TP1: PowerChess

Detalles del trabajo

Objetivos

- Analizar el problema planteado correctamente.
- Diseñar un modelo general de solución utilizando correctamente los conceptos de programación orientada a objetos.
- Demostrar el conocimiento de principios de diseño y de las buenas prácticas de programación.
- Desarrollar una interfaz de uso agradable para un potencial usuario.

Indicaciones del trabajo

Grupos de trabajo

El trabajo se desarrollará de forma grupal, en grupos de 4 alumnos. El trabajo práctico debe realizarse utilizando Git y Github como herramienta de colaboración.
Todos los alumnos integrantes del grupo **deben** participar activamente del desarrollo del trabajo de forma equitativa. Mientras la participación sea equitativa, queda a criterio de cada grupo la forma de trabajo y organización.

Tecnología

El trabajo debe realizarse en Java, utilizando cualquier versión LTS de Java 17 en adelante.
El proyecto debe ser un proyecto de Maven, con soporte para JavaFX. (recomendación: seguir instrucciones de creación de proyectos JavaFX con Maven, o bien las que les sugiera su IDE en caso de ofrecerles la opción).

Tiempo de trabajo y entregas parciales

Se contarán con 5 semanas completas para llevar a cabo el desarrollo del TP.
A la semana 2, el trabajo contará con un **checkpoint**, que consiste en una entrega parcial no obligatoria, para evaluar junto a su corrector asignado el estado de su entrega hasta el momento, y el diseño considerado hasta el momento, de forma que pueda hacerse una primera ronda de correcciones estructurales.
Finalmente, transcurrido el tiempo de trabajo, debe realizarse la **entrega definitiva**.

Sugerencia de distribución de carga

La distribución de carga sugerida por el curso para llevar a cabo el trabajo es contemplando los siguientes hitos semanales:

- Semana 0:
    - Conformación definitiva del grupo (consolidada en el formulario de inscripción de grupos).
    - Creación del repositorio de Github (privado).
    - Invitación al repositorio de Github a su corrector asignado.
- Semana 1:
    - Proyecto de JavaFX + Maven creado y corriendo.
    - Bosquejo preliminar del modelo, dadas las características del dominio, utilizando diagramas de clase, y de secuencia donde sea necesario.
    - Avances sobre implementación de lógica menos acoplada al modelo:
        - Por ejemplo, representación de un tablero y movimiento de las piezas.
- Semana 2 - **CHECKPOINT**:
    - Diagrama de clases contemplando Modelo, Vista y Controlador, cumpliendo la correcta separación entre las partes (listo para ser evaluado por el corrector).
    - Parte del modelo volcado en código.
    - Onboarding a JavaFX realizado y pantalla de UI base terminada.
        - Esto es, una primera pantalla interactiva que pueda dibujar el escenario básico del problema e interpretar eventos del usuario, sin conectar con el Modelo aún.
- Semana 3:
    - Implementar correcciones dadas por el corrector.
    - Desarrollo del Controlador.
    - Continuación del desarrollo del Modelo.
    - Continuación del desarrollo de la Vista.
    - (Opcional) Desarrollo de tests.
- Semana 4:
    - Integración parcial de las 3 componentes del programa.
    - Pruebas manuales base y corrección de los casos de error encontrados.
    - Comenzar a escribir el informe.
- Semana 5:
    - Terminar de integrar el programa completo.
    - Pruebas finales y completas.
    - Arreglos finales y puesta a punto definitiva.
    - Finalización del informe.
    - Efectuar la entrega.

Modalidad de entrega

Para realizar la entrega se debe:

- Github:
    - El repositorio tiene que ser privado y debe estar su corrector invitado al mismo.
    - Para realizar la entrega se debe crear un branch llamado `tp-1`, con el código correspondiente a la entrega. No se debe modificar más luego de la fecha de entrega.

    - Se debe contar con un `README.md` que explique cómo correr el programa “desde cero”, cómo instalar las dependencias y cómo correr el juego de forma clara y única. El código debe poder correrse con un comando que compile el código y lo ejecute, **no es válido subir un ejecutable y que el comando simplemente lo corra**.
- Mail: Mandar un mail a algoritmos3.fiuba@gmail.com indicando número de grupo y sus integrantes, aclarando cuál es la branch y el repositorio de la entrega. El mail debe llevar adjunto el informe en formato PDF.

Fecha de entrega

La entrega parcial opcional implica subir la última versión del código al repositorio, y notificar al corrector asignado. Esta entrega se espera aproximadamente el 20/09/2024.

La entrega definitiva, que debe alinearse a lo especificado en la Modalidad de entrega, debe realizarse con anterioridad a las 23:59hs del 10/10/2024.

Enunciado: Desarrollando el PowerChess

Dominio

El juego se encuentra basado en el clásico Ajedrez. En primera instancia, se debe poder jugar como un Ajedrez normal con todos los movimientos del Rey, Dama, Torre, Alfil, Caballo y Peón. Adicionalmente, se debe permitir:

- Coronar un Peón
- Realizar un enroque
- Que el primer turno sea siempre del jugador con las piezas blancas
- No permitir movimientos ilegales, ya sea por movimientos ajenos a la pieza o que conlleven a finalizar el turno con el Rey del jugador en jaque.
- Detectar cuando la partida se encuentra en tablas

Sobre la base del clásico juego, se quiere implementar un cambió en las reglas, con la utilización de **poderes** (o power-ups, si se quiere), que cada jugador puede decidir utilizar durante la partida. Ningún poder se puede aplicar sobre el Rey de cada jugador.
Los poderes que se contemplan agregar, junto a su categorización, son:

- De duración (duran X turnos):
    - **Freeze**: se aplica sobre una pieza rival, imposibilitando que el oponente pueda mover dicha pieza por X turnos.
    - **Escudo**: se aplica sobre una pieza propia, haciendo que no pueda ser “comida” por el oponente, durante X cantidad de sus turnos.
- De acción (aplican a la acción realizada ese turno):
    - **Vuelo**: permite a la pieza que elija moverse en ese turno, pasar por encima de otras que bloqueen su movimiento (tanto amigas como enemigas), tal como hace el caballo.
    - **Doble-juego**: permite a la pieza elegida mover 2 veces durante el mismo turno.
    - **Robar poder**: permite robar un poder disponible del jugador rival.
    - **Limpieza**: permite anular algún poder que posea alguna pieza, amiga o enemiga.
- De evolución:
    - Permite agregar a una pieza movimientos extra por el resto de la partida (ej: poder moverse 1 casilla al costado, o agregarle la diagonal). Solamente puede aplicarse sobre alfiles, caballos o torres.
        - Podrían existir varias versiones de este poder, agregando diferentes movimientos cada uno.

Requerimientos

Importante: En caso de tomar una decisión respecto a una definición ambigua del dominio o los requerimientos, volcarlo en la correspondiente sección de hipótesis del informe.

Modelo

En un proyecto Java, se debe volcar la representación del modelo del juego, con todas sus características enunciadas en el Dominio, contemplando tanto la lógica básica como sus *corner-cases*.

Jugabilidad

Como se indicó previamente, el juego contempla una partida entre 2 jugadores.
Al comenzar la partida, se deben poder asignar los nombres a los 2 jugadores, y se debe poder elegir quién arranca con cada color.

En cada turno, el jugador debe mover una de sus piezas. Solamente podrá realizar movimientos legales, dado el contexto de la partida (piezas en el tablero, si hay mates de por medio, los power-ups de las piezas, etc). El jugador elegirá la pieza que desea mover y la casilla a donde desea moverla. El usuario debe poder de-seleccionar una pieza seleccionada. Una vez movida una pieza, finalizará el turno de dicho jugador, y comenzará el turno del otro jugador.
En cada turno, el jugador puede aplicar uno de sus power-ups restantes a alguna de las piezas que lo pueda recibir.

En cualquiera de sus turnos, un jugador debe poder ofrecer tablas al oponente, el cual este puede aceptar o no.
En cualquier momento de la partida, un jugador puede rendirse, dando por victorioso a su oponente.

Al finalizar la partida, se debe indicar el resultado (victoria para blancas u oscuras, o tablas).

Interfaz gráfica

El juego debe poder ser jugado mediante una interfaz gráfica.

General:

- Dicha interfaz debe mostrar en todo momento el tablero, con las piezas presentes en el tablero con su respectivo identificador (idealmente una imagen de la pieza).
- La interfaz debe indicar claramente de qué jugador es el turno actual en cada turno.
- Se deben poder ver los detalles de cada jugador (nombre, color de piezas y power-ups que le quedan).
- Debe poder consultarse el estado actual de la pieza, en caso de que esta posea un power-up.

En cada turno:

- Al seleccionar una pieza, la interfaz debe indicar a qué casillas se puede mover dicha pieza (ya sea pintando las casillas, agregándoles una marca, o de cualquier otra forma que quede claro).
- Se debe poder seleccionar un power-up para aplicarlo. Idealmente, que se muestren únicamente las piezas que pueden recibir dicho power-up.

    - Luego de aplicar un power-up en un turno, no se debe poder dar la posibilidad de aplicar otro.
- Debe aparecer un botón para ofrecer tablas al rival, a lo que este debe poder confirmar o no el ofrecimiento.
- Debe aparecer un botón para que el jugador actual se rinda.

Al finalizar la partida:

- En caso de victoria, se debe indicar claramente el nombre del ganador.
- En caso de empate, se debe mostrar un mensaje acorde.

Informe

Se espera la realización de un informe que contenga:

- Un diagrama de clases UML **completo**, dando las explicaciones que consideren pertinentes respecto a la toma de decisiones en el diseño.
- Al menos un diagrama de secuencia, de algún flujo que consideren interesante. Explicar por qué eligieron mostrar ese flujo.
- Una sección de las hipótesis tomadas durante la realización del trabajo.
- Conclusiones.

Criterios de aprobación

Para que el trabajo práctico se considere aprobado, deberá cumplir con los siguientes requisitos mínimos:

- Cumplir con los objetivos enunciados en los detalles del trabajo, como así también cumplir con los lineamientos y buenas prácticas del POO.
- Implementación de un ajedrez básico, entre 2 jugadores anónimos y con asignación de colores aleatoria (es decir, no es necesario contemplar el nombramiento y la elección de color al iniciar la partida).
- Implementación de al menos 2 de los power-ups.
- El juego no debe permitir realizar movimientos ilegales bajo ningún concepto; solamente movimientos permitidos dado el contexto y las piezas presentes en el tablero en cada turno.
- Interfaz minimal que permita llevar a cabo una partida indeterminadamente:
    - No es necesario contemplar la finalización o el empate
    - No es necesario utilizar imágenes para mostrar las piezas o el todo estado actual de la partida
    - Sí debe indicar de quién es el turno actual, y las piezas presentes en el tablero en cada momento.

Si bien estos son los requisitos mínimos para la aprobación, un TP se considerará *completo* si cumple con todos los requerimientos descritos en las secciones anteriores. Un TP incompleto, puede no ser considerado para la promoción.