Universidad de Los Andes

Alejandro Hernández (202111716)
Pablo Pedreros (202112491)
Luis Ernesto Tejón (202113150)

DPOO

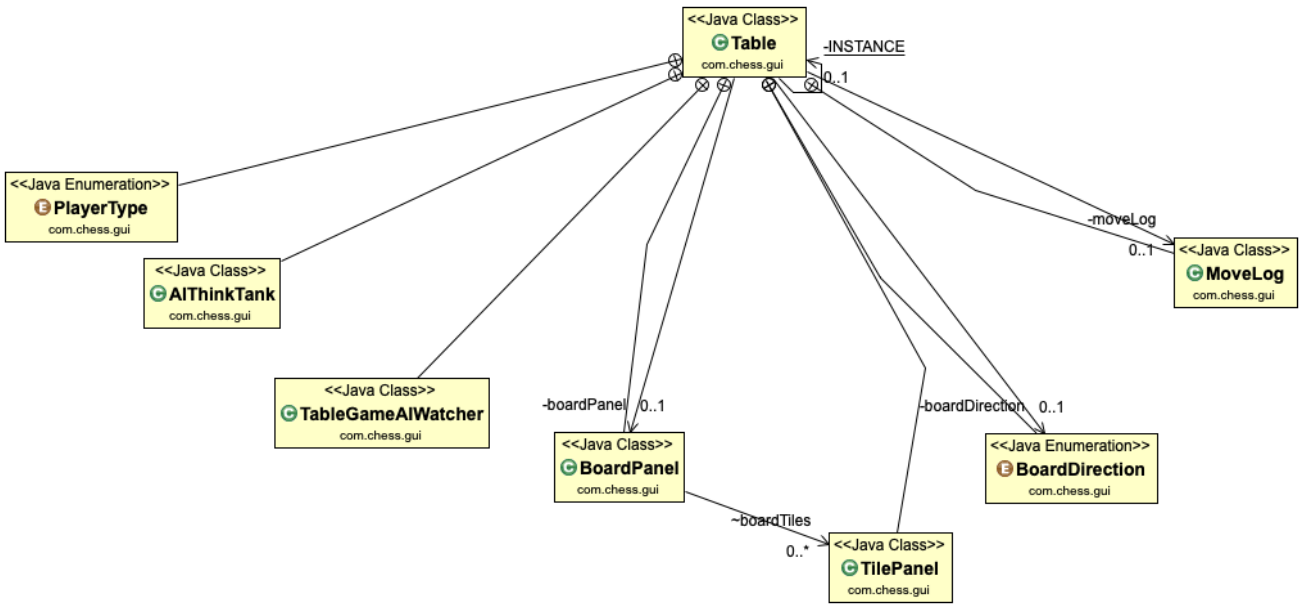
Patrón escogido: Singleton

Nombre del Proyecto de GIT: BlackWidow-Chess de amir 650
URL: https://github.com/amir650/BlackWidow-Chess

Información general y estructura del proyecto:

Este proyecto es un módulo de ajedrez con interfaz gráfica. En el paquete "gui" tiene la clase Table en la cual se define un singleton Table, es decir una clase estática con una instancia única con la que todas las demás clases trabajan.

Como podemos ver en el UML todas las clases están delegadas desde Table, y puesto que Table no tiene lógica alguna más que su constructor vemos que tiene como rol coordinator, pues delega a todas las demás clases labores de imprimir las piezas en pantalla y delega también a las clases del paquete "engine" el cálculo de las jugadas. La estructura del programa gira alrededor de esta instancia y es la siguiente:



Información del patrón:

Este patrón **singleton** se usa en casos en los que tenemos que asegurarnos de que una clase solo se instancie una sola vez, lo normal es usar el patrón en manejadores, coordinadores o cualquier otra clase que solo necesite instanciarse una vez en todo el programa y que esa instancia se conecte con el resto del proyecto.

En este caso estamos usando el patrón en la clase **Table**, pues en el programa va a haber una única instancia de la clase tablero que será la que se conecta y coordina todas las otras clases del programa. En este caso el patrón nos asegura que solo habrá un tablero desde el comienzo hasta el final de la aplicación y que no se crearán nuevos tableros que reemplacen al anterior o entren en conflicto de alguna forma. Adicionalmente, el patrón hará el programa más simple, pues será un único objeto el que se modifica para todo el programa.

La única desventaja que vemos es que, al no crearse nuevas instancias, si hay algún tipo de daño en el objeto nunca se actualizará para que se arregle, sino que toda la funcionalidad de la aplicación dejará de servir. Otra solución podría haber sido crear un nuevo objeto de tipo tablero cada vez que se quiere hacer cambios en la partida o crear nuevas partidas, lo que podría lograr la misma funcionalidad pero crearía muchas más clases y haría el código más propenso a errores.