Reporte del Sprint #5

Las principales tareas de esta asignación son:

(1) Agrega la función de grabar (record) un juego en un archivo de texto. Se requiere la historia de usuario y los criterios de aceptación tanto de grabación como de reproducción
(2) Realización de un ejercicio de revisión de código.
(3) Resumir las lecciones aprendidas del Sprint 0 al Sprint 5.

El siguiente es un diseño de GUI de muestra del producto final, donde "Replay" es opcional.

El trabajo es de caracter individual.

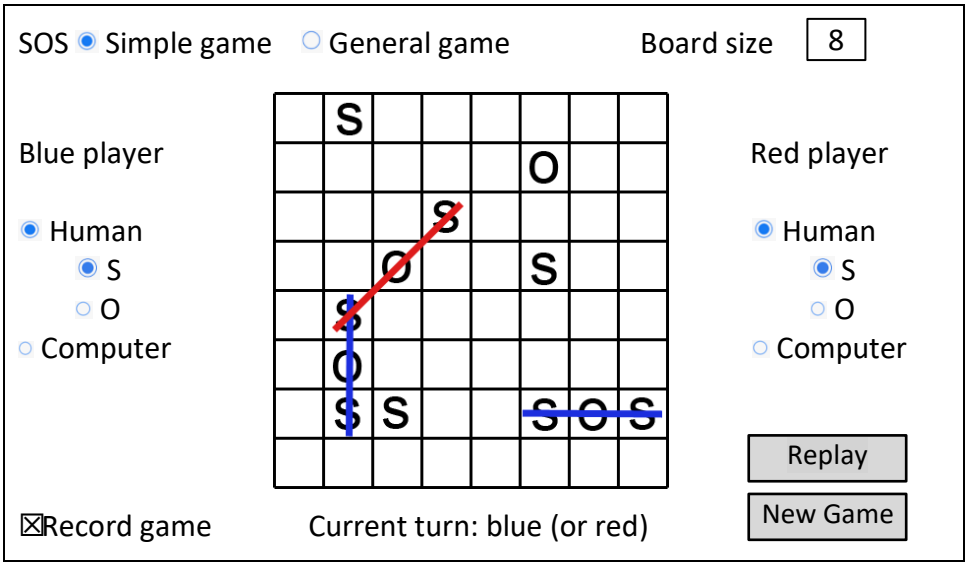

Figura 1. Sample GUI layout of the final product Diseño de GUI del producto final

Puntos totales

1. Demostración (10 puntos)

Envía un video de no más de 15 minutos, demostrando claramente que has implementado todas las funciones en la siguiente tabla. En el video, debes explicar lo que se está demostrando. **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.**

	Feature
1	Se graba un juego simple completo de dos jugadores humanos.
2	Se graba un juego general completo de dos jugadores humanos
3	Se graba un juego simple completo de jugadores humano-computadora
4	Se graba un juego general completo de jugadores humano-computadora
5	Se graba un juego simple completo de jugadores computadora-computadora
6	Se graba un juego general completo de jugadores computadora-computadora

Si has implementado la función de "replay" para obtener crédito adicional, debes incluir tu demostración en el video.

2. Historias de usuario y criterios de aceptación para los requisitos para los requerimientos Record/Replay (1 punto)

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

Agrega o elimina filas si es necesario

ID	Nombre de historia de usuario	Descripción de historia de usuario	Prioridad	Esfuerzo estimado (horas)
12	Grabar juego en archivo de texto	Como usuario solicito que el desarrollo del juego se guarde en un archivo de texto.txt para la visualización de las jugadas realizadas	5	7
13	Reproducir juego de un archivo de texto	Como usuario solicito que el archivo de texto donde se guarda el desarrollo del juego pueda ser reproducido en el tablero SOS	2	…

ID y nombre de la historia de usuario	AC ID	Descripción del criterio de aceptación	Estado (completado, por hacer, en progreso)
12	20.1	AC 20.1 <Grabación de un juego completo> Given se desea grabar un juego en un archivo txt When se inicializa un juego And se desarrollan las jugadas And se terminar un juego Then se guarda un archivo txt con los pasos del juego	Completado
	20.2	AC 20.2 <Grabación de un juego incompleto> Given no se finaliza un juego When se realizan los turnos And se cierra antes de terminar el juego Then se guarda el archivo txt hasta donde se realizó el juego	Completado
13	21.1	AC 21.1 <Reproducir juego> Given se quiere visualizar el archivo txt en el juego When se da clic en el botón Record Then se da la visualización del juego en el tablero	En Progreso

3. Revisión de código (4 puntos)

Aplica la revisión del código fuente a una o dos de las clases más importantes (y a otras clases si el tiempo te permite) e informa de los resultados. 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 olores de código que indican la necesidad de refactorización.

Las siguientes listas de verificación proporcionan pautas básicas. Puedes agregar nuevos elementos a cada una de las listas de verificación. Asegúrate de que tus respuestas sean el resultado del ejercicio de revisión del código. Si no hay hallazgos para una entrada, debes proporcionar una explicación. Por ejemplo, si tu respuesta a

"¿Se violan las convenciones de nomenclatura?" es no, debes describir una convención de nomenclatura y presentar un ejemplo. No recibirás puntaje por si tus respuestas son simplemente sí o no sin información adicional.
Clases que han sido revisadas: ClassPlayer, GuiSOS, ClassGeneralGame, ClassScore, ClassSimpleGame
Fecha/hora de duración del ejercicio de revisión del código: 3 semanas

Checklist	Items Checklist	Conclusiones
Estándares de codificación	Convenciones de nombres	Se utilizó nombres identificadores, es decir nombres los cuales sirven para identifiquen fácilmente lo que un método, clase, variable, parámetro, etc. realiza o identifica de modo que ésta trate de estar definida por sí misma. Realizado tanto en el código de producción como en el código de pruebas.
	Convención de ordenación de argumentos de método	Podemos definir como métodos secundarios a aquellos de prueba corta que se utilizar para auxiliar a otros métodos, dichos métodos están definidos en su mayoría al principio de otra clase. Mientras aquellos métodos que realizan la carga más importante del trabajo están definidos en la parte final de la clase.
	Comentarios significativos y válidos.	Los comentarios realizados son cortos para dar una orientación de lo que realiza cada instrucción, no es necesario realizar comentarios en cada línea ya que, gracias a la convención de nombres, estos pueden ser entendidos fácilmente.
	Estilo consistente de bloques de código	Se separó la mayoría de funciones repitentes para una mejor visualización del código, en el método Game de la clase GuiSOS hay dos funciones que realizan acciones muy similares, sin embargo, se optó por realizarlo de esta manera ya que el conocimiento sobre el uso de JButtom es intermedio.
	Indentación consistente	La declaración de la funcionalidad de un método está separada con un espacio de los demás métodos para realizar una mejor visualización del espacio de trabajo.
Principio de diseño	Clase o método no bien modularizado	Puede que la clase GuiSOS pueda estar no muy bien modularizada. Esta clase es clave para la funcionalidad del código ya que esta almacen los java.swing y java.awt entonces cuando se trata de trabajar con botones se trató de realizar todo en dicha clase. Punto que hay que rescatar es que se ha separado en clases las funciones del juego y además se ha utilizado herencia así pues se ha tratado de entablar un aspecto modularizado.
	Visibilidad adecuada de cada variable, método y clase.	Considero que es un sí ya que, al separarse en clases, utilizando la convención de ordenación de argumentos de método y las identaciones. Entonces tengo una mejor visualización de las partes importantes y no se aprecia un código sobrecargado.
	Alguna clase con pobre abstracción	Si hablamos de dos bandos codificación lógica y codificación visual. Podemos decir que la codificación netamente lógica está separa de la visual, sin embargo existen ciertas funcionalidades que deben ir de la mano con la parte visual ya que se usan Jbuttoms. Las clases de ClassSimpleGame y ClassGeneralGame no están del todo abstraídas ya que estas solo comparan el resultado final de juego mas no compara si el proceso se realiza. Es decir, para saber si se realiza un movimiento por segunda vez, esta función la realiza GuiSOS en el método generalGame el cual recibe parámetros Jbuttoms. Para comparar quien gana se realiza en la clase ClassGeneralGame.
	Diseño por contrato (pre/postcondiciones)	Se realiza el uso de las precondiciones al momento de exigir que se ingrese un n en un rango determinado, se utilizan las postcondiciones cuando se usan los mensajes de confirmación para saber quién gana.

	¿Se viola el Principio Abierto-Cerrado?	No, considero que aquella información que merece ser no modifica como en ClassPlayer al momento de declarar un jugador permanece así. Es inaccesible a modificaciones. A su vez se utilizan métodos públicos y privados los cuales realizan trabajos propios de la clase, y se necesita realiza un llamado a otro método se determina su inicialización mas no su modificación para ser usado ejem: se utiliza score.setScorePlayer2(…) una vez ya inicializado score en GuiSOS	
	¿Se viola el Principio de Responsabilidad Única [1]?	No, ya que no por ejemplo para la clase ClassPlayer se define los parámetros a introducir con ClassPlayer(String player, boolean isHuman), cada clase esta usada para su propósito en específico.	
Smells código	Números mágicos	No, se trató de utilizar parámetros los cuales su uso está de fácil compresión gracias a sus nombres	
	Variable global /clase innecesaria	No se utilizan variables globales y el código no presenta clases que no implementa	
	Código duplicado	Sí, pero solo en una sección, en la de Game de la calse GuiSOS, esto debido a que se tiene un conocimiento intermedio sobre botones entonces las dos funcionalidades que presentan son casi iguales pero necesarias.	
	Métodos largos	Solo en la sección Game de la clase GuiSOS	
	Larga lista de parámetros	Sí en GuiSOS	
	Expresión demasiado compleja	Se utilizan métodos que llaman a lo más 1 o 2 métodos de otras clases	
	Switch o if-then-else que necesita ser reemplazado con polimorfismo	Sí en la sección de GuiSOS en el método Game cuando se trata de hacer una diferenciación entre humano y computador para separar el tipo de movimiento.	
	Nombre de método o variable cuya intención no está clara	No, ya que se separó en clases la compresión es más entendible. Además, los comentarios ayudan a su interpretación.	
	¿Algún método similar en otras clases?	No, tal vez se tenga en cuenta al método generalGame el cual recibe el parámetro Jbuttoms con la clase ClassGeneralGame (lo mismo aplica para el juego simple), estos se diferencian en que el primero procesa la puntuación a lo largo del juego y por ende los movimiento, mientras que el segundo procesa la condicional de victoria.	
Errores	**Fragmento de código con errores**	**¿Cuál es el error?**	**¿Por qué es un error?**
	Sección Game de la Clase GuiSOS	Las funcionalidades que realizan para un jugador humano y computador	Dichas funcionalidades son muy similares y se repite la utilización de métodos (se usan 2 veces) que podrían ser usados solo 1 vez.

4. Resumen de todo el código (1 points)

Nombre del archivo de código fuente	Código de producción o de prueba	# lineas de código
Board	Producción	19
ClassGeneralGame	Producción	25
ClassPlayer	Producción	22
ClassScore	Producción	198
ClassSimpleGame	Producción	31
GuiSOS	Producción	537
TestComputerFinishGeneralGame	Prueba	143
TestComputerFinishSimpleGame	Prueba	122

[1] Revisa: Violation solution for single responsibility principle

TestComputerMoveGeneralGame	Prueba	55
TestComputerMoveSimpleGame	Prueba	52
TestGameMode	Prueba	34
TestHumanAndCompFinishGeneral Game	Prueba	142
TestHumanAndCompFinishSimpleG ame	Prueba	125
TestHumanAndCompMoveGeneral Game	Prueba	55
TestHumanAndCompMoveSimpleG ame	Prueba	52
TestHumanFinishGeneralGame	Prueba	147
TestHumanFinishSimpleGame	Prueba	127
TestHumanMoveGeneralGame	Prueba	55
TestHumanMoveSimpleGame	Prueba	52
TestNewGame	Prueba	29
TestSizeBoard	Prueba	42
TestWhoIam	Prueba	44
	Total	2108

No recibirás puntaje por esta tarea a menos que envíes tu código fuente completo.

5. Resume las lecciones aprendidas de todo el proyecto respondiendo las siguientes preguntas desde la perspectiva de los procesos de desarrollo, codificación, diseño, refactorización y prueba (**4 puntos**):

- ¿Qué ganaste personalmente con el proyecto?
- ¿Qué hace bien tu proyecto y qué podría hacer mejor tu proyecto?
- ¿Cómo podrías mejorar tu proceso de desarrollo si desarrollas un juego similar desde cero?

Requisito mínimo para (5): Una página completa a espacio simple, tamaño de fuente no mayor a 12 puntos.

- ¿Qué ganaste personalmente con el proyecto?

Antes no tenía conocimiento sobre el uso de botones en java, sin embargo, a lo largo del desarrollo del proyecto me vi en la oportunidad de explorar mis conocimientos utilizando dichas herramientas de java. También, pude desarrollar mis habilidades en POO los cuales tenía una formación intermedia y ahora tengo una mucha mejor compresión de dicha herramienta. En los primeros Sprint se realizó toda la lógica del proyecto y la visualización en una misma clase, si bien estos se habían separados en métodos, esto no era suficiente por lo que hacer uso de la refactorización era muy necesario sobre todo a la hora de implementar un jugador maquina y un jugador humano, por lo que para una mejor visualización del mismo fue importante separar el código en clases y usar herencias. Lo nuevo que aprendí fue la realización de la documentación, si bien se tiene una buena memoria cuando se realiza un proyecto, no se puede implementar algo que no existe, es decir, se tiene que tener en claro qué es lo que se va a desarrollar entonces el uso de las historias de usuario son muy importantes, así mismo de los criterios de aceptación; además al tener un registro del proceso de trabajo con ayuda de los Sprints pude

entender qué es lo que podría estar fallando en mi código o qué es lo que le hacía falta. El desarrollo de pruebas, si bien puede parecer que tu código puede funcionar por sí solo y ocurre un error inesperado, no se sabe qué es lo que fallo, es por eso que al momento de realizar pruebas se puede seccionar la funcionalidad del código de modo que cada trozo del mismo pueda ser probado y saber qué falla, identificar el error y corregirlo.

- ¿Qué hace bien tu proyecto y qué podría hacer mejor tu proyecto?

Si bien mi proyecto realiza la ejecución del trabajo solicitado de forma correcta de manera externa, por dentro podría separar aún más algunos métodos o en clases algunas funcionalidades, por ejemplo, una clase llamada movimiento para así poder realizar una mejor visualización del código. Además se tiene que mejorar la parte de Game de la class GuiSOS para que no tengan una funcionalidad muy similar entre entre las condicionales humano y computador para no "repetir código". También implementar que el método de análisis de puntos del método generalGame de GuiSOS esté dentro de la clase ClassGeneralGame (lo mismo para la funcionalidad de juego simple) porque no es correcto visualmente ni estructuralmente. Se pueden también realizar más pruebas que desarrollen aspectos menos generales, es decir se podría probar cada una de las clases en lugar de probar solamente la clase principal GuiSOS.

- ¿Cómo podrías mejorar tu proceso de desarrollo si desarrollas un juego similar desde cero?

En primer lugar, proyector mi código a futuro. Es decir, si bien en primera instancia el bosquejo del programa es eficiente, posteriormente cuando el trabajo esté escalando se va a tener que implementar las jerarquía de clases por lo que es necesario codificar el ahora pero pensando en el futuro ya que sino se puede complicar a la hora de querer separar la parte visual con la parte lógica del proyecto. Hay que respetar los plazos, se trabaja de una mejor manera si se instancia plazos de trabajos de modo que se desarrollo un proyecto por trozos ya que es imposible terminar un proyecto correcto al 100% de un día para otro, por lo que desarrollarlo de manera seccionada es mucho mejor.