodos definidos.	
Errores	Fragmento de código con errores	:Cuál es el error?	:Por qué es un error?
	<pre>private String millStatusCheck(Cell c, Player p) { String status = ""; // Check if most recent action formed a Mill status = getMillCells(c, p); // Non-empty delimited string == cells that formed mill; if (status.length() == 0){ status = "SUCCESS"; } return status; }</pre>	Non-empty delimited string == cells that formed mill;	No está delimitado la longitud del estado
	<pre>millCells = checkMillFormation(edgeCell, p, "right"); if (millCells.length() > 0) { //System.out.println("Row mill check result = \\"" + millCells + "\\""); return millCells; } </pre>	//System.out.println("Row mill check result = \\"" + millCells + "\\"")	Al inicio si tiene sentido el código ya que nos muestra que se va a retornar un molino en fila, pero sabiendo que esto nos puede traer problemas con la memoria caché lo comentamos.
	<pre>millCells = checkMillFormation(edgeCell, p, "bottom"); if (millCells.length() > 0) {</pre>	System.out.println("Column mill check result = \\"" + millCells + "\\"")	En este caso similar al anterior queremos ver que contiene antes de retornar como resultado del molino de columnas con el método checkMillFormation en la variable millcells. Pero lo comentamos para que

² Revisa: [Violation solution for single responsibility principle](#)

	<pre> // System.out.println("Column mill check result = """ + millCells + """); return millCells; } return ""; </pre>		no sobrecargue la memoria caché.
--	--	--	----------------------------------

VII. Acta de reuniones

Reporta las actas de todas las reuniones, incluidas, entre otras: reunión de planificación de proyecto/sprint, reunión de trabajo, backlog grooming , reunión retrospectiva y sesiones de programación en pares.

Fecha	Tiempo	Lugar	Nombre Participantes	Propósito de la reunión	Elementos de acciones específicos
19/7	1h 25 min	meet	kevin Pumayali Carlos Unda Julio Rosales Mario Aguilar	Revisar el código presentado anteriormente,y sugerir mejoras	Implementación del logeo considerando el juego con la máquina
22/7	1 Hora	meet	kevin Pumayali Carlos Unda Julio Rosales Mario Aguilar	Compartir el trabajo en dos grupos uno para la documentación y el otro para el avance de código	Diseño de las nuevas historias de usuario y criterios de aceptación ,para la ampliación de nuestro código.
24/7	3 Horas	meet	kevin Pumayali Carlos Unda Julio Rosales Mario Aguilar	Seguir con la implementación de la parte adicional de código y las nuevas historias de usuarios	Detección de los fragmentos de código small. y su refactorización,modificación del interface al momento del loguea
25/7	2.5 Horas	meet	kevin Pumayali Carlos Unda Julio Rosales Mario Aguila	Creación de más historias de usuarios y criterios de aceptación	Implementación de los historias de usuario y su respectiva líneas código .en el software principal
29/7	3 Horas	meet	kevin Pumayali Carlos Unda Julio Rosales Mario Aguila	Terminando de completar el código y avanzar la documentación	Diseño del diagrama de clases ,implementación complementaria de la interface con la participación de la computadora como un jugador mas
30/7	3.5 Horas	zoom	kevin Pumayali Carlos Unda Julio Rosales Mario Aguila	Complementando el código con los comentarios respectivos	Repartimos las zonas del código para implementar los comentarios
31/7	5 horas	zoom	kevin Pumayali Carlos Unda Julio Rosales Mario Aguila	Llenar la documentación faltante y colocar las capturas de código ,grabación de video y elaboración del trello	Llenar la documentación faltante y colocar las capturas de código ,grabación de video y elaboración del trello

VIII. Calificación de amigos

Calificación receptor

	Aguilar Ybarra	Rosales Julio	Pumayalli Kevin	Unda Carlos
Aguilar Ybarra Mario	X	17	17	17
Rosales Julio	16	X	16	16
Pumayallli Kevin	16	16	X	16
Unda Carlos	17	17	17	X
<i>Promedio</i>	16.3	16.6	16.6	16.3