UTN
Facultad Regional Córdoba

TUP

Tecnicatura Universitaria en Programación

LABORATORIO DE COMPUTACIÓN III

Trabajo Práctico Integrador
Chess (Ajedres)

TPI
2° Año – 3° Cuatrimestre

Trabajo Práctico Integrador 2

Trabajo Práctico Integrador

Introducción

El ajedrez es un juego de estrategia milenario que ha cautivado a personas de todas las edades y culturas a lo largo de la historia. Con su tablero de 64 casillas y sus piezas únicas, el ajedrez desafía la mente y pone a prueba las habilidades cognitivas, la planificación, el razonamiento lógico y la toma de decisiones. Además, el ajedrez es una disciplina que ha evolucionado a lo largo de los años, adaptándose a la tecnología y siendo un campo propicio para aplicar los conocimientos de la programación.

En este contexto, como parte de la materia Laboratorio de Computación III, se presenta el desafío de desarrollar un juego de ajedrez funcional en Java. Este desafío no solo requiere conocimientos técnicos en programación, sino también una comprensión profunda de las reglas y la lógica del ajedrez, así como la capacidad de aplicar conceptos aprendidos en las clases para crear un juego interactivo y totalmente funcional.

Imagen 1: extraída de internet.

Objetivos

El objetivo principal de este trabajo integrador es aplicar los conocimientos adquiridos durante las clases de Laboratorio de Computación III para desarrollar un juego de ajedrez completamente funcional en Java. Esto implica diseñar e implementar un programa que permita a dos jugadores competir entre sí en una partida de ajedrez, siguiendo todas las reglas del juego y permitiendo las funcionalidades esenciales de un juego de ajedrez, como el movimiento de las piezas, la validación de los movimientos legales, la detección de jaque y jaque mate, la gestión del tablero y la interacción con los jugadores.

El juego de ajedrez desarrollado también debe incluir características avanzadas que demuestren la comprensión y aplicación de los conceptos de programación aprendidos, como la implementación de patrones de diseño, la gestión eficiente de la memoria, la modularidad y la reutilización de código, la manipulación de estructuras de datos complejas, la implementación de algoritmos de búsqueda y evaluación de jugadas. También deben aplicarse los conocimientos adquiridos sobre la gestión del código fuente, su versionado y manipulación.

El desarrollo del juego de ajedrez en Java requerirá la aplicación de conocimientos de programación orientada a objetos, manejo de estructuras de datos, algoritmos de ajedrez, manejo de interfaces gráficas de usuario, pruebas unitarias, y el uso de herramientas como Maven, JUnit, Mockito y JaCoCo, entre otros.

Este proyecto no solo permitirá a los estudiantes aplicar los conceptos y técnicas aprendidas en el aula, sino que también los desafiará a enfrentarse a situaciones reales de desarrollo de software, como la planificación, diseño, implementación, pruebas y entrega de una aplicación de software completa y funcional. Además, el uso de herramientas de control de versiones y la colaboración en un entorno de desarrollo colaborativo como GitHub les brindará una experiencia valiosa en el uso de herramientas profesionales ampliamente utilizadas en la industria.

Opcionalmente, los alumnos podrán desarrollar (en un proyecto aparte) una interfaz gráfica que se integre al proyecto de BackEnd mediante apis Rest y que sea visualmente atractivo y fácil de usar, que permita a los jugadores interactuar de manera intuitiva con el tablero y las piezas. Dicha interfaz debe ser diseñada de manera profesional, con una disposición clara del tablero, una representación gráfica adecuada de las piezas y una respuesta rápida y precisa a las interacciones del usuario.

Requerimientos Funcionales

La siguiente lista de requerimientos debe cumplirse para que el trabajo se considere completo y correcto, **a excepción de los requerimientos opcionales, que serán considerados como parte de la nota normativa**.

1. Reglas del juego:
    a. Implementación de las reglas oficiales del ajedrez, incluyendo el movimiento de las piezas (peón, torre, caballo, alfil, dama, rey) de acuerdo a las reglas de movimiento específicas de cada pieza.
    b. Validación de movimientos legales basada en las reglas del ajedrez, incluyendo la verificación de los movimientos de las piezas, la detección de movimientos ilegales (como movimientos que dejen el propio rey en jaque)
    c. **Opcional**: la implementación de reglas especiales como el enroque, el peón al paso y la promoción del peón.
2. Gestión del juego:
    a. Gestión de la partida de ajedrez, incluyendo el inicio de una nueva partida, la gestión de los turnos de los jugadores, la detección de jaque y jaque mate, y la finalización de la partida.
    b. Gestión de los diferentes estados del juego, como el estado de juego en curso, el estado de jaque, el estado de jaque mate y la finalización del juego con la victoria de uno de los jugadores.
    c. El juego debe mantener un historial de movimientos realizados en cada partida.
    d. **Opcional**: El estado de empate (por ejemplo, tablas por repetición o ahogado), y la finalización del juego con la victoria de uno de los jugadores.
    e. **Opcional**: Gestión del historial de movimientos, permitiendo a los jugadores revisar los movimientos anteriores y retroceder en el historial.
3. Gestión de jugadores y partidas:
    a. Gestión de múltiples jugadores, permitiendo que dos jugadores humanos compitan entre sí en la misma computadora.
    b. **Opcional**: Gestión de múltiples partidas, permitiendo que varios juegos de ajedrez se puedan llevar a cabo de manera concurrente o consecutiva.

4. Guardado y carga de partidas:
    a. El juego debe permitir a los jugadores guardar y cargar partidas en curso para poder reanudarlas en otro momento. Esto implica implementar un sistema de guardado y carga de partidas que sea confiable y seguro, y que mantenga la integridad de los datos del juego.
5. **Opcional**: Gestión de tiempos:
    a. **Opcional**: El juego debe implementar la gestión del tiempo de juego, incluyendo la posibilidad de establecer límites de tiempo por jugador, llevar un registro del tiempo restante en cada turno y realizar acciones adecuadas cuando se agote el tiempo, como declarar la victoria del jugador que tenga tiempo restante.
6. Otras funcionalidades:
    a. Implementación de características avanzadas, como la notación algebraica, que permite a los jugadores registrar los movimientos en un formato estándar.
    b. Opcional: El juego debe permitir utilizar notaciones de ajedrez para reproducir partidas. Esto implica implementar un sistema de conversión y validación de notaciones de ajedrez, así como permitir la visualización de partidas en notaciones de ajedrez.
    c. **Opcional**: Gestión de temporizadores, permitiendo el uso de relojes de ajedrez para controlar el tiempo de juego de los jugadores.
    d. **Opcional**: El juego puede ser diseñado para ser multilingüe (Ingles y Español), permitiendo a los jugadores seleccionar diferentes idiomas para la interfaz de usuario y los mensajes del juego.

Requerimientos No Funcionales

La siguiente lista de requerimientos debe cumplirse para que el trabajo se considere completo y correcto, **a excepción de los requerimientos opcionales, que serán considerados como parte de la nota normativa**.

1. Rendimiento y optimización:
    a. El juego debe tener un rendimiento óptimo, con una respuesta rápida a las interacciones del usuario, incluso en condiciones de carga de trabajo intensa.

   b. Se debe realizar una optimización adecuada del código y del uso de recursos, considerando la eficiencia en la ejecución del juego y la gestión de memoria.

2. Calidad del código y buenas prácticas de programación:
   a. El código debe seguir las convenciones de codificación de **Java (en versión 11)** y mantener una alta calidad en términos de legibilidad, mantenibilidad y escalabilidad.
   b. Se debe aplicar el principio de responsabilidad única (Single Responsibility Principle) y otros principios de diseño **SOLID** para garantizar una arquitectura de software robusta y modular.
   c. Se debe utilizar **Maven** como herramienta de gestión de dependencias y construcción del proyecto, siguiendo las mejores prácticas de estructura de proyectos y gestión de versiones.
3. Pruebas unitarias y cobertura de código:
   a. Se deben implementar pruebas unitarias utilizando **JUnit** y **Mockito** para garantizar la calidad del código y su correcto funcionamiento.
   b. Se debe lograr una cobertura de código con **JaCoCo** de al menos **80%**, asegurando que las pruebas cubran la mayoría de las funcionalidades del juego.
4. Usabilidad y accesibilidad:
   a. La interfaz de usuario debe ser intuitiva y fácil de usar, con una experiencia de usuario agradable.
   b. **Opcional**: En caso de hacer una interfaz web, se deben considerar las mejores prácticas de desarrollo web.
5. Diseño de software y patrones de diseño:
   a. Se debe aplicar un enfoque de diseño de software modular y bien estructurado, utilizando patrones de diseño apropiados para mejorar la calidad y mantenibilidad del código.

      Ejemplos de patrones de diseño que podrían ser aplicados en el desarrollo del juego de ajedrez incluyen el patrón de diseño de Observador para gestionar eventos de usuario y el patrón de diseño de Estado para gestionar los diferentes estados del juego, como el estado de juego en curso, el estado de jaque, y el estado de jaque mate.

6. Seguridad:
    b. Se debe evitar el uso de librerías (dependencias) que poseen vulnerabilidades críticas/high conocidas.
    c. **Opcional**: Se deben implementar medidas de seguridad adecuadas para proteger la integridad del juego y la confidencialidad de los datos del usuario, si los hubiera.
    d. **Opcional**: Se deben evitar vulnerabilidades conocidas, como inyección de código, desbordamiento de búfer y otros ataques comunes.

- **NOTA: Es importante que los alumnos consideren estos requerimientos no funcionales como parte integral del desarrollo del juego de ajedrez en Java, para garantizar un software de alta calidad, eficiente y con características profesionales.**

Forma de Trabajo

Para este proyecto, **los alumnos trabajarán en grupos de 4 o 5 personas**, lo que fomentará el trabajo en equipo y la colaboración entre los miembros del grupo. Cada equipo será responsable de autogestionarse y organizarse de manera eficiente para llevar a cabo todas las etapas del desarrollo del juego, desde el diseño y la implementación hasta las pruebas y la entrega final.

Una parte fundamental de este proyecto será **el uso de herramientas de control de versiones, específicamente Git y GitHub, usando el workflow "GitFlow"** aprendido en clases. Cada equipo deberá utilizar Git como sistema de control de versiones para mantener un registro de los cambios realizados en el código fuente del juego y trabajar de manera colaborativa en un repositorio de GitHub. Además, **se creará un aula virtual (classroom) en GitHub específica para este proyecto**, donde los equipos podrán acceder a los recursos necesarios, realizar seguimiento del progreso del proyecto y entregar sus entregables.

El trabajo en equipo y la colaboración serán elementos clave en el desarrollo del proyecto. Los equipos deberán dividirse las tareas de manera equitativa, asignando responsabilidades a cada miembro en función de sus habilidades y conocimientos. Además, deberán comunicarse de manera efectiva para coordinar y sincronizar sus esfuerzos, asegurándose de que todos los miembros del equipo estén alineados y contribuyan activamente al progreso del proyecto.

Forma y Fechas de Entrega

La entrega del trabajo se realizará a través del aula virtual (classroom) específica para este proyecto en GitHub. Se establecerá una fecha límite de entrega, la cual deberá ser respetada por todos los equipos. Es importante tener en cuenta que todos los requerimientos funcionales y no funcionales obligatorios deberán estar cumplidos en la fecha de entrega para poder aprobar el proyecto. Cualquier entrega realizada después de la fecha límite será considerada como tardía y estará sujeta a penalizaciones en la evaluación final del proyecto.

La fecha de entrega será fijada hasta el día sábado 17 de junio del 2023 a las 23:00 Hs.

Evaluación y Calificación:

Es importante destacar que cumplir con los requerimientos establecidos es un criterio mínimo para aprobar el proyecto, pero para obtener una evaluación sobresaliente se valorará también la calidad del código, la eficiencia del diseño y la implementación, la cobertura de pruebas, la claridad de la documentación y la capacidad de trabajo en equipo y colaboración demostrada durante el desarrollo del proyecto.

- Se evaluará la calidad, complejidad y presentación del trabajo como así también la oportunidad de entrega en fecha.
- Este práctico tiene carácter de obligatorio y deberá aprobarse con un 60% (al igual que los parciales) para regularizar la materia.
- La calificación alcanzada por el grupo de trabajo será la calificación individual de cada uno de los integrantes del equipo.