Funcionalidad de guardado en PGN:

1. La clase GestorTimepoRealPgn tiene varias variables de instancia para realizar un seguimiento del estado del juego, incluida la ruta del archivo del juego, la lista de movimientos, los encabezados (metadatos) del juego, el número de movimiento actual y el resultado del juego.
2. El método inicializarEncabezado configura los encabezados iniciales del archivo PGN, como el nombre del evento, el sitio, la fecha, los jugadores, las clasificaciones Elo y el resultado (que inicialmente se establece en "\*" para un juego en curso).
3. Se llama al método grabarMovimiento cada vez que se realiza un movimiento en el juego. Construye la notación de movimiento en el formato PGN (por ejemplo, "1. Cf3", "2. C4 E6", etc.) y lo agrega a la lista de movimientos.
4. El método guardaJuego es responsable de escribir el archivo PGN. Primero escribe los encabezados en el orden PGN estándar, luego escribe los movimientos y, finalmente, agrega el resultado del juego si el juego ha terminado.
5. El método guardarJuego permite al usuario guardar el juego en una nueva ubicación de archivo. Primero guarda el estado actual en el archivo original, luego copia el archivo en la nueva ubicación y actualiza la variable rutaArchivo en consecuencia.
6. El método añadirUltimoMovimiento se usa para anexar una notación adicional (por ejemplo, para una promoción de peón) a la última jugada de la lista de jugadas y, a continuación, guarda la partida actualizada.
7. Se llama al método finalizarJuego cuando finaliza el juego. Actualiza el resultadoJuego y el encabezado correspondiente y, a continuación, guarda el juego.

Esta implementación realiza un seguimiento del estado del juego, actualiza el archivo PGN con cada movimiento y proporciona la capacidad de guardar el juego en una nueva ubicación o agregar información adicional al último movimiento. El diseño separa las preocupaciones del manejo del juego (en la clase ModeloTableroAjedrez) y la administración de archivos PGN (en la claseGestorTiempoRealPGN), lo que ayuda a mantener la organización y la capacidad de mantenimiento del código.

Implementación de las reglas:

La clase ValidadorMovimientoAjedrez es responsable de validar la legalidad de las jugadas de ajedrez en función de las reglas del juego. Este es un resumen de cómo se implementa la validación de movimiento:

1. El método MovimientoValido es el punto de entrada principal para comprobar la validez de un movimiento propuesto. Primero comprueba si hay una pieza en la posición "desde", si el color de la pieza coincide con el turno del jugador actual y si la jugada no captura la propia pieza del jugador.
2. A continuación, el método delega la validación del tipo de pieza específica a métodos auxiliares independientes:
   * MovimientoValidoPeon
   * MovimientoValidoTorre
   * MovimientoValidoCaballo
   * MovimientoValidoAlfil
   * MovimientoValidoAlfil
   * MoviminetoValidoAlfil

Estos métodos implementan las reglas para cada tipo de pieza, como los movimientos de avance y captura del peón, los movimientos lineales de la torre, los movimientos en forma de L del caballo, los movimientos diagonales del alfil, los movimientos combinados de torre y alfil de la dama y los movimientos de una casilla del rey.

1. El método caminoBloqueado se usa para comprobar si el camino entre las posiciones "desde" y "hasta" está bloqueado por otras piezas, lo cual es necesario para el movimiento de ciertas piezas (torre, alfil y reina).
2. El método enJake se utiliza para determinar si el rey del jugador está siendo atacado por las piezas del oponente. Esto es importante para comprobar si una jugada dejaría en jaque al rey del jugador.
3. El método deberiaPrevenirJak ese utiliza para comprobar si una jugada propuesta evitaría que el rey del jugador estuviera en jaque. Esto es necesario para asegurarse de que un jugador no pueda hacer un movimiento que deje a su rey en jaque.
4. El método esJakeMate se utiliza para determinar si el jugador está en una situación de jaque mate, donde el rey está en jaque y no hay movimientos válidos que puedan sacar al rey del jaque.
5. El método esEmpatese utiliza para determinar si el jugador se encuentra en una situación de punto muerto, en la que el jugador no tiene movimientos válidos pero el rey no está en jaque.

En general, la clase ValidadorMovimientoAjedrez encapsula la lógica para validar la legalidad de las jugadas de ajedrez de acuerdo con las reglas estándar del juego. Proporciona una clara separación de preocupaciones entre la gestión del tablero de juego y la validación del movimiento, lo que hace que el código sea más modular y fácil de mantener.

Aplicación de algunos principios POO

1. ControladorAjedrez: esta clase sirve como controlador principal para la aplicación de ajedrez. Controla la entrada del usuario, administra el estado del juego y coordina entre el modelo del tablero de ajedrez y la vista del tablero de ajedrez.
2. BasicPiezaAjedrez: Esta clase representa una pieza de ajedrez básica, encapsulando su símbolo, color y tipo.

La clase ControladorAjedrez es responsable de las siguientes funcionalidades clave:

* Inicializar el modelo y la vista del tablero de ajedrez, y configurar el estado inicial del juego.
* Controlar los clics del ratón para seleccionar y mover piezas de ajedrez, validando los movimientos mediante la clase ValidadorMovimientoAjedrez.
* Grabación de movimientos en el formato PGN utilizando la clase GestorTiempoRealPGN.
* Comprobar las condiciones de jaque mate y empate, y actualizar el estado del juego en consecuencia.
* Proporcionar métodos para colocar piezas de ajedrez en el tablero utilizando notación algebraica.
* Manejo de la promoción de peones cuando un peón llega al extremo opuesto del tablero.
* Implementación de una función de "guardar partida" que permite al usuario guardar el estado actual del juego como un archivo PGN.
* Proporciona una función de "reinicio del juego" para despejar el tablero y establecer la posición inicial.

La clase BasicPiezaAjedrez representa una sola pieza de ajedrez, con propiedades para su símbolo, color y tipo. Esta clase es utilizada por el ControladorAjedrez para administrar las piezas en el tablero de ajedrez.

En general, el archivo ControladorAjedrez.java define la lógica principal y el flujo de la aplicación de ajedrez, conectando el modelo, la vista y las interacciones del usuario para proporcionar una implementación funcional del juego de ajedrez.

El archivo ControladorAjedrez.java muestra los siguientes principios de programación orientada a objetos (POO):

1. **Encapsulación**: la clase ControladorAjedrez encapsula el modelo de tablero de ajedrez, la vista del tablero de ajedrez y el validador de movimientos, ocultando los detalles de implementación interna del mundo exterior.
2. **Abstracción**: La clase BasicPiezaAjedrez proporciona una representación abstracta de una pieza de ajedrez, ocultando la complejidad de los diferentes tipos de piezas y centrándose en las propiedades esenciales.
3. **Herencia**: La clase BasicPiezaAjedrez es una implementación concreta del concepto de pieza de ajedrez, heredada de una interfaz PiezaAjedrez más genérica o una clase abstracta (no se muestra en el código proporcionado).
4. **Polimorfismo**: el método GestorTiempoRealPGN() de la clase ControladorAjedrez utiliza el polimorfismo para crear diferentes tipos de piezas de ajedrez en función de los parámetros de entrada, sin necesidad de controlar cada tipo de pieza explícitamente.
5. **Separación de intereses**: el código separa las responsabilidades de ControladorAjedrez (control de la entrada del usuario, lógica del juego y coordinación) de ModeloTableroAjedrez (administración del estado del tablero de ajedrez) y VistaTableroAjedrez(representación del tablero de ajedrez).
6. **Inserción de dependencias**: la clase ControladorAjedrez toma los parámetros ModeloTableroAjedrez, VistaTableroAjedrezy GestorTiempoRealPGN como constructores, lo que facilita las pruebas, el mantenimiento y la flexibilidad en el diseño general del sistema.
7. **Principio de responsabilidad única**: La clase ControladorAjedrez tiene la responsabilidad única de administrar el juego de ajedrez en general, delegando tareas específicas a otras clases como ValidadorMovimientoAjedrez y GestorTiempoRealPGN.
8. **Principio de Apertura/Cierre**: El código está diseñado para ser abierto para la extensión, lo que permite la adición de nuevos tipos de piezas de ajedrez o incluso nuevos modos de juego, sin modificar la base de código existente.

Estos principios de POO ayudan a que el código sea más fácil de mantener, flexible y extensible. Al separar las preocupaciones, encapsular los datos y la lógica, y seguir las mejores prácticas de diseño orientado a objetos, el archivo de ControladorAjedrez.java demuestra una implementación de aplicación de ajedrez bien estructurada y modular.