**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.**

|  |  |  |
| --- | --- | --- |
| SOS Icon  Description automatically generated Simple game Icon  Description automatically generated General game Board size  8 | | |
| Blue player  Icon                                Description automatically generated Human  Icon  Description automatically generated S  Icon  Description automatically generated O  Icon                                Description automatically generated Computer | Chart, line chart  Description automatically generated | Red player  Icon  Description automatically generated Human  Icon  Description automatically generated S  Icon  Description automatically generated O  Icon  Description automatically generated Computer  Replay |
| ☒ Record game | Current turn: blue (or red) | New Game |

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.

1. **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)** |
| 20 | Grabar Juego | Como jugador  Quiero tener la opción de grabar una partida de SOS Para poder ver qué movimientos se realizaron paso a paso en el tablero | ALTA | 2 |
| 21 | Reproducir Juego Grabado | Como jugador  Quiero reproducir una partida de SOS grabada  Para ver de paso a paso cómo se jugó esa partida | MEDIA | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID y nombre de la historia de usuario** | **AC**  **ID** | **Descripción del criterio de aceptación** | **Estado (completado, por hacer, en progreso)** |
| Historia 20 | 20.1 | AC 20.1 <Grabar Juego Simple Humano vs Humano>  Dado un juego simple de Humano vs Humano  Cuando esté marcada la opción de grabar y acabe el juego  Entonces se crea un archivo de texto que contiene las jugadas realizadas paso a paso. | COMPLETADO |
| 20.2 | AC 20.2 <Grabar Juego Simple Computador vs Computador>  Dado un juego simple de Computador vs Computador  Cuando esté marcada la opción de grabar y acabe el juego  Entonces se crea un archivo de texto que contiene las jugadas realizadas paso a paso. | COMPLETADO |
| 20.3 | AC 20.3 <Grabar Juego General Humano vs Humano>  Dado un juego general de Humano vs Humano  Cuando esté marcada la opción de grabar y acabe el juego  Entonces se crea un archivo de texto que contiene las jugadas realizadas paso a paso junto con el puntaje de cada jugador. | COMPLETADO |
| 20.4 | AC 20.4 <Grabar Juego General Computador vs Computador>  Dado un juego general de Computador vs Computador  Cuando esté marcada la opción de grabar y acabe el juego  Entonces se crea un archivo de texto que contiene las jugadas realizadas paso a paso junto con el puntaje de cada jugador. | COMPLETADO |
| Historia 21 | 21.1 | AC 21.1 <Reproducir Juego Simple Grabado>  Dado un archivo de texto que contenga un juego simple grabado  Cuando se reproduzca dicho archivo  Entonces se muestran los movimientos paso a paso en pantalla | POR HACER |
| 21.2 | AC 21.2 <Reproducir Juego General Grabado>  Dado un archivo de texto que contenga un juego simple grabado  Cuando se reproduzca dicho archivo  Entonces se muestran los movimientos paso a paso en pantalla | POR HACER |

**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:

Fecha/hora de duración del ejercicio de revisión del código:

Revisaré las clases Tablero, Juego y un poco de Controller

|  |  |  |  |
| --- | --- | --- | --- |
| **Checklist** | **Items Checklist** | **Conclusiones** | |
| Estándares de codificación | Convenciones de nombres | Desde el inicio del proyecto se han seguido las convenciones, como podría ser escribir los métodos en Pascal Case. Como ejemplo: public void PrintGame(). También camel Casing, como ejemplo: public Juego(Tablero tablero). Sin embargo, se nos pasó las convenciones de no usar mayúsculas para constantes o variables de solo lectura, y la convención de no usar underlines antes de una variable sin asignar, sin embargo se hizo para evitar una mala admistración de variables: public override void MakeMove(int row, int column, Tablero.Cell \_ficha) por public override void MakeMove(int row, int column, Tablero.Cell ficha). Además, una pequeña omisión fue que el nombre del proyecto sea ReSOSgame en vez de ReSOSGame. | |
| Convención de ordenación de argumentos de método | C# no cuenta con esta convención per se, pero hemos ordenado de forma consistente los métodos. De esta forma el ejecutar los métodos ya sea dentro del código en las pruebas no se hizo confuso ni tedioso. | |
| Comentarios significativos y válidos. | A lo largo del código se logró comentar gran parte del código, los métodos, atributos y clases se han intentado hacer lo más autoexplicativos posibles. | |
| Estilo consistente de bloques de código | Los atributos y propiedades se han puesto en la parte de arriba de los bloques de código y luego los métodos. Hubo omisiones que ya se han corregido. El orden es: Atributos, Propiedades, Métodos. | |
| Indentación consistente | En aras de lograr un código ordenado y comprensible se ha mantenido una correcta identación. | |
| Typos | Hay muchos typos señalados por el IDE, sin embargo, estos se deben a que hay variables en español y los comentarios están en español también, por lo que después de revisar que no se tratan de violaciones de nomenclatura, se ha puesto como comentario. | |
| Principio de diseño | Clase o método no bien modularizado | En un principio la clase controller se encargaba no solo del juego, sino que también de su grabación. (Hace unos minutos). Pero ahora existe la clase GameRecorder que se encarga de la grabación del juego. | |
| Visibilidad adecuada de cada variable, método y clase. | Había variables y métodos cuya visibilidad no era la adecuada, muchas estaban en public cuando bien podrían estar en private o en el caso de muchas Propiedades, en protected. Con cada sprint se ha mantenido constante la omisión de buena encapsulación, sin embargo, en esta revisión de código he cambiado muchas de estas visibilidades, principalmente en las propiedades. | |
| Alguna clase con pobre abstracción | Se ha tenido muy en cuenta la abstracción para sacarle el mayor provecho al polimorfismo. Sin embargo, hay métodos o atributos que hacen que las clases estén estrechamente ligadas, como es el caso de Tablero y Juego, pues juego necesita de un tablero para poder trabajar. Quizá haya otro enfoque para trabajar esto. | |
| Diseño por contrato (pre/postcondiciones) |  | |
| ¿Se viola el Principio Abierto-Cerrado? | En este caso no, tal como pide este principio, las clases cumplen con estar abiertas a adicionar/quitar funcionalidades, pero cerradas a la modificación de la clase per se. Esto se ha logrado gracias a un buen seguimiento del SRP y de la modularización de los métodos de clase. | |
| ¿Se viola el Principio de Responsabilidad Única [[1]](#footnote-1)? | En un principio sí, de forma muy evidente. La clase controller se encargaba de manejar el juego y además de grabar la partida, como ya mencioné antes se separó a la clase GameRecorder | |
| Smells código | Números mágicos |  | |
| Variable global /clase innecesaria |  | |
| Código duplicado |  | |
| Métodos largos |  | |
| Larga lista de parámetros |  | |
| Expresión demasiado compleja |  | |
| Switch o if-then-else que necesita ser reemplazado con polimorfismo |  | |
| Nombre de método o variable cuya intención no está clara |  | |
| ¿Algún método similar en otras clases? |  | |
| If anidados | A lo largo del código anidamos muchos if porque era la solución de ese momento, ahora se ha reemplazado para que en vez de hacer comprobaciones largas, si no cumple la primer condición se ejecute un return | |
| **Errores** | **Fragmento de código con errores** | **¿Cuál es el error?** | **¿Por qué es un error?** |
|  |  |  |
|  |  |  |
|  |  |  |

**4. Resumen de todo el código (1 points)**

|  |  |  |
| --- | --- | --- |
| Nombre del archivo de código fuente | Código de producción o prueba? | # lineas de código |
|  |  |  |
|  |  |  |
| Total de líneas de código | |  |

**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.

1. Revisa: [Violation solution for single responsibility principle](https://softwareengineering.stackexchange.com/questions/342051/violation-solution-for-single-responsibility-principle) [↑](#footnote-ref-1)