Juan Diego Sánchez
202214625

TALLER 5

1. **Patrón *Abstract Factory***: Este patrón se utiliza por la facilidad que proporciona para crear familias de objetos relacionados. Esto debido a que se utiliza una clase abstracta o una interfaz para trabajar con familias de objetos. Debido a que cumple el Open-Closed Principle los proyectos son más adaptables y se permite agregar nuevas funcionalidades o familias de objetos sin modificar el código existente. Adicionalmente, hace que se creen objetos que sean consistentes entre ellos, lo que evita posibles errores futuros. No obstante, el patrón debería utilizarse con cuidado de forma que mejore la adaptabilidad del código. Sin olvidar de que puede hacer que el código sea menos eficiente que otros patrones de diseño que requieran de la creación de objetos innecesarios.

2. Elegí este repositorio para el análisis del patrón https://github.com/amir650/BlackWidow-Chess.git. Este repositorio corresponde a un juego de ajedrez escrito en Java para utilizar un . Y que adicionalmente, considero usa el patrón elegido (*Abstract Factory*).

3. De esta forma, dentro de la carpeta *Chess/src/pieces/* se puede observar la implementación de las piezas del tablero. La cual utiliza la clase abstracta *Piece* que se utiliza como base para las otras piezas. Cómo se ve en el diagrama de clases realizado por los autores del código. Figura 1. Queda claro que este diseño sigue el patrón escogido ya que tiene una familia de objetos, que son las piezas. Lo que tiene mucho sentido y considero una buena decisión de diseño ya que todas las piezas del ajedrez tienen diferencias en importancia y mecánicas de juego, pero se deberían mover igual. Adicionalmente, si se desea implementar nuevas mecánicas de juego o modificar el movimiento de solo algunas piezas la implementación actual lo permite muy bien.

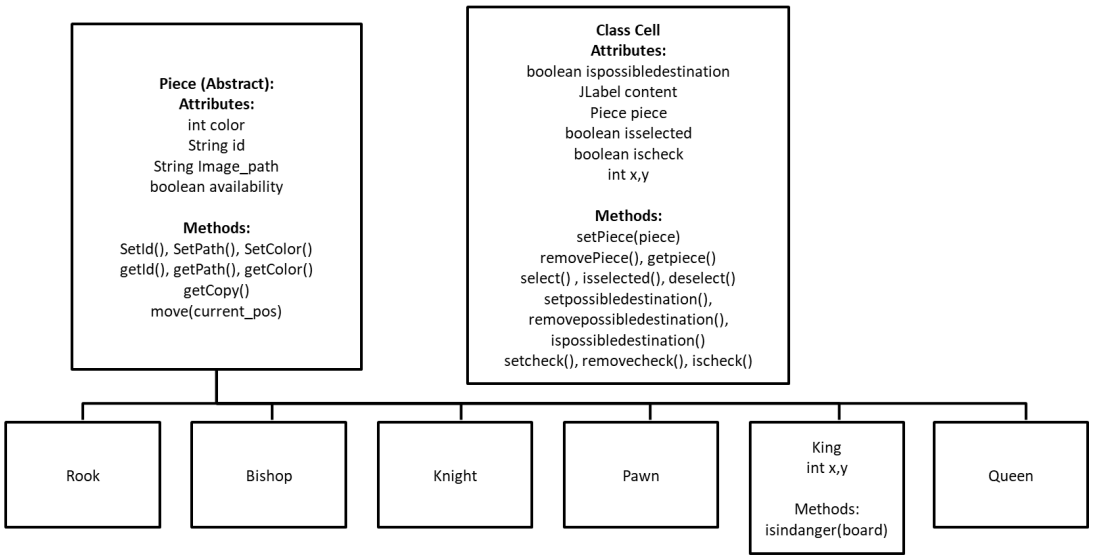

Figura 1. Diagrama de clases en el repositorio