Informe Sprint #3
Informe Grupal

Instrucciones

Lee atentamente las instrucciones. Todos los miembros del equipo deben discutir las instrucciones juntos para asegurarse de que todos estén en la misma página. Esta actividad es la continuación a una trabajo entregado anteriormente, por lo que se debe mantener las ideas y código realizado.

Objetivos

1. Actualiza y completa las historias de usuario y los criterios de aceptación del software a presentar que permite que un jugador humano juegue contra un oponente humano o de máquina.
2. Completa la implementación de todas las historias de usuario, incluidas las mejoras en el sprint anterior. Ten en cuenta que el oponente de la computadora debe hacer un intento razonable de vencer al jugador humano. Por ejemplo, puede derrotar al peor jugador humano. No llame**s AI** al oponente de la computadora.
3. Realiza un ejercicio de revisión del código en equipo (durante al menos una hora) e informa los hallazgos. Todos deberían haber leído las siguientes instrucciones antes del ejercicio.

En este ejercicio, cada equipo aplicará prácticas de revisión de código a la clase más importante del juego Mill (y otras clases si el tiempo lo permite) en el proyecto. Un miembro debe liderar la **revisión del código** (utiliza el siguiente enlace: https://web.mit.edu/6.031/www/sp21/classes/04-code-review/) y otro miembro debe tomar notas y documentar los hallazgos. La revisión debe ser constructiva. No critiques a los desarrolladores que escribieron el código bajo revisión. No es necesario probar el programa ni realizar cambios durante el ejercicio de revisión. Los resultados se pueden utilizar para mejorar tu proyecto después de la revisión.

Además de buscar errores, la revisión debe verificar: (1) si todo el proyecto ha seguido el estándar de codificación de manera consistente, (2) si el proyecto ha seguido los principios de diseño presentados en clase y (3) si hay smells de código que indican la necesidad de refactorización (revisa: https://medium.com/oceanize-geeks/code-smells-and-refactoring-c2c0e0642582) y http://www.cs.kent.edu/~jmaletic/cs63902/Papers/DiPenta09.pdf. Las siguientes listas de verificación proporcionan pautas básicas. Puedes agregar nuevos elementos a cada una de las listas de verificación.

Checklist #1: Estándares de codificación

- ¿Hay alguna violación de las convenciones de nomenclatura?
  - Paquetes, clases, métodos, variables, constantes
  - Código de producción vs código de prueba

- ¿Se sigue la convención de ordenación (https://stackoverflow.com/questions/4668218/are-there-any-java-method-ordering-conventions) de argumentos de método en cada método?
- ¿Son todos los comentarios significativos y válidos?
  - ¿Están documentadas las precondiciones y postcondiciones de cada método?

- ¿Se usa el mismo estilo para todas las llaves de los bloques de código?
- ¿Indentación consistente?
- …

Checklist #2: Principio de diseño

- ¿Cada clase tiene una buena abstracción y una buena interfaz de clase?
- ¿Es adecuada la visibilidad de cada variable, método y clase (privada, protegida, pública, predeterminada)?
- Diseño por contrato: para cada método público, ¿se sigue el diseño por contrato? Si es así, ¿la condición previa especificada es razonable y está disponible?
- ¿Se viola el Principio Abierto-Cerrado?
- ¿Se utiliza SOLID?
- …

Checklist #3: Smells de código

- ¿Hay algún número mágico o constante sin nombre?
- ¿Hay alguna variable global o de clase innecesaria?
- ¿Hay código duplicado?
- ¿Hay métodos largos (long methods)[1]?
- ¿Algún método tiene una larga lista de parámetros?
- ¿Hay alguna expresión demasiado compleja?
- ¿Hay algún switch o declaración if-then-else que deba reemplazarse con polimorfismo?
- ¿Hay alguna variable o nombre de método cuya intención no esté clara?
- ¿Existen métodos similares en múltiples clases?
- …

4. Revisar el diseño de software del código de producción final.

- Resume el diseño de la interfaz de usuario mediante una combinación de capturas de pantalla y descripciones textuales.
- Presenta el diagrama de clases completo del código de producción final.
- Describe los algoritmos para que el oponente de la computadora coloque una pieza, quite una pieza, mueva y vuele una pieza. Las descripciones deben ser comprensibles sin hacer referencia al código fuente. Por ejemplo, puedes usar pseudocódigo.
- Discute cómo se puede ampliar tu código para las variantes de Nine Men's Morris, incluidos Six Men' s Morris y Twelve Men 's Morris. ¿Qué clases y métodos deben cambiarse y cómo? ¿Cómo se aplicó el principio abierto-cerrado (es decir, qué funciones o clases están abiertas para la extensión, pero cerradas para la modificación)?.

Entregables y Política de Calificación

Comprima el informe del proyecto, el video de demostración y todo el código fuente en un archivo .zip antes de enviarlo. Por favor, no cargues archivos rar y no envíes enlaces. **Puntuación máxima 20 puntos**

1. Informe del proyecto

[1] Se conoce como Long Method, Long Class, y God Class (https://www.javacodegeeks.com/2019/09/identifying-code-smells-in-java.html)

El proyecto debe incluir todas las siguientes secciones.

I. Historias de usuario completas y criterios de aceptación actualizados (3 puntos).

II. Tareas de implementación (5 puntos)
Describe el código de producción, el código de **prueba automatizado** o los **casos de prueba manual** para todas las historias de usuario. Para cada criterio de aceptación de cada historia de usuario, debe implementar al menos una prueba (ya sea código de prueba o caso de prueba manual).

III. El diseño del código de producción final (5 puntos)
Diseño de interfaz de usuario (1 punto)
Arquitectura de software (1 punto)
Algoritmos (1 punto)
Extensibilidad (2 puntos)

IV. Hallazgos del ejercicio de revisión del código (3 puntos)

V. Resumen del código fuente (2 puntos)
Proporciona un archivo zip de todo el código fuente y resume la contribución de cada miembro. No recibirás ningún punto si no se envía el código fuente. Asegúrate de que el informe de tu proyecto sea coherente con el código fuente.

VI. Anota las reuniones, incluidas, entre otras: reunión de planificación de proyecto/sprint, reunión de stand-ups, backlog grooming, reunión retrospectiva y sesión de programación (o desarrollo) en pares (2 puntos)

VII. Una tabla de calificaciones de amigos. Los miembros individuales pueden enviar sus calificaciones de compañeros por correo electrónico al profesor del curso.

Cada equipo solo necesita presentar un informe. **Para que un miembro individual reciba puntaje por esta parte del proyecto, el informe del proyecto del equipo debe incluir evidencia explícita de tu contribución.**

2. Demostración

Envía un video de 15-20 minutos en formato .mp4, que debe demostrar claramente que:

a) tu proyecto ha completado la implementación de todas las características requeridas para el oponente de la computadora.
   (1) Un juego completo ganado por el jugador humano
   (2) Un juego completo ganado por el oponente de la computadora sin volar
   (3) Un juego completo ganado por el oponente de la computadora con vuelo
b) para cada criterio de aceptación de cada historia de usuario para el oponente de la computadora, tu proyecto implementó un método de prueba automatizado o realizó una prueba de aceptación manualmente.
c) tu proyecto tiene algunas características o mejoras únicas (opcional).

La calificación de la demostración se basa en la finalización de las funciones requeridas y la presentación general utilizando la siguiente rúbrica de evaluación:

	Pobre	Justo	Bueno	Muy bueno	Excelente
¿Estaba la demostración lógicamente organizada?					
¿Se formularon los puntos de forma clara y concisa?					
¿Se respondieron satisfactoriamente las preguntas del calificador o del profesor?					

Informe Sprint #3

Plantilla del informe

Nombre del equipo:

Información dada por el equipo del Proyecto		A ser usado por el profesor	
NombreEstudiante	**Contribuciones específicas para este Sprint**	**Puntaje Equipo**	**Puntaje Individual**
Lezama Verástegui Dagmar Nicole	Historias de Usuario. Criterios de aceptación. Tareas de Implementación. Trello. Edición del video. Código de pruebas. Documentación y notas de revisión del código		
Quispe Olaechea Pablo Alejandro	Historias de Usuario. Criterios de aceptación. Tareas de Implementación. Trello. Código de pruebas. Líder de revisión de código.		
Tello León José Joaquín	Historias de Usuario. Criterios de aceptación. Tareas de Implementación. Trello. Código de pruebas. Documentación y notas de revisión de código.		

Un estudiante sin ninguna participación no recibirá puntaje.

I. Actualización de las historias de usuarios

ID	Nombre Historia de Usuario	Descripción Historia Usuario	Prioridad	Esfuerzo estimado(horas)	Esfuerzo real (si se completó)	Status (completado, por Hacer, en Progreso)	Nombre desarrollador
1	Configuración inicial para empezar una partida	Como cliente debo poder escoger al tipo de contrincante de juego: contra otro humano o contra la máquina.	Alta	2h	2h 15 min	Completado	Dagmar Lezama
2	Desplegar ficha azul como jugador humano	Como jugador humano debo poder colocar mi ficha azul en cualquier casilla disponible en el tablero.	Alta	2h	2h	Completado	Pablo Quispe
3	Desplegar ficha roja como jugador máquina	Como jugador máquina debo poder colocar mi ficha roja en cualquier casilla disponible en el tablero.	Alta	2h	2h	Completado	Pablo Quispe
5	Retirar pieza siendo jugador humano	Como jugador humano debo poder retirar una pieza roja al formar una línea de 3 fichas azules	Alta	3h	3h	Completado	Pablo Quispe
6	Retirar pieza siendo máquina	Como máquina debo poder retirar una pieza azul al formar una línea de 3 fichas rojas	Alta	3h	3h	Completado	Pablo Quispe

7	Pasar a fase II como jugador máquina	Como jugador máquina necesito saber si el juego pasa a la fase II luego de colocar mi pieza.	Alta	2h	2h	Completado	José Tello
8	Pasar a fase II como jugador humano	Como jugador humano necesito saber si el juego pasa a la fase II luego de colocar mi pieza.	Alta	2h	2h	Completado	José Tello
9	Desplazar pieza siendo jugador humano	Como jugador humano, necesito mover una pieza azul colocada en el tablero hacia una posición adyacente disponible	Alta	3h	3h	Completado	José Tello
10	Desplazar pieza siendo jugador máquina	Como jugador máquina, necesito mover una pieza roja colocada en el tablero hacia una posición adyacente disponible	Alta	3h	3h	Completado	José Tello
11	Pasar a fase III como jugador máquina	Como jugador máquina necesito saber si el juego pasa a la fase III luego de colocar mi pieza.	Alta	1h	1h	Completado	José Tello
12	Pasar a fase III como jugador humano	Como jugador humano necesito saber si el juego pasa a la fase III luego de colocar mi pieza.	Alta	1h	1h	Completado	José Tello
13	“Vuelo” para jugador máquina	Como jugador máquina debo poder mover una pieza roja hacia cualquier lugar disponible cuando tenga solo 3 piezas restantes.	Alta	1h	1h 30 min	Completado	Dagmar Lezama Pablo Quispe
14	“Vuelo” para jugador humano	Como jugador humano debo poder mover una pieza azul hacia cualquier lugar disponible cuando tenga solo tenga 3 piezas restantes.	Alta	1h	1h 30 min	Completado	Dagmar Lezama Pablo Quispe
15	Fin de juego	Como jugador, necesito saber si el juego termina después de cada movimiento.	Alta	1h	1h	Completado	Dagmar Lezama

II. Actualización de los criterios de aceptación (AC)

ID Historia de Usuario y Nombre	AC ID	Descripción de los criterios de aceptación	Status (completado, por Hacer, en Progreso)	Nombre desarrollador
1. Configuración inicial para empezar una partida	1.1	AC 1.1 Inicio de Menú Dado el inicio de la ejecución del juego Cuando se abre la ventana del mismo Entonces se muestra el menú de Bienvenida al usuario Y este tiene que escoger un modo de juego para empezar su partida	Completado	Dagmar Lezama
2. Desplegar ficha azul como jugador humano	2.1	AC 2.1 Posicionamiento de despliegue azul válido Dada la fase de despliegue de un juego en curso con el turno del jugador humano Cuando el jugador selecciona una posición disponible Entonces su ficha azul es colocada en dicha posición en el tablero Y el turno cambia a jugador rojo.	Completado	Pablo Quispe
	2.2	AC 2.2 Posicionamiento de despliegue azul inválido en una celda ocupada Dada la fase de despliegue de un juego en curso con el turno del jugador humano Cuando el jugador selecciona una posición ocupada dentro del tablero Entonces su ficha no es colocada Y el juego espera una posición válida manteniendo su turno en curso.	Completado	Dagmar Lezama
	2.3	AC 2.3 Posicionamiento de despliegue azul inválido fuera del tablero Dada la fase de despliegue de un juego en curso con el turno del jugador humano Cuando el jugador selecciona cualquier parte no correspondiente a las posiciones designadas dentro tablero Entonces su ficha no es colocada Y el juego espera una posición válida manteniendo su turno en curso.	Completado	Pablo Quispe
3. Desplegar ficha roja como jugador máquina	3.1	AC 3.1 Posicionamiento de despliegue de máquina válido Dada la fase de despliegue de un juego en curso con el turno del jugador máquina Cuando la máquina utiliza su lógica para desplegar una ficha en una posición disponible Entonces su ficha roja es colocada en dicha posición en el tablero Y el turno cambia a jugador humano.	Completado	Pablo Quispe
4. Retirar pieza siendo jugador humano	4.1	AC 4.1 Retiro común al detectar un tres en raya azul recién armado Dado un juego en curso con el turno del jugador azul en cualquier fase Cuando este haya formado una secuencia consecutiva de 3 fichas azules Entonces se registra temporalmente la nueva secuencia y el jugador debe capturar cualquier pieza roja que se encuentre en el tablero Y al hacerlo, se libera la posición de la pieza roja seleccionada y el turno pasa a ser del jugador rojo	Completado	Pablo Quispe
	4.2	AC 4.2 Retiro no procedente al detectar un tres en raya azul ya formado Dado un juego en curso con el turno del jugador azul en cualquier fase Cuando este haya formado una secuencia consecutiva de 3 fichas azules que ya ha sido registrada como activa Entonces no le es posible capturar ninguna pieza roja que se encuentre en el tablero y debe continuar con su fase respectiva Y tras concretar la acción adecuadamente, el turno cambia al jugador rojo	Completado	Pablo Quispe
	4.3	AC 4.3 Retiro de una pieza parte de un mill enemigo Dado un juego en curso con el turno del jugador azul Cuando este haya formado una secuencia consecutiva de 3 fichas Y se quiera eliminar una pieza perteneciente a un mill enemigo Entonces la pieza no se elimina	Pendiente	Pablo Quispe

		Y se debe seleccionar otra pieza a eliminar		
5. Retirar pieza siendo máquina	5.1	AC 5.1 Retiro común al detectar un tres en raya rojo recién armado Dado un juego en curso con el turno del jugador máquina en cualquier fase de un juego en curso Cuando este haya formado una secuencia consecutiva de 3 fichas rojas Entonces se registra temporalmente la nueva secuencia y el jugador debe capturar cualquier pieza azul que se encuentre en el tablero Y al hacerlo, se libera la posición de la pieza azul seleccionada y el turno pasa a ser del jugador azul.	Completado	Pablo Quispe
6. Desplazar pieza siendo jugador humano	6.1	AC 6.1 Identificación de casillas disponibles para jugador azul Dada la fase de movimiento de un juego en curso con el turno del jugador azul Cuando el jugador selecciona una ficha suya Entonces las posiciones disponibles adyacentes se resaltarán como señal de “disponible” Y podrá desplazar su ficha a cualquiera de esas casillas adyacentes.	Completado	José Tello
	6.2	AC 6.2 Desplazamiento azul válido a una casilla disponible Dada la fase de movimiento de un juego en curso con el turno del jugador azul Cuando el jugador selecciona una ficha suya y la dirige hacia alguno de sus adyacentes resaltados disponibles Entonces su ficha azul es colocada en dicha posición en el Tablero Y el juego continúa su curso habitual	Completado	José Tello
	6.3	AC 6.3 Desplazamiento azul inválido hacia una celda ocupada Dada la fase de movimiento de un juego en curso con el turno del jugador azul Cuando el jugador selecciona una ficha suya y la dirige hacia alguno de sus adyacentes que se encuentra ocupado Entonces su ficha no es trasladada a ninguna posición Y el juego espera a que se realice algún movimiento válido con alguna ficha de color azul con su turno en curso.	Completado	José Tello
7. Desplazar pieza siendo jugador máquina	7.1	AC 9.1 Desplazamiento válido a una casilla disponible Dada la fase de movimiento de un juego en curso con el turno del jugador máquina Cuando la máquina utiliza su lógica para desplazar una ficha en una posición adyacente disponible Entonces su ficha roja es colocada en dicha posición en el Tablero Y el juego continúa su curso habitual	Completado	José Tello
8. “Vuelo” para jugador máquina	8.1	AC 8.1 Desplazamiento de vuelo para jugador máquina Dada la fase de vuelo para el turno del jugador máquina en un juego en curso Cuando la máquina utiliza su lógica para desplazar una ficha en una posición disponible Entonces su ficha azul es colocada en dicha posición en el Tablero Y el juego continúa su curso habitual	Completado	José Tello Pablo Quispe
9. “Vuelo” para jugador humano	9.1	AC 9.1 Identificación de casillas disponibles para el jugador rojo Dada la fase de vuelo para el turno del jugador rojo en un juego en curso Cuando el jugador selecciona una ficha suya Entonces todas las posiciones disponibles, adyacentes o no, se resaltarán como señal de “disponible” Y el jugador podrá desplazar su ficha a cualquiera de esas casillas	Completado	José Tello
	9.2	AC 9.2 Desplazamiento rojo de vuelo Dada la fase de vuelo para el turno del jugador rojo en un juego en curso Cuando el jugador selecciona una ficha suya y la dirige hacia alguna de las casillas resaltadas como disponibles Entonces su ficha rojo es colocada en dicha posición en el Tablero	Completado	José Tello

		Y el juego continúa su curso habitual		
10. Culminación del juego	10.1	AC 10.1 Culminación por insuficiencia de fichas Dado un juego en proceso Cuando algún jugador llegue a tener solo 2 fichas Entonces el jugador oponente es declarado ganador Y el juego culmina	Completado	Dagmar Lezama
	10.2	AC 10.2 Culminación por insuficiencia de movimientos Dado un juego esté en la fase de movimiento de piezas Cuando algún jugador no pueda realizar más movimientos permitidos Entonces el oponente es declarado ganador Y el juego culmina	Completado	Jose Tello

III. Actualización de las tareas de implementación

Incluye las tareas del informe anterior y resalte las tareas nuevas con un color diferente.

Resumen del código de producción.

ID Historia de Usuario y Nombre	AC ID	Nombre clase(s)	Nombre método(s)	Nombre desarrollador(es)	Status
1. Configuración inicial para empezar una partida	1.1	MenuMain	MenuMain()	Dagmar Lezama	Hecho
2. Desplegar ficha azul como jugador humano	2.1	JuegoFase1 JugadorHumano	desplegarFicha(int, int) eventClick(int, int)	José Tello	Hecho
	2.2				Hecho
	2.3				Hecho
3. Desplegar ficha roja como jugador máquina	3.1	JuegoFase1 JugadorMaquina	deployingBot() eventClick(int, int)	Pablo Quispe	Hecho
4. Retirar pieza siendo jugador humano	4.1	Juego	capturarPieza(Jugador, Point)	Pablo Quispe	Hecho
	4.2	Juego	findTri() checkStillMill() contieneCombinacion()	Pablo Quispe José Tello	Hecho
	4.3			Pablo Quispe	Hecho
5. Retirar pieza siendo máquina	5.1	JugadorMaquina	capturingBot()	Pablo Quispe	Hecho
6. Desplazar pieza siendo jugador humano	6.1	JuegoFase2	seleccionarInicio()	José Tello	Hecho
	6.2		seleccionarInicio() seleccionarDestino() moverFicha(int, int)	Jose Tello	Hecho
	6.3		seleccionarInicio() seleccionarDestino()	José Tello	Hecho
7. Desplazar pieza siendo jugador máquina	7.1	Jugador JugadorMaquina	setMoving() selectOwnFichaBot() casillasDisponiblesBot()	Pablo Quispe	Hecho
8. “Vuelo” para jugador máquina	8.1	Jugador JuegoFase2	setFlying() seleccionarInicio() seleccionarDestino()	Pablo Quispe	Hecho
9. “Vuelo” para jugador humano	9.1	Jugador JuegoFase2	setFlying() seleccionarInicio()	Jose Tello	Hecho
	9.2		seleccionarInicio()	Jose Tello	Hecho

			seleccionarDestino()		
10. Culminación del juego	10.1	Juego	isEliminar()	Pablo Quispe	Hecho
	10.2	Juego	winForNotBeingAbleToMove()	Pablo Quispe	Hecho

Resumen del código de prueba automatizado (correspondiente directamente a algunos criterios de aceptación)

ID Historia de Usuario y Nombre	**CA ID**	**Nombre de clase (s) del código de prueba**	**Nombre del método(s) del código de prueba**	**Descripción del caso de prueba (entrada y salida esperada)**	**Status**	**Nombre desarrollador(es)**
2. Desplegar ficha azul como jugador humano	2.1	HumanVsHumanDeployTest()	testDespliegueAzulValido()	Verifica que al ser la fase de despliegue, al momento de seleccionar una celda disponible se coloque una ficha azul y el turno cambie a rojo	Completo Test passed	Dagmar Lezama
	2.2	HumanVsHumanDeployTest()	testDespliegueAzulOcupado()	Verifica que al ser la fase de despliegue, al momento de seleccionar una celda ocupada no se coloque nada y el turno azul se mantenga	Completo Test passed	Dagmar Lezama
	2.3	HumanVsHumanDeployTest()	testDespliegueAzulFueraTablero()	Verifica que al ser la fase de despliegue, al momento de seleccionar una celda que no esté en el tablero, no se coloque nada y el turno azul se mantenga	Completo Test passed	Dagmar Lezama
3. Desplegar ficha roja como jugador máquina	3.1	MachineVsHumanDeployTest()	testDespliegueMaquina()	Verifica que al ser la fase de despliegue, al momento de seleccionar una celda disponible se coloque una ficha roja y el turno cambie a azul	Completo Test passed	Dagmar Lezama
4. Retirar pieza siendo jugador humano	4.1	HumanVsHumanDeployTest()	millAzulTest()	Verifica que al ser la fase de despliegue, al momento de formar un mill azul, se pueda capturar una ficha roja y el turno cambie a rojo	Completo Test passed	Dagmar Lezama
6. Desplazar pieza siendo jugador humano	6.2	HumanVsHumanMovingTest()	movingTest()	Verifica que tras haber culminado la fase de despliegue, el juego pase a la fase de movimiento y al mover una pieza, el origen quede vacío y la ficha movida se encuentre en el destino.	Completo Test passed	Dagmar Lezama

Resumen de casos de prueba manual (correspondientes directamente a algunos criterios de aceptación)

ID Historia de Usuario y Nombre	CA ID	Entrada de casos de prueba	Prueba Oráculo (salida esperada)	Status	Nombre desarrollador (es)
1. Configuración inicial para empezar una partida	1.1	El inicio de la ejecución del programa.	Menu	Completo	Dagmar Lezama
5. Retirar pieza siendo máquina	5.1	Formación de un mill de piezas rojas en el turno del jugador máquina	Eliminación de alguna pieza azul del tablero y cambio de turno al jugador humano	Completo	Pablo Quispe
6. Desplazar pieza siendo jugador humano	6.1	Selección de una ficha de color azul posicionada en el tablero	Resaltado de las casillas adyacentes disponibles	Completo	Jose Tello
	6.2	Selección de una ficha de color azul hacia una casilla adyacente disponible	Se pone la pieza azul en la casilla seleccionada y la casilla de origen queda vacía	Completo	Dagmar Lezama
	6.3	Selección de una ficha de color azul hacia una casilla adyacente ocupada	No se coloca ninguna pieza y se mantiene su turno esperando a que el jugador azul seleccione una pieza de su color hacia un sitio disponible	Completo	Dagmar Lezama
7. Desplazar pieza siendo jugador máquina	7.1	Turno del jugador humano terminado en la fase de movimiento para ambos jugadores	Movimiento de alguna pieza de color rojo hacia una posición adyacente disponible del tablero con intenciones coherentes para ganar	Completo	Pablo Quispe
8. “Vuelo” para jugador máquina	8.1	Turno del jugador humano terminado cuando haya dejado con tres piezas al jugador máquina	Movimiento de alguna pieza de color rojo hacia cualquier posición disponible en el tablero	Completo	Dagmar Lezama
9. “Vuelo” para jugador humano	9.1	Tablero en la fase de movimiento, cuando el jugador azul se encuentre en la fase de vuelo y seleccione una de sus fichas posicionadas en el tablero	Resaltado de todas las casillas, adyacentes o no, disponibles	Completo	Jose Tello
	9.2	Selección de una ficha de color azul hacia una casilla resaltada disponible	Se pone la pieza azul en la casilla disponible y la casilla de origen queda vacía	Completo	Dagmar Lezama
10. Culminación del juego	10.1	Tablero en la fase de movimiento cuando alguno de los jugadores se haya quedado solo con 2 piezas en el tablero	El juego culmina y declara como ganador al jugador que corresponda al turno contrario	Completo	Dagmar Lezama
	10.2	Tablero en la fase de movimiento, cuando alguno de los jugadores no tenga casillas resaltadas para ninguna de sus piezas en el tablero	El juego culmina y declara como ganador al jugador que corresponda al turno contrario	Pendiente	Dagmar Lezama

IV. Resumen del código fuente

Código de producción o prueba ?	Nombre del archivo de código fuente	# lineas código	Nombre desarrolladores y contribuciones (% de código fuente)
Código de producción	Ficha	9	Pablo Quispe, Dagmar Lezama
Código de producción	FichaState	2	Pablo Quispe, Dagmar Lezama
Código de producción	GameState	2	José Tello
Código de producción	Juego	136	Pablo Quispe, José Tello, Dagmar Lezama
Código de producción	JuegoFase1	18	José Tello
Código de producción	JuegoFase2	80	Pablo Quispe, José Tello
Código de producción	Jugador	31	Pablo Quispe, José Tello, Dagmar Lezama
Código de producción	JugadorHumano	8	José Tello
Código de producción	JugadorMaquina	85	Pablo Quispe
Código de producción	Tablero	102	Pablo Quispe, José Tello, Dagmar Lezama
Código de producción	TableroGUI	118	Pablo Quispe, José Tello
Código de producción	MenuMain	24	Dagmar Lezama
Código de prueba	HumanVsHumanDeployTest	132	Dagmar Lezama
Código de prueba	HumanVsHumanMovingTest	55	Dagmar Lezama
Código de prueba	MachineVsHumanDeployTest	103	Dagmar Lezama
	Total		

V. Documentación de diseño

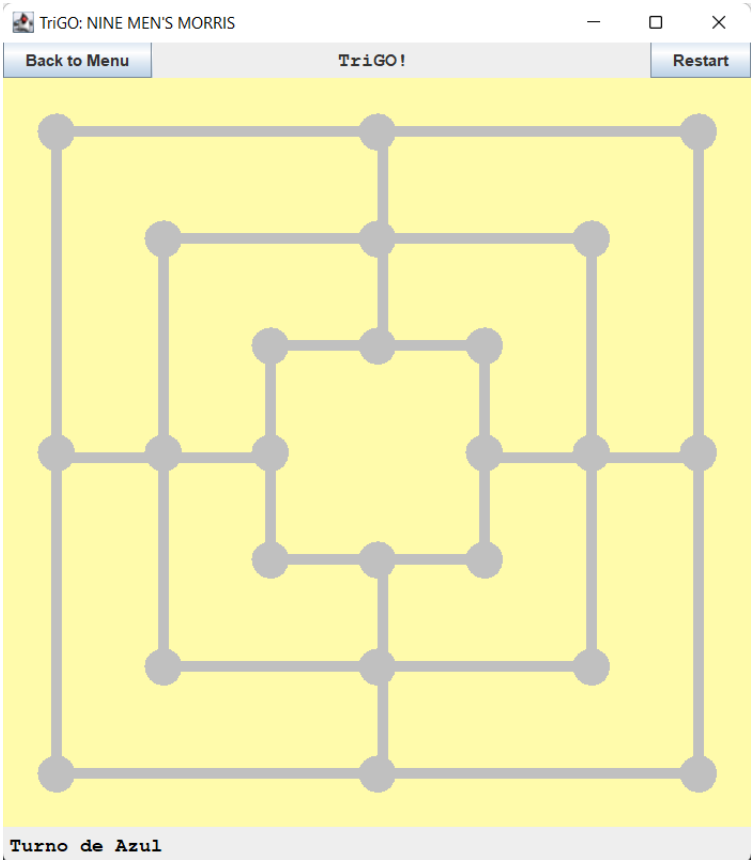
1. Diseño de interfaz de usuario

Miembros: Tello Jose, Quispe Pablo, Lezama Dagmar

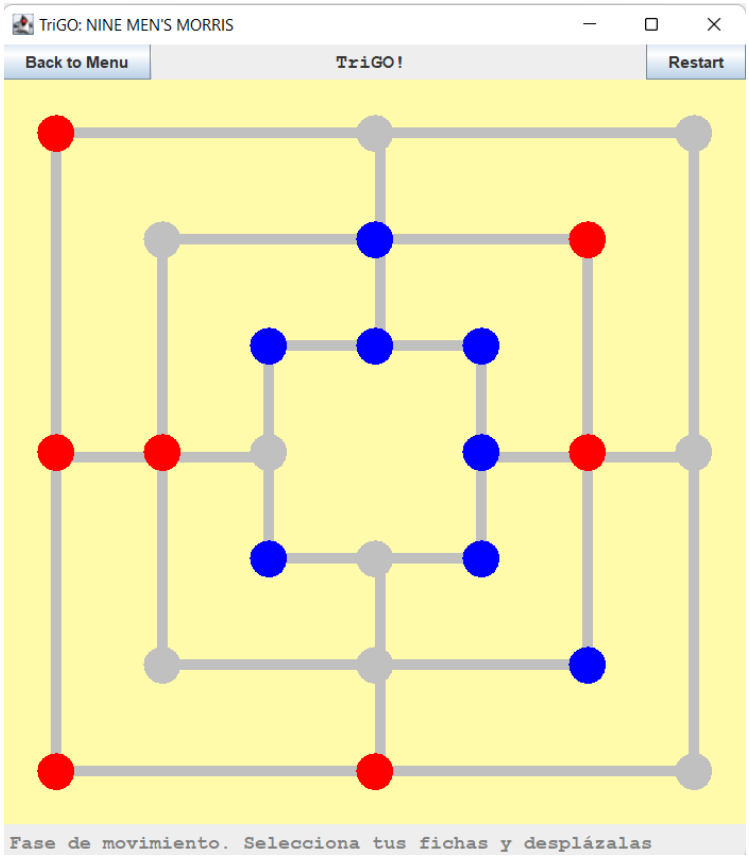
Resume el diseño de la interfaz de usuario mediante una combinación de capturas de pantalla y descripciones textuales.

Menú Principal al momento de ejecutar el juego

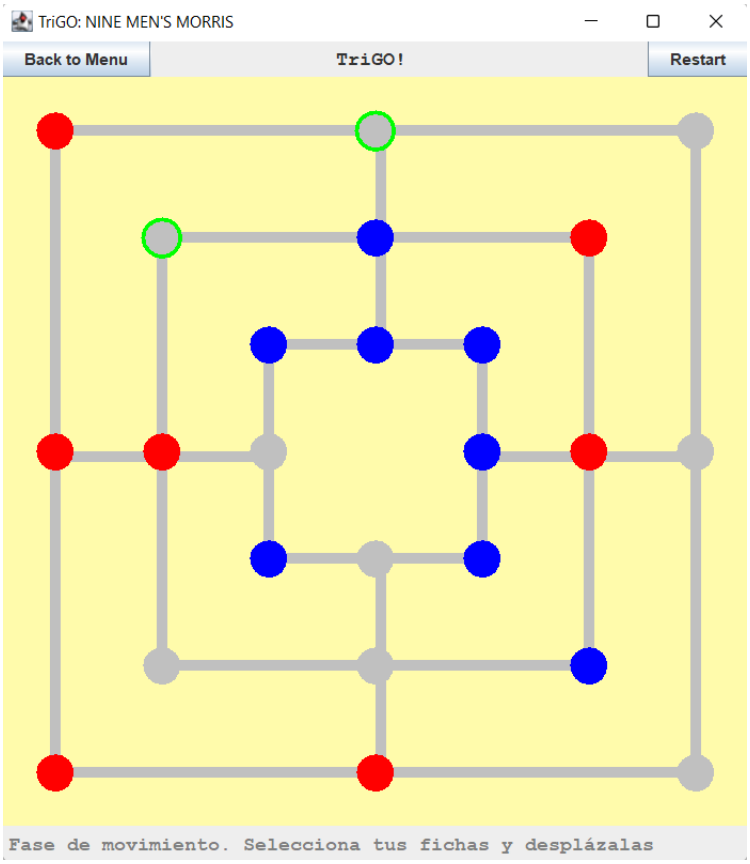
Ventana emergente al seleccionar el botón How To Play



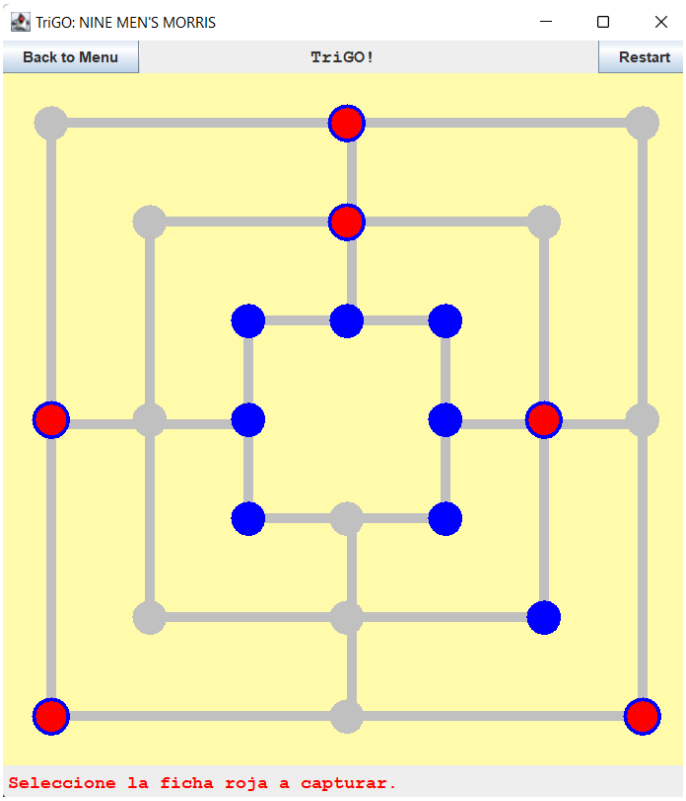
Tablero vacío tras la selección del tipo de oponente



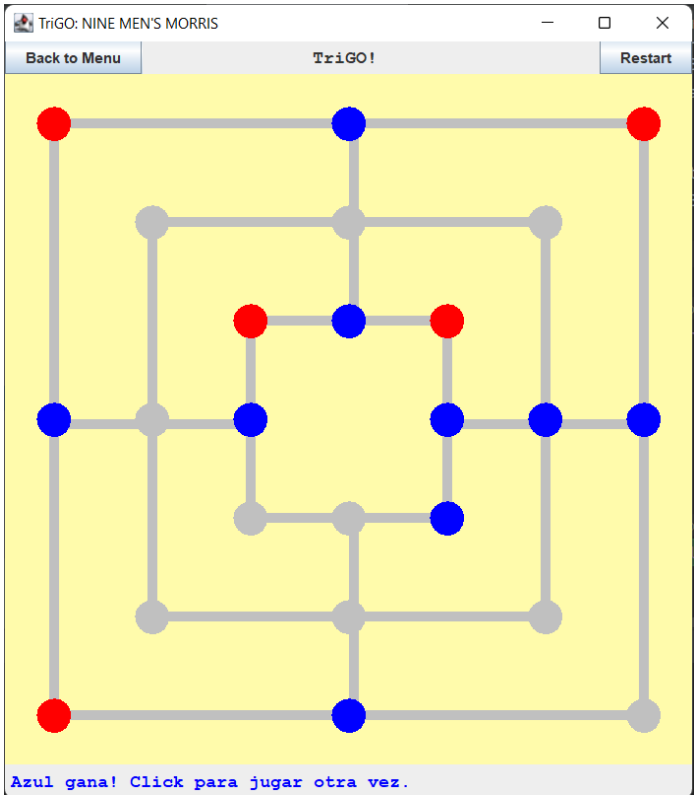
Fase de Despliegue terminada para ambos jugadores



Resaltado de lugares adyacentes disponibles al momento de seleccionar una ficha



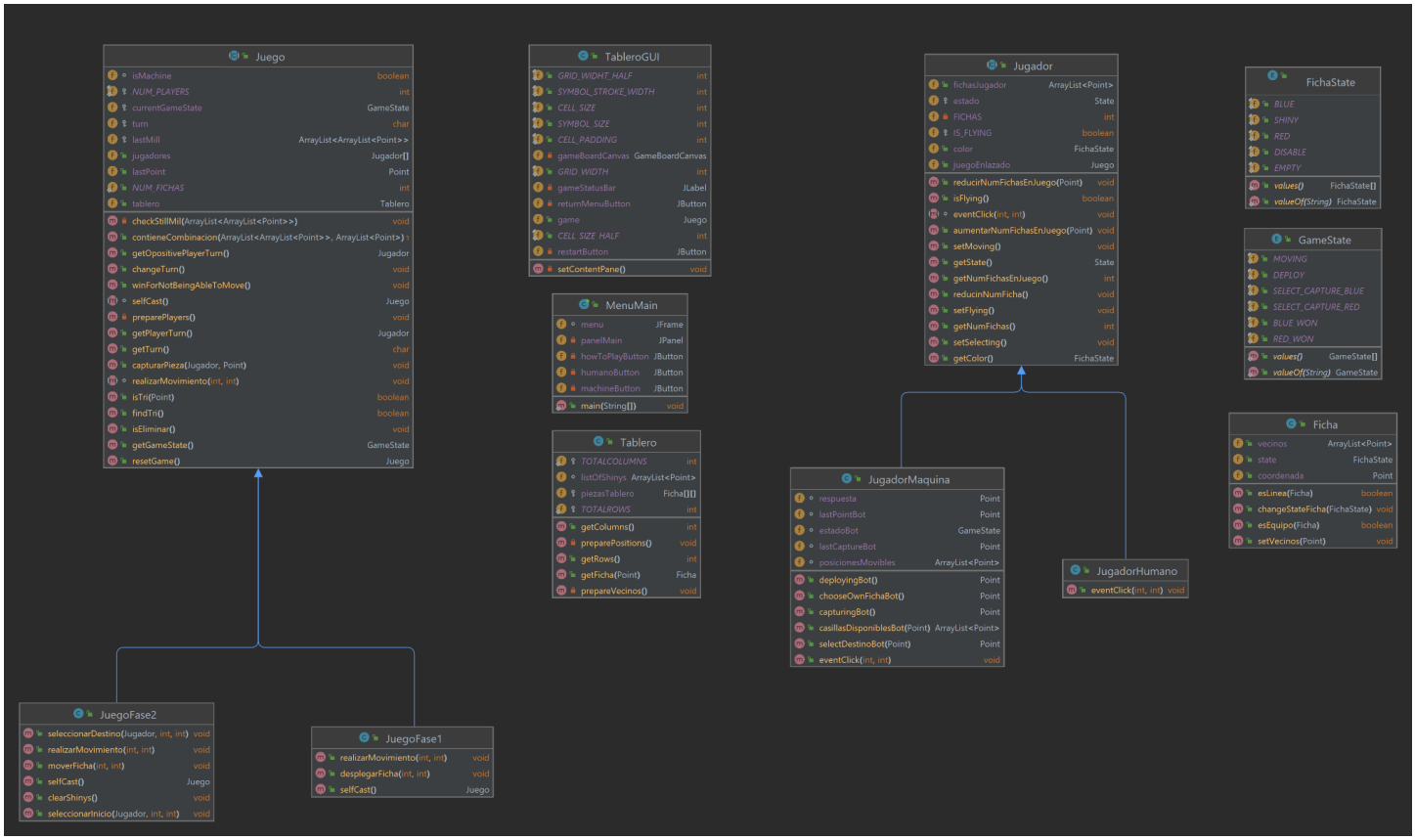
Resaltado de fichas posibles de capturar al momento de armar un tres en raya de color azul



Jugador de fichas azules resultante como ganador

2. Diagrama de clases

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



3. Diseño de algoritmos

Miembros: Quispe Pablo.

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.

```
Algoritmo BotNineMensMorrys()
        lastPosition = null
        Si eventClick() es detectado:
                Si juego.estado == “Deploy”:
                        Mientras(newPosition NO esta alineada con lastPosition)
                                newPosition = NumeroAleatorio % anchoTablero

                        MakeMove( newPosition);
                Si Entonces juego.estado == “CapturarPieza”:
                        index = NumeroAleatorio % listaPiezasEnemigas.size()
                        MakeMove( ListaPiezasEnemigas[index]);
                Si Entonces juego.estado == “Moviendo”:
                        index = NumeroAleatorio % listaPiezasPropias().size()
                        MakeMove( listaPiezasPropias[index]);
```

```
                    Si isFlying:
                        index = NumeroAleatorio % posicionesLibreVecinas.size()
                    Entonces:
                        index = NumeroAleatorio % posicionesLibre.size()
                    MakeMove( posicionesLibresTotales[index]);
FinAlgoritmo
```

4. Extensibilidad

Miembro: Quispe Pablo.

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. ¿Qué clases y métodos deben cambiarse y cómo? ¿Cómo se aplicó el principio abierto-cerrado (es decir, qué funciones o clases están abiertas para la extensión, pero cerradas para la modificación)? o algún otro principio de SOLID

Para ampliar el código a las demás variantes se deben implementar una clase heredada de la clase tablero que sobreescriba la el método que inicializa las posiciones del tablero, ya que cada tablero es una matriz de objetos Ficha, los cuales están entrelazadas mutuamente entre aquellas que sean adyacentes bajo la disposición del tablero dado. Esta nueva disposición de los enlaces entre posiciones, y el redimensionar los constructores bastarían para esta finalidad, dado que el dibujado del tablero también puede darse reutilizando esa arquitectura de direccionamiento que facilita casi todo el sistema del juego abstrayendo en un grafo bidireccional.

El principio de abierto-cerrado está aplicado a toda la lógica del juego, pues de querer añadirse otras funcionalidades tales como una partida online, bastaría con crear un flujo de datos al método realizar movimiento de la clase Juego, ya que todo el funcionamiento es abstraído internamente. Por otra parte si se quisiera añadir una nueva fase del juego, tambien solo se tendría que extender la clase abstracta Juego, y que así sea incluida en la secuencia de fases gracias al método selfCast(), que permite que una clase “se castee a sí misma” y así utilizar polimorfismo haciendo que cada una de las fases sean completamente independientes entre sí. Por otra parte si se desea añadir otro tipo de jugador, o conectarlo a otro tipo de bot o jugador, tambien solo tendría que extenderse la clase abstracta Jugador, dado que también todos los tipos de jugadores son independientes entre sí, y son creados por el juego acorde a la elección del usuario desde el menú principal. Por tanto, las clases están abiertas a nuevas implementaciones, pero cerradas cualquier modificación interna de las mismas.

VI. Conclusiones del ejercicio de revisión del código

Utiliza la siguiente plantilla para documentar los resultados de la revisión del código.

Nombre de los participantes: José Tello, Pablo Quispe, Dagmar Lezama
Clase que fue revisada: **Juego.java**

Checklist	Items Checklist	Conclusiones
Estándares de codificación	Convenciones de nombres	Se utilizan las convenciones para nombres recomendadas por Google para Java, y se alternan entre nombres en español e inglés para los métodos, el que sea más corto tiene prioridad.
	Convención de ordenación de argumentos de método	Se priorizaron los tipos básicos de java (int, double, char, …) y seguidos de estos, las clases definidas para el programa.
	Comentarios significativos y válidos.	Los comentarios sirvieron para descartar código a la hora de refactorizar.

	Estilo consistente de bloques de código	Se siguieron las convenciones para la correcta escritura de código en Java impartidas por Google.
	Indentación consistente	Se usaron tabulaciones de 4 espacios para la identación en el código y así asegurar que sea legible.
Principio de diseño	Buena abstracción de clase e interfaz	Las subclases usan totalmente el código heredado de sus superclases, e implementan de manera individual sus respectivos métodos únicos, de manera que no hay repetición de clases y código.
	Visibilidad adecuada de cada variable, método y clase.	Los atributos críticos para cada clase están encapsulados dentro de las mismas, los métodos y clases son públicas para que puedan verse dentro del mismo paquete y para futuras mejoras poder ser importadas sin ser extraídas.
	Alguna violación del principio de separación comando-consulta[2]	Todas las llamadas a métodos están condicionadas de manera que se cubran todas las posibles circunstancias en las que deben ser llamados.
	Diseño por contrato (pre/postcondiciones)	Todos los métodos son llamados solamente si cumplen sus respectivas precondiciones
	¿Se viola el Principio Abierto-Cerrado?	Al principio sí, pues cambiar el juego para añadirle funcionalidades nuevas implicaba cambiar de gran manera la clase Juego. Sin embargo, luego de la refactorización, todas las clases están abiertas para nuevas implementaciones sin alterar su funcionamiento principal.
	¿Se viola el Principio de Responsabilidad Única [3]?	No, pues al refactorizar tenemos bien definidas las funciones de cada clase y su independencia en funcionalidad de cada una.
Smells código	Números mágicos	Cada número tiene asociada una constante que explica el significado de dicho número
	Variable global /clase innecesaria	Todas las clases y variables globales cumplen una función específica
	Código duplicado	El código duplicado se eliminó al refactorizar el código para el Sprint 3
	Métodos largos	Los métodos no suelen pasar de las 25 líneas, y aquellos que lo hacen pertenecen a Clases como la de TableroGUI y Tablero, que no afectan a la lógica base necesaria para entender el programa.
	Larga lista de parámetros	No hay parámetros de más luego de la refactorización, todos los que existen un valor u objeto indispensable en el funcionamiento del programa.
	Expresión demasiado compleja	Hay líneas de código que son extensas y complejas de leer debido a la longitud de los nombres de los objetos, variables y métodos a usar.
	Switch o if-then-else que necesita ser reemplazado con polimorfismo	Se reemplazó con polimorfismo el cambio de fases del juego, creando dos subclases de la superclase Juego que representen las fases de despliegue y de movimiento. Cuando se han desplegado todas las fichas, el juego se cambia a una instancia de JuegoFase2.
	Nombre de método o variable cuya intención no está clara	Todos los métodos y parámetros de la clase Juego tienen nombres claros que especifican sus propósitos.
	¿Algún método similar en otras clases?	La clase Juego no comparte similitudes con otras clases, más que con sus subclases, las cuales sobreesciben métodos dependiendo de la fase en la que se encuentre el Juego.

Luego de realizar numerosas pruebas y juegos distintos, no se detectaron errores a la hora de desarrollar el juego de inicio a fin

VII. Acta de reuniones

[2] Revisa: Writing professional code with command-query separation
[3] Revisa: Violation solution for single responsibility principle

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 y Depuración	Lugar	Nombre Participantes	Propósito de la reunión	Elementos de acciones específicos
13/11/22	1h 10 min	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Reunión retrospectiva	Analizar las deficiencias del código en el sprint #2 y presentar propuestas para solucionarlas Coordinar las nuevas futuras reuniones
16/11/22	2h	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Primera revisión del Sprint #3	Analizar la cantidad de trabajo a implementar para este Sprint Completar las historias de usuario basándonos en las propuestas en el Sprint #2
19/11/22	2h 20min	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Avance de documento y enfoque SOLID	Modificar los criterios de aceptación Examinar la estructura del código para proponer una refactorización que se base en el enfoque de los principios SOLID
23/11/22	1h 10 min	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Revisión del avance del documento Sprint #3	Refinar los criterios de aceptación para cada historia de usuario modificada Dividir otras responsabilidades que conciernen al documento de Sprint
26/11/22	1h	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Refactorización principal	Modificar el diseño de la estructura del juego utilizando las ideas de refactorización planteadas anteriormente
27/11/22	45 min	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Refactorización	Dividir responsabilidades para poder mantener la lógica propuesta inicialmente con la estructura del código refactorizado
27/11/22	2h 30 min	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Revisión del avance del código	Revisar los avances individuales correspondientes al código refactorizado del juego
28/11/22	3h	Presencial	Dagmar Lezama Jose Tello Pablo Quispe	Programación en pares para el avance de la lógica	Avanzar las implementaciones en conjunto a través de la herramienta Codewithme
29/11/22	6h	Presencial	Dagmar Lezama Jose Tello Pablo Quispe	Programación en pares para arreglar ciertos bugs	Avanzar las implementaciones en conjunto a través de la herramienta Codewithme Avanzar los campos faltantes de documento
30/11/22	4h	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Revisión del juego en conjunto	Revisar los avances individuales del código Actualizar del Trello y realización de pruebas manuales y automatizadas Verificar el correcto funcionamiento del juego Proponer últimas mejoras
02/12/22	3h	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Revisión del documento del Sprint #2	Concluir con el documento Sprint #2
04/12/22	1h 45min	Discord (online)	Dagmar Lezama Jose Tello Pablo Quispe	Revisión del Sprint #2	Revisar últimas modificaciones del Sprint Refactorizar ciertos puntos finales del código
06/12/22	1h	Presencial	Dagmar Lezama Jose Tello Pablo Quispe	Revisión final	Revisión final del informe
07/12/22	3h	Presencial	Dagmar Lezama Jose Tello	Entrega final del Sprint	Realizar la grabación y edición del video de la presentación final del Sprint #3

			Pablo Quispe		

VIII. Calificación de amigos

Si no te sientes cómodo al incluir tus calificaciones en este informe, puedes enviarlas por correo electrónico al profesor.

Calificación receptor

	Lezama V. Dagmar	Quispe O. Pablo	Tello L. Jose
Lezama V. Dagmar	X	20	20
Quispe O. Pablo	20	X	20
Tello L. Jose	20	20	X
Promedio	20	20	20

Observaciones

1 . Se pide la participación de cada estudiante, ningún estudiante puede quedarse sin participar ya que eso afectará su calificación y la de su grupo. Un estudiante que no participe en ningún sprint tendrá la nota de **No participó 0**. **Todos los estudiantes deben participar en cada sprint. La nota no es igual para todos los estudiantes, depende de tu participación. Eso significa que se tomará en cuenta la asistencia, la resolución de los cuestionarios y la verificación de sus repositorios en el archivo excel de la clase.**

3. Durante la emisión de cada vídeo deben utilizar un tablero de Trello para poder justificar las actividades de scrum utilizados. Sugerencia: https://blog.trello.com/es/metodologia-scrum-agiles. Es imperativo mostrar el orden de sus actividades

4. Se evaluará la forma de presentarse y los términos utilizados desarrollados en clase.

5. Al final del sprint, tu equipo demostrará el software en funcionamiento. La demostración debe mostrar que su software de trabajo ha cumplido con las condiciones mínimas de "Definiciones de Listo" del sprint correspondiente y presentar características únicas o mejoras de su proyecto.

6. Las instrucciones detalladas y las plantillas de los informes del proyecto se proporcionarán en documentos separados. Los informes serán incrementales, ampliados y mejorados de un sprint al siguiente.

7. El informe individual debe ser lo más técnico posible, se bajaran puntos por fallas ortográficas y por comentarios poco técnicos que no estén en sus presentaciones. Los miembros del equipo deben asegurarse de utilizar la mayor cantidad de definiciones vistas en clase, ya que eso es un parámetro de calificación.

8. En cada vídeo se debe mostrar los avances de un sprint a otro, si eso no está claro entonces el proyecto va por mal camino.

9. Parte de la calificación es el avance por github y Trello. Se revisará la analítica de los avances y los tableros de trello.

10. La entrega individual del proyecto se entregará a parte de las entregas grupal y es requisito necesario para ser evaluado en esta asignación. Además que repercutirá en la evaluación final del grupo.

11. En este sprint no hay retroalimentación por lo que los grupos deben asegurar de presentar las cosas de acuerdo a los criterios de evaluación.