**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)** |
| 14 | Grabar un juego | Como usuario quiero guardar en un archivo de texto todos los movimientos realizados durante un juego. | “debería tener” | 4 |
| 15 | Reproducir un juego | Como usuario quiero poder reproducir en el tablero el juego guardado | “podría tener” | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ID y nombre de la historia de usuario** | **AC**  **ID** | **Descripción del criterio de aceptación** | **Estado (completado, por hacer, en progreso)** |
| 14. Grabar un juego | 14.1 | AC 14.1 Guardado de un juego en un archivo de texto  Dado que aún no comienza el juego y se ha seleccionado la opción para grabar el juego  Cuando se inicie el juego  Entonces cada jugada será guardad en un archivo de texto | Completado |
| 15. Repoducir un juego | 15.1 | AC 15.1 Reproducciñon de un juego guardado  Dado que aún no comienza el juego y existiendo un juego guardado en un archivo de texto  Cuando se presione el botón reproducir  Entonces se ejecutarán los movimientos del juego guardado y se mostrarán en el tablero | Completado |

**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: JuegoSimple

Fecha/hora de duración del ejercicio de revisión del código: 03/06/2023

**Juego Simple**

|  |  |  |  |
| --- | --- | --- | --- |
| **Checklist** | **Items Checklist** | **Conclusiones** | |
| Estándares de codificación | Convenciones de nombres | Se utilizó la guia de estilo de Java de Google (<https://google.github.io/styleguide/javaguide.html>) . Por lo cual los nombres de las clases usan UpperCamelCase, por ejemplo la clase JuegoSimple. Los nombres de los métodos, campos no constantes, parámetros y variables locales usan lowerCamelCase, por ejemplo el método guardarJuego. Las constantes usan UPPER\_CAMEL\_CASE.  Además se intenta que los nombres sean significativos para dar una idea de que representan, por ejemplo el método getNumeroCeldasVacias retorna el número de celdas vacías en el tablero en el momento actual. | |
| Convención de ordenación de argumentos de método | No se siguió alguna convención para el ordemaniento de los argumentos de método. | |
| Comentarios significativos y válidos. | Se añadieron comentarios a los campos para aclarar a que representan, por ejemplo:  private int totalFilas; // número de filas del tablero  Se añadieron comentarios a los métodos para aclarar que hacen, sus entradas y salidas. Por ejemplo:  */\*\**  *\* Añade la jugada actual a la cadena de texto juegoGuardado*  *\* @param fila fila de la jugada*  *\* @param columna columna de la jugada*  *\* @param valorCelda valor de la celda de la jugada*  *\*/* public void guardarJugada(int fila, int columna, Celda valorCelda) { | |
| Estilo consistente de bloques de código | Se siguió la guia de estilo de Java de Google, la cual indica que para bloques no vacíos se use el estilo de Kernighan y Ritchie. Esto se implementó de manera automática la opción del IDE IntellijIdea de aplicar un estilo de código desde un archivo XML. | |
| Indentación consistente | Se siguió la guia de estilo de Java de Google, la cual indica que para indentación se usen dos espacios. Esto se implementó de manera automática la opción del IDE IntellijIdea de aplicar un estilo de código desde un archivo XML. | |
| Principio de diseño | Clase o método no bien modularizado | Se observó que tanto en la clase JuegoSimple como JuegoGeneral el método setTurno:  public void setTurno(Turno turno) {  this.turno = turno; }  solo se llama dentro de la expresión:  setTurno((getTurno() == Turno.*ROJO*) ? Turno.*AZUL* : Turno.*ROJO*);  por lo que se decidió eliminar el método setTurno y crear el método cambiarTurno:  public void cambiarTurno() {  turno = (getTurno() == Turno.*ROJO*) ? Turno.*AZUL* : Turno.*ROJO*; }  y llamarlo desde donde antes se usaba la expresión anterior. | |
| Visibilidad adecuada de cada variable, método y clase. | Se cambió la visibilidad del campo juegoDebeGuardarse de protected a private y se creó el método isJuegodebeGuardarse para acceder a su valor. De esta manera todos los campos quedan privados y los métodos públicos. | |
| Alguna clase con pobre abstracción | No se observa que la clase JuegoSimple tenga probre abstracción, ya que todos los métodos están relacionados con el realizar un juego simple. | |
| Diseño por contrato ( pre/postcondiciones) | Las precondiciones se verifican dentro de cada método, por ejemplo en las clase getCelda, que tiene los parámetros fila y columna, se verifica los valores de fila y columna dentro del método y retorna null en caso la fila o columna no esté dentro del rango adecuado:  public Celda getCelda(int fila, int columna) {  if (fila >= 0 && fila < totalFilas && columna >= 0 && columna < totalColumnas) {  return tablero[fila][columna];  } else {  return null;  } } | |
| ¿Se viola el Principio Abierto-Cerrado? | No se observó violaciones al Principio Abierto-Cerrado ya que por ejemplo si se quiere modificar el tamaño del tablero del juego, no es necesario modificar alguna linea de código que defina el tamaño del tablero sino que se debe usar el método setTablero(tamanio). | |
| ¿Se viola el Principio de Responsabilidad Única [[1]](#footnote-2)? | Se observa que la clase JuegoSimple viola el principio de responsabilidad única, ya que algunos de sus métodos como guardarJuego, guardarJugada deberían estar en una clase aparte que maneje el guardado de juegos. Por esto se creará la clase Guardado que sea responsable de estos métodos. | |
| Smells código | Números mágicos | Se observaron números mágicos en el código, por lo que se crearon constantes para mejorar la legibilidad del código:  private final int TAMANIO\_MINIMO\_TABLERO = 2;  private final int TAMANIO\_MAXIMO\_TABLERO = 20; | |
| Variable global /clase innecesaria | En la clase JuegoSimple no se han creado variables globales. | |
| Código duplicado | Se observa que el código del método esEmpate es similar al del método getNumeroCeldasVacias y que esEmpate solo se llama una vez. Por lo que para no repetir código se decide eliminar el método es empate y reemplazar su llamado por la expresion getNumeroCeldasVacias() == 0 | |
| Métodos largos | El método hizoSos es demasiado largo por lo que se crean dos métodos hizoSosConS e hizoSosConO que dividen la funcionalidad de hizoSos. | |
| 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? |  | |
| … |  | |
| **Errores** | **Fragmento de código con errores** | **¿Cuál es el error?** | **¿Por qué es un error?** |
|  |  |  |
|  |  |  |
|  |  |  |

**Clase**

|  |  |  |  |
| --- | --- | --- | --- |
| **Checklist** | **Items Checklist** | **Conclusiones** | |
| Estándares de codificación | Convenciones de nombres |  | |
| Convención de ordenación de argumentos de método |  | |
| Comentarios significativos y válidos. |  | |
| Estilo consistente de bloques de código |  | |
| Indentación consistente |  | |
| … |  | |
| Principio de diseño | Clase o método no bien modularizado |  | |
| Visibilidad adecuada de cada variable, método y clase. |  | |
| Alguna clase con pobre abstracción |  | |
| Diseño por contrato ( pre/postcondiciones) |  | |
| ¿Se viola el Principio Abierto-Cerrado? |  | |
| ¿Se viola el Principio de Responsabilidad Única [[2]](#footnote-3)? |  | |
| 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? |  | |
| … |  | |
| **Errores** | **Fragmento de código con errores** | **¿Cuál es el error?** | **¿Por qué es un error?** |
|  |  |  |
|  |  |  |
|  |  |  |

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

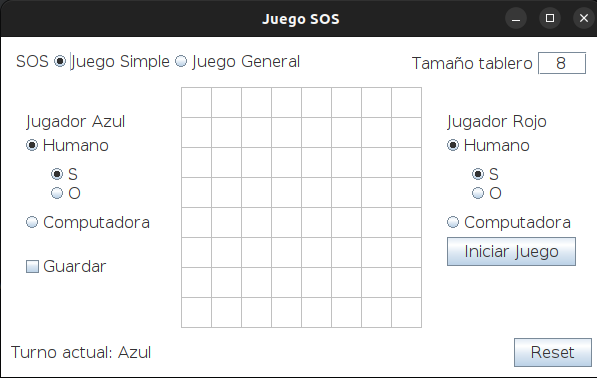
|  |  |  |
| --- | --- | --- |
| Nombre del archivo de código fuente | Código de producción o prueba? | # lineas de código |
|  |  |  |
|  |  |  |
| Total de líneas de código | |  |

**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-2)
2. Revisa: [Violation solution for single responsibility principle](https://softwareengineering.stackexchange.com/questions/342051/violation-solution-for-single-responsibility-principle) [↑](#footnote-ref-3)