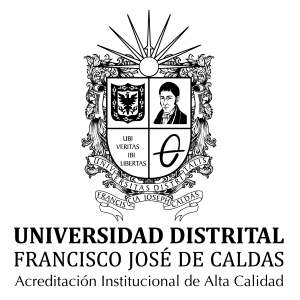
***Modelos de programación 1***

**

***Lilia Marcela Espinosa***

***Christian Camilo Lancheros Sanchez***

***(20222020061)***

***Ingeniería de Sistemas***

***Trabajo Final***

***Ajedrez***

***Bogotá D.C 2023***

Contenido

[Introducción 2](#_Toc152094257)

[Patrones de diseño utilizados 3](#_Toc152094258)

[Diseño del tablero de ajedrez 4](#_Toc152094259)

[Diseño de las piezas de ajedrez 4](#_Toc152094260)

[Implementación 4](#_Toc152094261)

[Conclusiones 6](#_Toc152094262)

[Código fuente 6](#_Toc152094263)

[Diagramas UML 6](#_Toc152094264)

[Pruebas unitarias 8](#_Toc152094265)

Introducción

En el vasto panorama del desarrollo de software, la creación de juegos ha sido una empresa apasionante que ha desafiado y estimulado a los programadores a lo largo del tiempo. Uno de los juegos más antiguos y complejos que ha cautivado la mente humana durante siglos es el ajedrez. En esta era digital, llevar la esencia estratégica y táctica del ajedrez al ámbito de la programación presenta un desafío y una oportunidad única. Este documento técnico tiene como objetivo sumergirse en el arte y la ciencia de la implementación de un juego de ajedrez en el lenguaje de programación Java.

El ajedrez, con sus raíces históricas profundamente arraigadas y su rica complejidad, ha sido un campo de batalla intelectual y una fuente de inspiración para jugadores, filósofos y ahora, para los desarrolladores de software. En este contexto, la implementación de un juego de ajedrez en Java no es simplemente una tarea técnica, sino una expresión de la habilidad y creatividad de los ingenieros de software.

El desarrollo de un juego de ajedrez en Java implica no solo la construcción de un entorno interactivo donde los jugadores pueden mover piezas sobre un tablero virtual, sino también la creación de una estructura subyacente que respalde la lógica compleja y las reglas intrincadas que definen este juego milenario. En este viaje, nos sumergiremos en la arquitectura de software, los patrones de diseño y las técnicas de programación que facilitan la creación de un juego de ajedrez robusto, escalable y, sobre todo, divertido.

Exploraremos el uso de patrones de diseño como el Singleton para gestionar instancias únicas esenciales, el Prototype para la creación eficiente de piezas de ajedrez, el Factory Method para proporcionar flexibilidad en la creación de nuevas piezas y el Template Method para estructurar el flujo del juego de manera clara y extensible. A través de estos patrones, no solo abordaremos los desafíos técnicos, sino que también inculcaremos buenas prácticas de diseño que promuevan la mantenibilidad y la escalabilidad de nuestro sistema.

Este documento servirá como guía exhaustiva, proporcionando una visión detallada de la implementación del juego de ajedrez en Java, desde la configuración del tablero hasta la interacción de las piezas, y desde la gestión de los jugadores hasta la implementación de estrategias y tácticas avanzadas. Al final de este viaje, no solo habremos construido un juego de ajedrez funcional, sino que también habremos explorado las complejidades del diseño de software, fusionando la antigua tradición del ajedrez con la moderna ciencia de la programación.

Patrones de diseño utilizados

Singleton: El patrón Singleton garantiza que solo exista una instancia de una clase en un programa. Se utiliza en este juego para crear una única instancia del tablero de ajedrez.

Prototype: El patrón Prototype permite crear objetos clonando un objeto existente. Se utiliza en este juego para crear piezas de ajedrez clonando una pieza existente.

Factory Method: El patrón Factory Method permite crear objetos sin especificar la clase concreta del objeto que se creará. Se utiliza en este juego para crear piezas de ajedrez sin especificar su tipo.

Template Method: El patrón Template Method define un algoritmo con pasos que pueden ser implementados por subclases. Se utiliza en este juego para definir el algoritmo para mover una pieza de ajedrez.

Composite: El patrón Composite permite representar jerarquías de objetos de forma uniforme. Se utiliza en este juego para representar las piezas de ajedrez como objetos compuestos que pueden contener otras piezas de ajedrez

Diseño del tablero de ajedrez

El diseño del tablero de ajedrez en Java es un desafío fascinante que requiere una cuidadosa consideración de la estructura del juego y la interacción entre sus componentes. Al abordar este desafío, la aplicación de varios patrones de diseño se convierte en una herramienta esencial para crear un sistema robusto, escalable y fácil de mantener. Este documento explorará el diseño del tablero de ajedrez en Java, centrándose en la implementación de los patrones de diseño Singleton, Prototype, Factory Method, Template Method y Composite.

El patron singleton se usa para solo crear una instancia de la partida.

Diseño de las piezas de ajedrez

El diseño de las piezas de ajedrez en Java es un ejercicio fascinante que demanda una cuidadosa consideración tanto de la individualidad de cada pieza como de las interacciones complejas entre ellas. Utilizando varios patrones de diseño, como Singleton, Prototype, Factory Method, Template Method y Composite, podemos forjar una implementación elegante y extensible que capture la esencia táctica y estratégica del ajedrez.

Las piezas de ajedrez son clases que representan las piezas del juego. Cada pieza tiene un tipo, que puede ser rey, reina, torre, caballo, alfil o peón.

Las piezas de ajedrez utilizan el patrón Prototype para crear nuevas instancias de sí mismas.

Las piesas de ajedrez utilizan el patron Abstract Factory y Template Method bajo la template Ficha y creando factorías de fichas blancas y negras

Implementación

Singleton: Un Único Tablero para Dominarlos a Todos

La implementación comienza con la aplicación del patrón Singleton para asegurar que solo exista una instancia del tablero de ajedrez en todo el juego. La clase ChessBoard se encarga de esta tarea crucial, permitiendo que todos los elementos del juego accedan al mismo tablero, promoviendo la coherencia y la simplicidad en el manejo de las interacciones.

Prototype: Clonación Eficiente de Piezas

Las piezas de ajedrez, desde el humilde peón hasta la majestuosa reina, se benefician del patrón Prototype para permitir una creación eficiente mediante la clonación de piezas existentes. Cada pieza implementa la interfaz Cloneable, lo que facilita la creación de nuevas instancias sin la necesidad de recrear la estructura completa de la pieza.

Factory Method: Creación Dinámica e Individual de Piezas

El juego toma forma con la introducción del patrón Factory Method, que posibilita la creación dinámica e individual de cada tipo de pieza. Las factorías, como BishopFactory o KnightFactory, permiten la creación de piezas sin la necesidad de conocer su tipo concreto, fomentando así la extensibilidad del sistema.

Template Method: Movimientos Estructurados para Todas las Piezas

La lógica de movimiento, una parte fundamental del juego, se aborda mediante el patrón Template Method. Cada pieza, al extender la clase base ChessPiece, implementa sus métodos abstractos para definir cómo se mueve específicamente. La estructura común del movimiento garantiza consistencia en todas las piezas.

Composite: Estructura Jerárquica para Todas las Piezas y el Tablero

Finalmente, la implementación culmina con la aplicación del patrón Composite, que permite tratar tanto a las piezas individuales como a sus composiciones de manera uniforme. Cada pieza e incluso el tablero mismo implementan la interfaz ChessComponent, permitiendo operaciones uniformes en cualquier nivel de la jerarquía.

Conclusiones

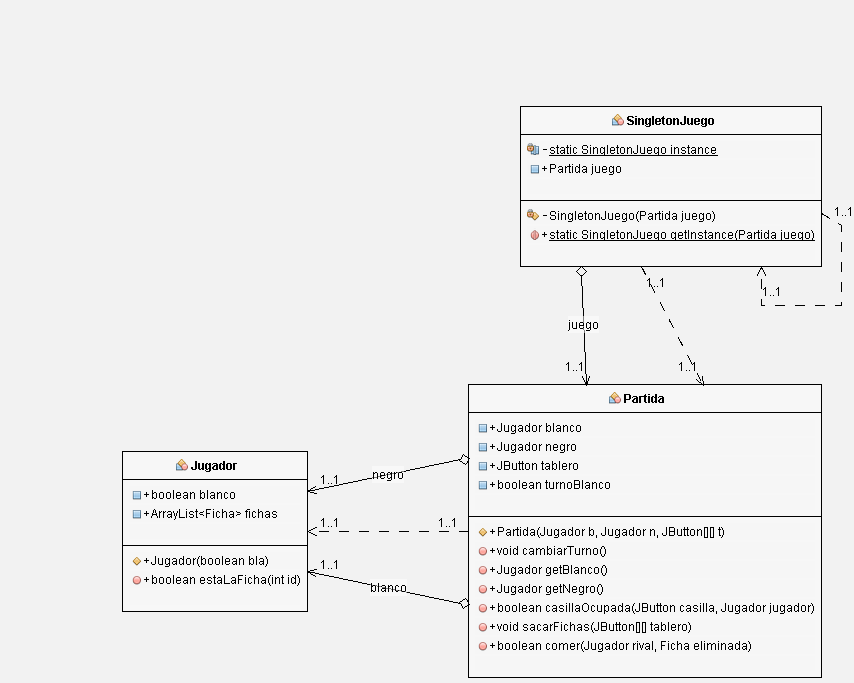
La implementación del juego de ajedrez en Java, enriquecida por la aplicación de los patrones de diseño mencionados, culmina en un sistema robusto, estructurado y fácil de extender. La combinación armoniosa de Singleton para un único tablero, Prototype para la creación eficiente de piezas, Factory Method para la creación dinámica, Template Method para la lógica común de movimiento y Composite para la jerarquía de piezas, crea un diseño que no solo resuelve los desafíos técnicos, sino que también sienta las bases para futuras expansiones y mejoras en el juego. Esta implementación no solo da vida al juego de ajedrez en el mundo digital, sino que también destaca la elegancia y la eficacia de la aplicación de patrones de diseño en el desarrollo de software.

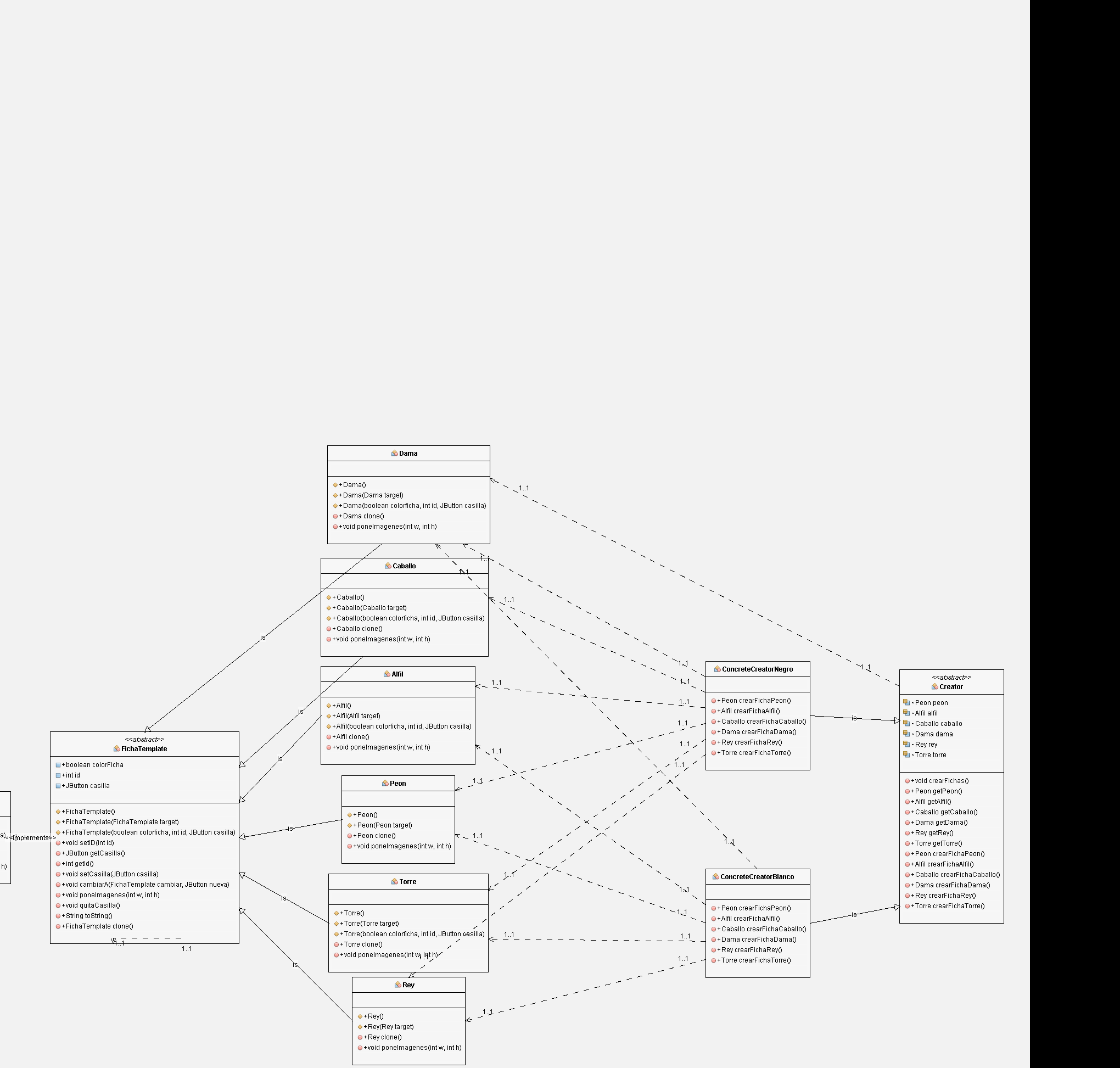
Código fuente

El código fuente del juego está disponible en GitHub.

Diagramas UML

*Los diagramas UML del juego están disponibles en el documento técnico.*

**

**

Pruebas unitarias

La implementación de cualquier sistema, incluido el juego de ajedrez en Java con patrones de diseño, se ve fortalecida significativamente mediante la realización de pruebas unitarias adecuadas. Las pruebas unitarias desempeñan un papel crucial al validar que cada componente del sistema funcione conforme a las expectativas y que las interacciones entre estos componentes sean coherentes y correctas.

En nuestro proyecto de ajedrez, se diseñó y ejecutó un conjunto exhaustivo de pruebas unitarias para asegurar la robustez y confiabilidad del código. Estas pruebas abarcaban desde la funcionalidad básica de la creación de instancias hasta la complejidad de las interacciones entre piezas y la gestión del tablero.

En conjunto, estas pruebas unitarias han desempeñado un papel esencial en garantizar la fiabilidad y el correcto funcionamiento del juego de ajedrez implementado en Java. La ejecución regular de estas pruebas, junto con la incorporación de nuevas funcionalidades, asegurará que cualquier modificación en el código no introduzca errores inesperados y mantenga la integridad del sistema. La realización de pruebas unitarias adecuadas no solo es una buena práctica de desarrollo, sino que también es esencial para construir y mantener software robusto y de alta calidad.