**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 | | |
| --- | --- | --- |
| 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 |
| ☒ Record game | Current turn: blue (or red) |  |

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 usuario quiero grabar el juego tal que pueda ver todas las jugadas realizadas | media | 1 |
| .. |  |  |  |  |
|  |  |  |  |  |

| **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 <descripción del escenario>  Given  When  Then |  |
| 20.2 | AC 20.2 <descripción del escenario>  Given  When  Then |  |
| … |  |  |
| Historia 21 | 21.1 | AC 21.1 <descripción del escenario>  Given  When  Then |  |
| … |  |  |

**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:

| **Checklist** | **Items Checklist** | **Conclusiones** | |
| --- | --- | --- | --- |
| Estándares de codificación | Convenciones de nombres | cada metodo y clase realiza lo que su nombre define | |
| Convención de ordenación de argumentos de método | se ordena correctamente los metodos de cada clase | |
| Comentarios significativos y válidos. | cada aplicacion o desarrollo de algo importante tiene un comentario entendible | |
| Estilo consistente de bloques de código | se siguieron algunos estilo que se maneja en python https://peps.python.org/pep-0008/ | |
| Indentación consistente | si | |
| … |  | |
| Principio de diseño | Clase o método no bien modularizado | check\_status/ red\_disable | |
| Visibilidad adecuada de cada variable, método y clase. | no se realiza | |
| Alguna clase con pobre abstracción | board creo que se podria manejar de otra manera los jugadores/ aunque se quiso implementar una clase player no c realizo | |
| Diseño por contrato ( pre/postcondiciones) | Se siguio cada instruccion del los sprints aunque se añadio ciertas condiciones logicas en las interfaz como validar que el tipo y el tamaño ingresado sea diferente de null lo mismo aplica para las piezas de cada jugador | |
| ¿Se viola el Principio Abierto-Cerrado? | sin la implementación de computador no | |
| ¿Se viola el Principio de Responsabilidad Única [[1]](#footnote-0)? | sin la implementacion de computador, no ya que como se menciona arriba en check status se redunda una y otra vez el dibujo de la pieza ademas debe comprobar si el tipo es computador y tambien el turno en que se encuentra | |
| Smells código | Números mágicos |  | |
| Variable global /clase innecesaria | creo que no existe una clase innecesaria | |
| Código duplicado | si en el metodo check\_status | |
| Métodos largos | computer\_move | |
| Larga lista de parámetros | no, lo maximo son 3 parametros | |
| Expresión demasiado compleja | no, se trato de realizar lo mas simple | |
| Switch o if-then-else que necesita ser reemplazado con polimorfismo | si, en el metodo check\_status se deberia reemplazar todo los if para computer | |
| Nombre de método o variable cuya intención no está clara | en check\_status la parte en la que le toca al computer sin esa parte todo estaria ok, sin embargo con esa parte pos se redunda innecesariamente | |
| ¿Algún método similar en otras clases? |  | |
| … |  | |
| **Errores** | **Fragmento de código con errores** | **¿Cuál es el error?** | **¿Por qué es un error?** |
| check\_status | No se implementa de forma correcta el movimiento que debe realizar computer ni se muestra en pantalla a pesar que si se define el metodo apra que se dibuje | El error surge al no refactorizar el codigo completo para el tipo de jugador computer, por ello no funciona correctamente al hacer cambio en check\_status. La idea era que tanto humano como computer sean jugadores sin tener que crear una clase player solo identificando el color que se elige al inicio del juego. Lo cual no se pudo realizar |
|  |  |  |
|  |  |  |

**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?

Gracias al proyecto, obtuve valiosos aprendizajes en cuanto a la modulación de clases, el establecimiento de un estilo consistente en Python, la creación de un jugador computadora, y la refactorización y encapsulación del código. Estos conocimientos adquiridos me permitirán abordar futuros proyectos de manera más eficiente, garantizando un código comprensible y limpio. Estas experiencias han sido de gran beneficio personal, ya que ahora poseo las habilidades necesarias para desarrollar proyectos de manera más efectiva y con un enfoque en la calidad del código.

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

El proyecto actual tiene fortalezas en cuanto a la eliminación de clases innecesarias. La separación de responsabilidades entre las clases "Board" y "Game" es acertada, donde "Game" utiliza "Board" y también "Computer". Sin embargo, una posible mejora sería el manejo de los jugadores. Sería beneficioso crear una clase adicional para representar a los jugadores y permitir que "Game" utilice esta clase en lugar de manejar directamente las instancias de "Computer" y otros jugadores. Esta refactorización evitaría la redundancia y facilitaría la implementación en la interfaz gráfica. Al tener una clase dedicada a los jugadores, se puede centralizar la lógica relacionada con ellos y simplificar el proceso de agregar, eliminar o cambiar jugadores en el juego. En resumen, el proyecto actual muestra eficiencia al evitar clases innecesarias, pero puede mejorar mediante la introducción de una clase específica para los jugadores y así optimizar el manejo de estos en el juego.

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

Lo que mejoraría el proceso de desarrollo al crear un juego similar desde cero. En primer lugar, asignaría la misma prioridad a cada historia de usuario, lo que ayudará a mantener un enfoque equilibrado en todas las funcionalidades del juego. Además, es crucial priorizar la refactorización del código, ya que esto garantizará un diseño más limpio y mantenible a medida que el proyecto avance. Asimismo, es recomendable utilizar herramientas de control de versiones, realizar pruebas exhaustivas y buscar retroalimentación de los usuarios para mejorar continuamente el juego. Ya con este proyecto como experiencia también adoptaría otros estilos que existen en python. Finalmente, reconozco la necesidad de enfocarme en la realización de pruebas unitarias. Aunque he implementado pruebas para las clases existentes en este proyecto, considero que puedo fortalecer mis habilidades en esta área mediante una mayor lectura de teoría y una mejor definición de los dominios de prueba. Al hacerlo, podré asegurarme de que cada componente del juego funcione correctamente y aumentar la confiabilidad y robustez del código en general.s

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-0)